

EBook Gratis

APRENDIZAJE Android

Free unaffiliated eBook created from
Stack Overflow contributors.

#android

Tabla de contenido

Acerca de 1

Capítulo 1: Empezando con Android 2

Observaciones 2

Versiones 2

Examples 3

Configuración de Android Studio 3

Configurar Android Studio 4

Cambiar / agregar tema 4

Compilando apps 4

Creando un Nuevo Proyecto 4

Configurar Android Studio 4

Configure su proyecto 4

Configuracion basica 5

Seleccione los factores de formulario y el nivel de API 6

Añadir una actividad 9

Inspeccionando el proyecto 10

Ejecutando la aplicación 15

Configuración de un dispositivo Android 15

Ejecutando desde Android Studio 15

Ubicación del archivo APK 15

Programación de Android sin un IDE 16

Requisitos y suposiciones 16

Configurando el SDK de Android 16

Codificando la aplicación 17

Construyendo el código 18

Instalación y ejecución 19

Declarar un recurso 20

Desinstalando la aplicación 21

Ver también 21

Fundamentos de la aplicación ... 21
Componentes de la aplicación ... 21
Contexto ... 22
Configuración de un AVD (dispositivo virtual de Android) ... 23
Capítulo 2: ¿Qué es ProGuard? ¿Qué es el uso en Android? ... 29
Introducción ... 29
Examples ... 29
Reduce tu código y recursos con proguard ... 29
Capítulo 3: Accediendo a bases de datos SQLite usando la clase ContentValues ... 31
Examples ... 31
Insertar y actualizar filas en una base de datos SQLite ... 31
Insertando datos ... 31
Actualización de datos ... 31
Capítulo 4: ACRA ... 32
Sintaxis ... 32
Parámetros ... 32
Observaciones ... 32
Examples ... 32
ACRAHandler ... 32
Ejemplo manifiesto ... 33
Instalación ... 33
Capítulo 5: Actividad ... 34
Introducción ... 34
Sintaxis ... 34
Parámetros ... 35
Observaciones ... 35
Examples ... 35
Excluir una actividad del historial de back-stack ... 35
Actividad de Android LifeCycle explicó ... 36
Actividad launchMode ... 39
Estándar: ... 40

SingleTop: 40
SingleTask: 40
Única instancia: 40
Presentando UI con setContentView 40
Ejemplos 41
Establecer contenido desde el archivo de recursos: 41
Establecer contenido a una vista explícita: 41
Borra tu pila de actividades actual y lanza una nueva actividad 42
Finalizar la aplicación con excluir de Recientes 42
Navegación para actividades 43
Capítulo 6: Actividades de pantalla dividida / multipantalla 45
Examples 45
Pantalla dividida introducida en Android Nougat implementado 45
Capítulo 7: ADB (Android Debug Bridge) 47
Introducción 47
Observaciones 47
Examples 47
Imprimir lista detallada de dispositivos conectados 47
Ejemplo de salida 47
Leer información del dispositivo 48
Ejemplo completo de salida 48
Conecta ADB a un dispositivo a través de WiFi 51
Dispositivo no rooteado 51
Dispositivo rooteado 52
Cuando tienes un dispositivo rooteado pero no tienes acceso a un cable USB 52
Evitar el tiempo de espera 53
Tire de (empuje) archivos desde (hacia) el dispositivo 53
Reiniciar dispositivo 53
Encender / apagar Wifi 54
Ver dispositivos disponibles 54
Conectar dispositivo por IP 54
Iniciar / detener adb 55

Ver logcat 55
Dirigir el comando ADB a un dispositivo específico en una configuración de múltiples dispo 56
Captura de pantalla y video (solo para kitkat) desde la pantalla del dispositivo 57
Captura de pantalla: Opción 1 (adb puro) 57
Captura de pantalla: Opción 2 (más rápido) 58
Vídeo 58
Borrar datos de la aplicación 59
Enviando transmisión 59
Instalar y ejecutar una aplicación 60
Apoyo 60
Instalar ADB en el sistema Linux 61
Listar todos los permisos que requieren la concesión de tiempo de ejecución de los usuario 61
Ver los datos internos de una aplicación (datos / datos /) en un dispositivo 61
Ver pila de actividades 62
Ver y extraer archivos de caché de una aplicación 62
Capítulo 8: AdMob 64
Sintaxis 64
Parámetros 64
Observaciones 64
Examples 64
Implementar 64
Build.gradle en el nivel de aplicación 64
Manifiesto 64
XML 65
Java 65
Capítulo 9: Advertencias de la pelusa 67
Observaciones 67
Documentación oficial: 67
Examples 67
Usando herramientas: ignorar en archivos xml 67
Importando recursos sin error "En desuso" 67

Configurar LintOptions con gradle ... 68
Cómo configurar el archivo lint.xml ... 69
Configuración de la comprobación de pelusas en archivos fuente de Java y XML ... 69
Configurando la comprobación de pelusas en Java ... 70
Configurando la comprobación de pelusas en XML ... 70
Marca suprimir advertencias ... 70
Capítulo 10: AIDL ... 72
Introducción ... 72
Examples ... 72
Servicio AIDL ... 72
Capítulo 11: AlarmManager ... 74
Examples ... 74
Ejecutar una intención en un momento posterior ... 74
Cómo cancelar una alarma ... 74
Creando alarmas exactas en todas las versiones de Android ... 75
El modo API23 + Doze interfiere con AlarmManager ... 75
Capítulo 12: Almacenamiento de archivos en almacenamiento interno y externo ... 77
Sintaxis ... 77
Parámetros ... 77
Examples ... 77
Uso de almacenamiento interno ... 77
Uso de almacenamiento externo ... 78
Android: Almacenamiento interno y externo - Aclaración de terminología ... 79
Guardar base de datos en la tarjeta SD (Copia de seguridad de base de datos en SD) ... 84
Fetch Directorio de dispositivos: ... 85
Capítulo 13: Añadiendo un FuseView a un proyecto de Android ... 88
Introducción ... 88
Examples ... 88
aplicación hikr, solo otro android.view.View ... 88
Capítulo 14: Android NDK ... 97
Examples ... 97
Construyendo ejecutables nativos para Android ... 97

Cómo limpiar la construcción ... 98
Cómo usar un makefile que no sea Android.mk ... 98
Cómo iniciar sesión en ndk ... 98
Capítulo 15: Android Studio ... 100
Examples ... 100
Filtrar los registros de la interfaz de usuario ... 100
Crear configuración de filtros ... 101
Colores personalizados del mensaje logcat basado en la importancia del mensaje ... 103
Activar / Desactivar copia de línea en blanco ... 104
Atajos útiles de Android Studio ... 105
Android Studio Mejorar la punta de rendimiento ... 107
Configurar Android Studio ... 107
Ver y agregar accesos directos en Android Studio ... 108
Proyecto de construcción Gradle toma para siempre ... 109
Crear carpeta de activos ... 110
Capítulo 16: Android Vk Sdk ... 112
Examples ... 112
Inicialización y login ... 112
Capítulo 17: Android-x86 en VirtualBox ... 114
Introducción ... 114
Examples ... 114
Configuración de la máquina virtual ... 114
Configuración de disco duro virtual para soporte de SDCARD ... 114
Instalación en partición ... 117
Capítulo 18: Animadores ... 121
Examples ... 121
Agitar la animación de un ImageView ... 121
Fade in / out animación ... 122
Animación transitionDrawable ... 122
ValueAnimator ... 123
ObjectAnimator ... 124
ViewPropertyAnimator ... 125

Expandir y contraer la animación de la vista....125
Capítulo 19: Anotaciones Typedef: @IntDef, @StringDef....127
Observaciones....127
Examples....127
Anotaciones IntDef....127
Combinando constantes con banderas....128
Capítulo 20: API de Android Places....129
Examples....129
Ejemplo de uso del selector de lugar....129
Obtener lugares actuales utilizando la API de lugares....130
Integración automática de lugares....131
Agregando más de una actividad de google auto complete....132
Configuración de filtros de tipo de lugar para PlaceAutocomplete....133
Capítulo 21: API de conocimiento de Google....135
Observaciones....135
Examples....135
Obtenga la actividad actual del usuario utilizando la API de instantáneas....136
Obtener el estado de los auriculares con la API de instantáneas....136
Obtener ubicación actual utilizando API de instantáneas....136
Obtener lugares cercanos utilizando API de instantáneas....136
Obtener el clima actual utilizando API de instantáneas....137
Obtén cambios en la actividad del usuario con Fence API....137
Obtenga cambios para la ubicación dentro de un cierto rango usando la API de Fence....138
Capítulo 22: API de Google Drive....141
Introducción....141
Observaciones....141
Examples....141
Integrar Google Drive en Android....141
Crear un archivo en Google Drive....152
Controlador de resultados de DriveContents....152
Crear archivo programáticamente....153
Manejar el resultado del archivo creado....154

Capítulo 23: API de Google Maps v2 para Android 155
Parámetros 155
Observaciones 155
Examples 155
Actividad predeterminada de Google Map 155
Estilos de mapas de Google personalizados 156
Añadiendo marcadores a un mapa 166
MapView: incrustar un mapa de Google en un diseño existente 167
Mostrar ubicación actual en un mapa de Google 169
Obtención de la huella digital SH1 de su archivo de almacén de claves de certificado 175
No inicie Google Maps cuando se hace clic en el mapa (modo lite) 176
UISettings 176
Obtener debug SHA1 huella digital 177
InfoWindow Click Listener 178
Cambiar Offset 180
Capítulo 24: API de la cámara 2 181
Parámetros 181
Observaciones 181
Examples 182
Vista previa de la cámara principal en un TextureView 182
Capítulo 25: API de Twitter 191
Examples 191
Crear login con el botón de twitter y adjuntarle una devolución 191
Capítulo 26: API de Youtube 193
Observaciones 193
Examples 193
Lanzamiento de StandAlonePlayerActivity 193
Actividad que extiende YouTubeBaseActivity 193
YoutubePlayerFragmento en retrato Activty 194
API de reproductor de YouTube 197
Consumiendo API de datos de YouTube en Android 199
Capítulo 27: Archivo zip en android 203

Examples ... 203
Archivo zip en Android ... 203
Capítulo 28: Arquitectura MVP ... 205
Introducción ... 205
Observaciones ... 205
Definición de MVP ... 205
Estructura de aplicación recomendada (no requerida) ... 205
Examples ... 206
Ejemplo de inicio de sesión en el patrón de Model View Presenter (MVP) ... 206
Diagrama de clase ... 209
Notas: ... 210
Ejemplo de inicio de sesión simple en MVP ... 210
Estructura del paquete requerido ... 210
XML activity_login ... 211
Actividad Clase LoginActivity.class ... 212
Creando una interfaz ILoginView ... 213
Creando una interfaz ILoginPresenter ... 214
ILoginPresenter.class ... 214
LoginPresenterCompl.class ... 214
Creando un UserModel ... 215
UserModel.class ... 215
Clase de usuario ... 215
MVP ... 216
Capítulo 29: AsyncTask ... 218
Parámetros ... 218
Examples ... 218
Uso básico ... 218
Ejemplo ... 218
Uso: ... 219
Nota ... 219

Cancelando AsyncTask ... 220
Nota ... 221
Progreso de publicación ... 221
Descarga la imagen usando AsyncTask en Android ... 221
Entendiendo Android AsyncTask ... 222
Descarga de imágenes usando Android AsyncTask ... 222
Pase la actividad como WeakReference para evitar pérdidas de memoria ... 225
Orden de ejecución ... 226
AsyncTask: Ejecución en serie y ejecución paralela de tareas ... 227
THREAD_POOL_EXECUTOR ... 227
SERIAL_EXECUTOR ... 227
Tarea ejecutada en grupo de subprocesos (1) ... 229
Capítulo 30: AudioManager ... 231
Examples ... 231
Solicitud de enfoque de audio transitorio ... 231
Solicitando Audio Focus ... 231
Capítulo 31: Autentificador de Android ... 232
Examples ... 232
Servicio Autenticador de Cuenta Básico ... 232
Capítulo 32: AutocompletarTextView ... 235
Observaciones ... 235
Examples ... 235
Autocompletar, autocompletar, ver texto ... 235
Autocompletar con CustomAdapter, ClickListener y Filter ... 235
Diseño principal: activity_main.xml ... 235
Diseño de fila row.xml ... 236
strings.xml ... 236
MainActivity.java ... 236
Clase de modelo: People.java ... 237
Clase de adaptador: PeopleAdapter.java ... 238
Capítulo 33: Autosize TextViews ... 240
Introducción ... 240

Examples ... 240
Granularidad ... 240
Tamaños preestablecidos ... 241
Capítulo 34: Barra de progreso ... 242
Observaciones ... 242
Examples ... 242
Barra de progreso indeterminado ... 242
Barra de progreso determinada ... 242
Barra de progreso personalizada ... 244
Barra de progreso de tintado ... 247
Material Linear ProgressBar ... 248
Indeterminado ... 249
Determinado ... 249
Buffer ... 250
Indeterminado y determinado ... 250
Creación de un diálogo de progreso personalizado ... 251
Capítulo 35: Base de datos en tiempo real de Firebase ... 253
Observaciones ... 253
Otros temas relacionados: ... 253
Examples ... 253
Controlador de eventos Firebase Realtime DataBase ... 253
Configuración rápida ... 254
Diseño y comprensión de cómo recuperar datos en tiempo real de la base de datos de Firebas ... 254
Paso 1: Crea una clase llamada Chat ... 255
Paso 2: Crea algunos datos JSON ... 255
Paso 3: Añadiendo los oyentes ... 255
Paso 4: Agregar datos a la base de datos ... 256
Ejemplo ... 257
Desnormalización: Estructura de base de datos plana ... 257
Entendiendo la base de datos JSON de base de fuego ... 260
Recuperando datos de base de fuego ... 261

Escuchando actualizaciones de niños 262
Recuperando datos con paginación 263
Capítulo 36: Base de fuego 265
Introducción 265
Observaciones 265
Firebase - Documentación extendida: 265
Otros temas relacionados: 265
Examples 265
Crear un usuario de Firebase 265
Iniciar sesión en Firebase usuario con correo electrónico y contraseña 266
Enviar correo electrónico de restablecimiento de contraseña de Firebase 268
Actualización del correo electrónico de un usuario de Firebase 269
Cambia la contraseña 270
Volver a autenticar al usuario de Firebase 271
Operaciones de almacenamiento de Firebase 273
Firebase Cloud Messaging 279
Configurar Firebase y el FCM SDK 279
Edita tu manifiesto de aplicación 279
Agrega Firebase a tu proyecto de Android 281
Agrega Firebase a tu aplicación 281
Agrega el SDK 281
Firebase Realtime Database: cómo configurar / obtener datos 282
Demostración de notificaciones basadas en FCM 284
Firebase cerrar sesión 292
Capítulo 37: Biblioteca de enlace de datos 293
Observaciones 293
Examples 293
Enlace de campo de texto básico 293
Encuadernación con un método de acceso 295
Clases de referencia 295
Encuadernación de datos en Fragmento 296

Enlace de datos bidireccional incorporado ... 297
Enlace de datos en el adaptador RecyclerView ... 298
Modelo de datos ... 298
Diseño XML ... 298
Clase de adaptador ... 298
Click listener con Binding ... 299
Evento personalizado usando la expresión lambda ... 300
Valor por defecto en enlace de datos ... 302
Enlace de datos con variables personalizadas (int, booleano) ... 303
Encuadernación de datos en diálogo ... 303
Pase el widget como referencia en BindingAdapter ... 304
Capítulo 38: Bluetooth Low Energy ... 305
Introducción ... 305
Examples ... 305
Buscando dispositivos BLE ... 305
Conectando a un servidor GATT ... 306
Escritura y lectura de características ... 306
Suscripción a notificaciones desde el servidor Gatt ... 307
Publicidad de un dispositivo BLE ... 308
Usando un servidor Gatt ... 308
Capítulo 39: Bluetooth y Bluetooth LE API ... 311
Observaciones ... 311
Examples ... 311
Permisos ... 311
Compruebe si Bluetooth está habilitado ... 311
Hacer que el dispositivo sea detectable ... 312
Encuentra dispositivos bluetooth cercanos ... 312
Conectar a dispositivo Bluetooth ... 313
Encuentra dispositivos Bluetooth de baja energía cercanos ... 315
Capítulo 40: Botón ... 320
Sintaxis ... 320
Examples ... 320

en línea enClickListener ... 320
Usando el diseño para definir una acción de clic ... 320
Usando el mismo evento de clic para una o más Vistas en el XML ... 321
Escuchando los eventos de clic largo ... 321
Definiendo el oyente externo ... 321
¿Cuándo debo usarlo? ... 322
Personalizado Click Listener para evitar múltiples clics rápidos ... 322
Personalizar estilo de botón ... 323
Capítulo 41: Botón de acción flotante ... 328
Introducción ... 328
Parámetros ... 328
Observaciones ... 328
Documentación oficial: ... 328
Especificaciones de materiales de diseño: ... 329
Examples ... 329
Cómo agregar el FAB al diseño ... 329
Mostrar y ocultar el botón de acción flotante al deslizar ... 330
Mostrar y ocultar el botón de acción flotante en el desplazamiento ... 332
Ajuste del comportamiento de FloatingActionButton ... 335
Capítulo 42: Caché de mapa de bits ... 336
Introducción ... 336
Sintaxis ... 336
Parámetros ... 336
Examples ... 336
Caché de mapa de bits utilizando caché LRU ... 336
Capítulo 43: Camara y galeria ... 338
Examples ... 338
Tomando fotos de tamaño completo de la cámara ... 338
AndroidManifest.xml ... 338
Tomar foto ... 340
Cómo iniciar la cámara o la galería y guardar el resultado de la cámara en el almacenamien ... 343
Establecer la resolución de la cámara ... 346

Decodifique el mapa de bits correctamente girado desde el uri obtenido con la intención 346
Capítulo 44: Cambios de orientación 350
Observaciones 350
Examples 350
Ahorro y restauración del estado de actividad 350
Guardando y restaurando el estado del fragmento 351
Fragmentos de retención 352
Orientación de la pantalla de bloqueo 353
Gestionar manualmente los cambios de configuración 353
Manejo AsyncTask 354
Problema: 354
Solución: 354
Ejemplo: 354
Actividad principal: 354
AsyncTaskLoader: 355
Nota: 355
Bloquear la rotación de la pantalla programáticamente 355
Capítulo 45: Captura de capturas de pantalla 357
Examples 357
Captura de captura de pantalla a través de Android Studio 357
Captura de captura de pantalla a través del monitor del dispositivo Android 357
Captura de pantalla de captura a través de ADB 358
Captura de captura de pantalla a través de ADB y guardando directamente en tu PC 358
Tomando una captura de pantalla de una vista particular 358
Capítulo 46: CardView 360
Introducción 360
Parámetros 360
Observaciones 361
Documentación oficial: 361
Examples 361
Empezando con CardView 361
Personalizando el CardView 363

Añadiendo animación de rizo ... 363
Uso de imágenes como fondo en CardView (problemas con el dispositivo Pre-Lollipop) ... 364
Animar CardView color de fondo con TransitionDrawable ... 366
Capítulo 47: Cargador ... 367
Introducción ... 367
Parámetros ... 367
Observaciones ... 367
Cuando no usar cargadores ... 367
Examples ... 368
AsyncTaskLoader básico ... 368
AsyncTaskLoader con caché ... 369
Recarga ... 370
Pasar parámetros utilizando un paquete ... 371
Capítulo 48: Cargador de Imagen Universal ... 372
Observaciones ... 372
Examples ... 372
Inicializar Universal Image Loader ... 372
Uso básico ... 372
Capítulo 49: Cargando Bitmaps Efectivamente ... 374
Introducción ... 374
Sintaxis ... 374
Examples ... 374
Cargue la imagen desde el recurso desde el dispositivo Android. Usando intenciones ... 374
Capítulo 50: carril rápido ... 377
Observaciones ... 377
Examples ... 377
Archivo rápido para crear y cargar múltiples versiones a Beta by Crashlytics ... 377
Fastfile lane para crear e instalar todos los sabores para un tipo de compilación dado en ... 380
Capítulo 51: Ciclo de vida de la interfaz de usuario ... 381
Examples ... 381
Guardar datos en el recorte de memoria ... 381

Capítulo 52: Cifrado / descifrado de datos 382
Introducción 382
Examples 382
Cifrado AES de datos mediante contraseña de forma segura 382
Capítulo 53: CleverTap 384
Introducción 384
Observaciones 384
Examples 384
Obtener una instancia del SDK para grabar eventos 384
Configuración del nivel de depuración 384
Capítulo 54: Colores 385
Examples 385
Manipulación de color 385
Capítulo 55: Comenzando con OpenGL ES 2.0+ 386
Introducción 386
Examples 386
Configurando GLSurfaceView y OpenGL ES 2.0+ 386
Compilación y vinculación de sombreadores GLSL-ES desde un archivo de activos 387
Capítulo 56: Cómo almacenar contraseñas de forma segura 389
Examples 389
Usando AES para el cifrado de contraseña salada 389
Capítulo 57: Cómo utilizar SparseArray 393
Introducción 393
Observaciones 393
Examples 394
Ejemplo básico utilizando SparseArray 394
Capítulo 58: Componentes de la arquitectura de Android 396
Introducción 396
Examples 396
Añadir componentes de arquitectura 396
Usando Lifecycle en AppCompatActivity 396

ViewModel con transformaciones de LiveData ... 397
Habitación peristence ... 398
LiveData personalizado ... 400
Componente personalizado de ciclo de vida ... 401
Capítulo 59: Compresión de imagen ... 403
Examples ... 403
Cómo comprimir la imagen sin cambio de tamaño ... 403
Capítulo 60: Compruebe la conectividad a internet ... 406
Introducción ... 406
Sintaxis ... 406
Parámetros ... 406
Observaciones ... 406
Examples ... 406
Compruebe si el dispositivo tiene conectividad a internet ... 406
¿Cómo comprobar la fuerza de la red en Android? ... 407
Cómo comprobar la fuerza de la red ... 407
Capítulo 61: Compruebe la conexión de datos ... 411
Examples ... 411
Comprobar conexión de datos ... 411
Compruebe la conexión utilizando ConnectivityManager ... 411
Use los intentos de la red para realizar tareas mientras se permiten los datos ... 411
Capítulo 62: Conexiones Wi-Fi ... 412
Examples ... 412
Conectar con cifrado WEP ... 412
Conectar con cifrado WPA2 ... 412
Escanear en busca de puntos de acceso ... 413
Capítulo 63: Configuración de Jenkins CI para proyectos de Android ... 416
Examples ... 416
Enfoque paso a paso para configurar Jenkins para Android ... 416
PARTE I: configuración inicial en su máquina ... 416
PARTE II: configurar Jenkins para construir trabajos de Android ... 417

Parte III: crea un trabajo de Jenkins para tu proyecto de Android ... 418
Capítulo 64: Construyendo aplicaciones compatibles hacia atrás ... 420
Examples ... 420
Cómo manejar API en desuso ... 420
Alternativa más fácil: usar la biblioteca de soporte ... 421
Capítulo 65: Contador regresivo ... 423
Parámetros ... 423
Observaciones ... 423
Examples ... 423
Creando un simple temporizador de cuenta regresiva ... 423
Un ejemplo más complejo ... 423
Capítulo 66: Contexto ... 426
Introducción ... 426
Sintaxis ... 426
Observaciones ... 426
Examples ... 426
Ejemplos básicos ... 426
Capítulo 67: Conversión de voz a texto ... 428
Examples ... 428
Discurso a texto con el diálogo predeterminado de solicitud de Google ... 428
Discurso a texto sin diálogo ... 429
Capítulo 68: Convertir cadena vietnamita a la cadena inglesa Android ... 431
Examples ... 431
ejemplo: ... 431
Chuyn chui Ting Vit thành chui không du ... 431
Capítulo 69: Coordinador de Aula y Comportamientos ... 432
Introducción ... 432
Observaciones ... 432
Examples ... 432
Creando un comportamiento simple ... 432
Extender el CoordinatorLayout.Behavior ... 432

Adjuntar un comportamiento programáticamente 433
Adjuntar un comportamiento en XML 433
Adjuntar un comportamiento automáticamente 433
Usando el comportamiento de SwipeDismiss 434
Crear dependencias entre vistas 434
Capítulo 70: Cosas de Android 436
Examples 436
Controlando un Servo Motor 436
Capítulo 71: Crea ROMs personalizadas de Android 438
Examples 438
¡Preparando su máquina para construir! 438
Instalando Java 438
Instalando Dependencias Adicionales 438
Preparando el sistema para el desarrollo. 438
Capítulo 72: Creación de superposición (siempre en la parte superior) de Windows 440
Examples 440
Superposición de ventanas emergentes 440
Asignando una vista al WindowManager 440
Concesión del permiso SYSTEM_ALERT_WINDOW en Android 6.0 y superior 441
Capítulo 73: Creación de vistas personalizadas 442
Examples 442
Creación de vistas personalizadas 442
Agregando atributos a las vistas 444
Creando una vista compuesta 447
Consejos de rendimiento de CustomView 450
Vista compuesta para SVG / VectorDrawable as drawableRight 451
Nombre del módulo: custom_edit_drawable (nombre corto para prefijo-c_d_e) 451
construir.gradle 451
Archivo de diseño: c_e_d_compound_view.xml 452
Atributos personalizados: attrs.xml 452
Código: EditTextWithDrawable.java 452

Ejemplo: Cómo usar la vista superior 453
Diseño: activity_main.xml 453
Actividad: MainActivity.java 454
Respondiendo a los eventos táctiles 454
Capítulo 74: Creando pantalla de bienvenida 456
Observaciones 456
Examples 456
Una pantalla de bienvenida básica 456
Pantalla de bienvenida con animación 458
Paso 1: Crea una animación. 458
Paso 2: Crear una actividad 458
Paso 3: Reemplazar el lanzador predeterminado 459
Capítulo 75: Creando tus propias bibliotecas para aplicaciones de Android 461
Examples 461
Creando proyecto de biblioteca 461
Uso de biblioteca en proyecto como módulo 462
Crear una biblioteca disponible en Jitpack.io 462
Capítulo 76: Crear una clase Singleton para un mensaje de Toast 464
Introducción 464
Sintaxis 464
Parámetros 464
Observaciones 464
Examples 465
Crea tu propia clase de singleton para masajes de tostadas 465
Capítulo 77: Cuadro de diálogo animado de alerta 467
Introducción 467
Examples 467
Poner código debajo para diálogo animado ... 467
Capítulo 78: Cuchillo de mantequilla 470
Introducción 470
Observaciones 470

Cuchillo de mantequilla ... 470
Examples ... 470
Configurando ButterKnife en tu proyecto ... 471
Encuadernar vistas usando ButterKnife ... 473
Vistas obligatorias ... 473
Vistas obligatorias en actividad ... 473
Encuadernación de vistas en fragmentos ... 473
Encuadernar vistas en diálogos ... 474
Vistas vinculantes en ViewHolder ... 474
Recursos vinculantes ... 474
Encuadernación de listas de vistas ... 474
Fijaciones opcionales ... 475
Oidores obligatorios usando ButterKnife ... 475
Vistas sin compromiso en ButterKnife ... 476
Android Studio ButterKnife Plugin ... 477
Capítulo 79: Cuentas y AccountManager ... 479
Examples ... 479
Comprensión de cuentas personalizadas / autenticación ... 479
Capítulo 80: Daga 2 ... 482
Sintaxis ... 482
Observaciones ... 482
Examples ... 482
Configuración de componentes para inyección de aplicación y actividad ... 482
Alcances personalizados ... 484
Inyección Constructor ... 484
Usar @Subcomponent en lugar de @Component (dependencias = {...}) ... 485
Cómo agregar Dagger 2 en build.gradle ... 486
Creando un componente a partir de múltiples módulos ... 486
Capítulo 81: Defina el valor del paso (incremento) para la barra de barras personalizada ... 489
Introducción ... 489
Observaciones ... 489

Examples ... 490
Definir un valor de paso de 7 ... 490
Capítulo 82: Desarrollo de juegos para Android ... 491
Introducción ... 491
Observaciones ... 491
Examples ... 491
Juego usando Canvas y SurfaceView ... 491
Capítulo 83: Descomprimir archivo en Android ... 498
Examples ... 498
Descomprimir archivo ... 498
Capítulo 84: Deslizamiento ... 499
Introducción ... 499
Observaciones ... 499
Examples ... 499
Agrega Glide a tu proyecto ... 499
Cargando una imagen ... 500
ImageView ... 500
RecyclerView y ListView ... 501
Transformación del círculo deslizante (Cargar imagen en una vista de imagen circular) ... 501
Transformaciones por defecto ... 502
Imagen de esquinas redondeadas con objetivo Glide personalizado ... 503
Precarga de imagenes ... 503
Marcador de posición y manejo de errores ... 504
Cargar imagen en un ImageView circular sin transformaciones personalizadas ... 504
Falló la carga de la imagen de Glide ... 505
Capítulo 85: Deslizar para actualizar ... 507
Sintaxis ... 507
Examples ... 507
Deslizar para actualizar con RecyclerView ... 507
Cómo agregar Swipe-to-Refresh a tu aplicación ... 507
Capítulo 86: Detección de gestos ... 509
Observaciones ... 509

Examples ... 509
Detección de deslizamiento ... 509
Detección de gestos básicos ... 510
Capítulo 87: Detect Shake Event en Android ... 512
Examples ... 512
Shake Detector en el ejemplo de Android ... 512
Usando detección de sacudidas sísmicas ... 513
Instalación ... 513
Capítulo 88: Diálogo ... 514
Parámetros ... 514
Observaciones ... 514
Examples ... 514
Diálogo de alerta ... 514
Un diálogo de alerta básica ... 515
Selector de fecha dentro de DialogFragment ... 515
DatePickerDialog ... 517
Selector de fechas ... 518
Ejemplo de uso de DatePickerDialog ... 518
Adición de Material Design AlertDialog a su aplicación usando Appcompat ... 519
ListView en AlertDialog ... 520
Cuadro de diálogo de alerta personalizada con EditText ... 521
Cuadro de diálogo personalizado a pantalla completa sin fondo y sin título ... 522
Cuadro de diálogo de alerta con título de multilínea ... 522
Capítulo 89: Dibujables ... 525
Examples ... 525
Tintar un dibujo ... 525
Hacer ver con esquinas redondeadas ... 525
Vista circular ... 526
Dibujo personalizable ... 527
Capítulo 90: Dibujos vectoriales ... 529
Introducción ... 529
Parámetros ... 529

Observaciones ... 529
Examples ... 530
Ejemplo de uso de VectorDrawable ... 530
Ejemplo de VectorDrawable xml ... 531
Importando archivo SVG como VectorDrawable ... 531
Capítulo 91: Diseño de materiales ... 534
Introducción ... 534
Observaciones ... 534
Examples ... 534
Aplicar un tema de AppCompat ... 534
Agregar una barra de herramientas ... 535
Agregando un FloatingActionButton (FAB) ... 537
Botones de estilo con Material Design ... 538
Cómo utilizar TextInputLayout ... 539
Añadiendo un TabLayout ... 540
RippleDrawable ... 542
Añadir un cajón de navegación ... 547
Hojas inferiores en la biblioteca de soporte de diseño ... 550
Hojas inferiores persistentes ... 551
Hoja inferior DialogFragment ... 552
Añadir un Snackbar ... 554
Capítulo 92: Diseños ... 556
Introducción ... 556
Sintaxis ... 556
Observaciones ... 556
LayoutParams y Layout_ Attributes ... 556
Impacto en el rendimiento del uso de RelativeLayouts cerca de la parte superior de la jera ... 557
Examples ... 558
LinearLayout ... 558
Disposición relativa ... 559
Gravedad y diseño de gravedad ... 561
Diseño de cuadrícula ... 564

Porcentaje de diseños ... 566
FrameLayout ... 567
CoordinatorLayout ... 568
CoordinatorLayout Scrolling Behavior ... 569
Ver peso ... 571
Creando LinearLayout programáticamente ... 573
LayoutParams ... 574
Capítulo 93: Editar texto ... 578
Examples ... 578
Trabajando con EditTexts ... 578
Personalizando el tipo de entrada ... 580
atributo `inputype` ... 581
Ocultar SoftKeyboard ... 583
Icono o botón dentro de Texto de edición personalizado y su acción y haga clic en escuchas ... 583
Capítulo 94: Ejecución instantánea en Android Studio ... 586
Observaciones ... 586
Examples ... 586
Habilitar o deshabilitar la ejecución instantánea ... 586
Tipos de swaps de código en ejecución instantánea ... 589
Cambios de código no admitidos al usar la ejecución instantánea ... 589
Capítulo 95: El archivo de manifiesto ... 591
Introducción ... 591
Examples ... 591
Declarando componentes ... 591
Declarando permisos en su archivo manifiesto ... 592
Capítulo 96: Emulador ... 593
Observaciones ... 593
Examples ... 593
Tomando capturas de pantalla ... 593
Abra el administrador de AVD ... 596
Simular llamada ... 597
Resolviendo errores al iniciar el emulador ... 597

Capítulo 97: Entrenador de animales 599
Observaciones 599
Examples 599
Uso de un controlador para ejecutar código después de un período de tiempo retrasado 599
HandlerThreads y comunicación entre hilos. 599
Creación de un controlador para el hilo actual 599
Creación de un controlador para el subproceso principal (subproceso de la interfaz de usua 600
Enviar un Runnable de otro hilo al hilo principal 600
Creando un Handler para otro HandlerThread y enviándole eventos 600
Detener el manejador de la ejecución 600
Use el controlador para crear un temporizador (similar a javax.swing.Timer) 601
Capítulo 98: Escribir pruebas de interfaz de usuario - Android 603
Introducción 603
Sintaxis 603
Observaciones 603
Reglas de JUnit: 603
Apio 603
Parámetros 603
Examples 604
Ejemplo de MockWebServer 604
IdlingResource 606
Implementación 607
NOTAS 607
Ejemplo 607
Uso 608
Combinación con regla JUnit 608
Capítulo 99: Eventos / intenciones de botón de hardware (PTT, LWP, etc.) 610
Introducción 610
Examples 610
Dispositivos Sonim 610
PTT_KEY 610

YELLOW_KEY 610
SOS_KEY 610
GREEN_KEY 610
Registrando los botones 610
Dispositivos RugGear 611
Botón PTT 611
Capítulo 100: Eventos táctiles 612
Examples 612
Cómo variar entre los eventos táctiles de grupo de vista infantil y padre 612
Capítulo 101: Excepciones 616
Examples 616
NetworkOnMainThreadException 616
ActivityNotFoundException 617
Error de memoria insuficiente 617
DexException 618
UncaughtException 618
Registro de manejador propio para excepciones inesperadas 619
Capítulo 102: ExoPlayer 621
Examples 621
Agrega ExoPlayer al proyecto 621
Utilizando ExoPlayer 621
Pasos principales para reproducir video y audio usando las implementaciones estándar de Tr 622
Capítulo 103: Facebook SDK para Android 623
Sintaxis 623
Parámetros 623
Examples 623
Cómo agregar Facebook Login en Android 623
Configuración de permisos para acceder a los datos desde el perfil de Facebook 625
Crea tu propio botón personalizado para iniciar sesión en Facebook 626
Una guía minimalista para la implementación de inicio de sesión / registro en Facebook 627
Cerrar sesión de Facebook 628

Capítulo 104: Facturación en la aplicación ... 629
Examples ... 629
Compras consumibles en la aplicación ... 629
Pasos en resumen: ... 629
Paso 1: ... 629
Paso 2: ... 629
Paso 3: ... 629
Etapa 4: ... 630
Paso 5: ... 630
Paso 6: ... 633
(Tercero) In-App v3 Library ... 634
Capítulo 105: Fastjson ... 636
Introducción ... 636
Sintaxis ... 636
Examples ... 636
Analizando JSON con Fastjson ... 636
Convertir los datos de tipo Map to JSON String ... 638
Capítulo 106: Fecha / Hora localizada en Android ... 639
Observaciones ... 639
Examples ... 639
Formato de fecha personalizado localizado con DateUtils.formatDateTime () ... 639
Formato de fecha / hora estándar en Android ... 639
Fecha / hora totalmente personalizada ... 639
Capítulo 107: FileIO con Android ... 641
Introducción ... 641
Observaciones ... 641
Examples ... 641
Obtención de la carpeta de trabajo. ... 641
Escritura de matriz en bruto de bytes ... 641
Serialización del objeto. ... 642
Escritura en almacenamiento externo (tarjeta SD) ... 642

Resolviendo el problema de "archivos MTP invisibles"....643
Trabajando con archivos grandes....643
Capítulo 108: FileProvider....645
Examples....645
Compartiendo un archivo....645
Especifique los directorios en los que se ubican los archivos que desea compartir.....645
Defina un FileProvider y vincúlelo con las rutas del archivo....645
Generar el URI para el archivo....646
Comparte el archivo con otras aplicaciones....646
Capítulo 109: Firebase Cloud Messaging....647
Introducción....647
Examples....647
Configurar una aplicación de cliente de mensajería en la nube Firebase en Android....647
Token de registro....648
Este código que he implementado en mi aplicación para enviar imágenes, mensajes y también 648
Recibir mensajes....649
Suscribirse a un tema....650
Capítulo 110: Firebase Crash Reporting....652
Examples....652
Cómo agregar Firebase Crash Reporting a tu aplicación....652
Cómo reportar un error....653
Capítulo 111: Firma tu aplicación de Android para su lanzamiento....654
Introducción....654
Examples....654
Firma tu aplicación....654
Configurar el build.gradle con la configuración de firma....655
Capítulo 112: Formato de cadenas....657
Examples....657
Formato de un recurso de cadena....657
Formato de una marca de tiempo a la cadena....657
Formateo de tipos de datos a cadena y viceversa....657

Capítulo 113: Formato de números de teléfono con patrón. 658
Introducción 658
Examples 658
Patrones + 1 (786) 1234 5678 658
Capítulo 114: Fragmentos 659
Introducción 659
Sintaxis 659
Observaciones 660
Constructor 660
Examples 660
El patrón newInstance () 660
Navegación entre fragmentos usando backstack y patrón de tela estático 662
Pasa los datos de la Actividad al Fragmento usando Bundle 663
Enviar eventos de nuevo a una actividad con interfaz de devolución de llamada 663
Ejemplo 663
Enviar devolución de llamada a una actividad, cuando se hace clic en el botón del fragment 663
Animar la transición entre fragmentos 664
Comunicación entre fragmentos 665
Capítulo 115: Fresco 671
Introducción 671
Observaciones 671
Examples 671
Empezando con Fresco 671
Usando OkHttp 3 con Fresco 672
Streaming JPEG con Fresco utilizando DraweeController 672
Capítulo 116: Fuentes personalizadas 674
Examples 674
Poner una fuente personalizada en tu aplicación 674
Inicializando una fuente 674
Usando una fuente personalizada en un TextView 674
Aplicar fuente en TextView por xml (No requiere código Java) 674
Fuente personalizada en texto lienzo 675

Tipografía eficiente cargando 676
Fuente personalizada para toda la actividad 676
Trabajando con fuentes en Android O 677
Capítulo 117: Genymotion para android 679
Introducción 679
Examples 679
Instalando Genymotion, la versión gratuita 679
Paso 1 - instalando VirtualBox 679
Paso 2 - descargando Genymotion 679
Paso 3 - Instalando Genymotion 679
Paso 4 - Instalando los emuladores de Genymotion 679
Paso 5 - Integración de genymotion con Android Studio 679
Paso 6 - Ejecutando Genymotion desde Android Studio 680
Marco de Google en Genymotion 680
Capítulo 118: Gerente de empaquetación 681
Examples 681
Recuperar la versión de la aplicación 681
Nombre de la versión y código de la versión 681
Instalar tiempo y tiempo de actualización 681
Método de utilidad utilizando PackageManager 682
Capítulo 119: Google Play Store 684
Examples 684
Abra el listado de Google Play Store para su aplicación 684
Abra Google Play Store con la lista de todas las aplicaciones de su cuenta de editor 684
Capítulo 120: Gradle para Android 686
Introducción 686
Sintaxis 686
Observaciones 686
Gradle para Android - Documentación extendida: 687
Examples 687
Un archivo build.gradle básico 687

DSL (lenguaje específico de dominio) 687
Complementos 688
Entendiendo los DSLs en el ejemplo anterior 688
Dependencias 688
Especificando dependencias específicas para diferentes configuraciones de compilación 689
firmaConfig 690
Definición de sabores de producto 690
Adición de dependencias específicas del sabor del producto 691
Añadiendo recursos específicos del sabor del producto 691
Definir y usar los campos de configuración de construcción 692
BuildConfigField 692
Valorar 693
Centralizando dependencias a través del archivo "dependencies.gradle" 695
Otro enfoque 696
Estructura de directorio para recursos específicos de sabor 696
¿Por qué hay dos archivos build.gradle en un proyecto de Android Studio? 697
Ejecutando un script de shell desde gradle 697
Depurando tus errores de Gradle 698
Especificar diferentes ID de aplicación para tipos de compilación y sabores de producto 699
Firmar APK sin exponer la contraseña del keystore 700
Método A: configure la firma de liberación utilizando un archivo keystore.properties 700
Método B: utilizando una variable de entorno 701
Versiones de sus compilaciones a través del archivo "version.properties" 702
Cambiar el nombre del apk de salida y agregar el nombre de la versión: 703
Deshabilite la compresión de imágenes para un tamaño de archivo APK más pequeño 703
Habilitar Proguard usando gradle 703
Habilitar el soporte experimental del complemento NDK para Gradle y AndroidStudio 704
Configurar el archivo MyApp / build.gradle 704
Configurar el archivo MyApp / app / build.gradle 705
Probar si el plugin está habilitado 705
Mostrar todas las tareas del proyecto Gradle 706

Eliminar "no alineado" apk automáticamente ... 708
Ignorando la variante de construcción ... 708
Viendo arbol de dependencias ... 708
Use gradle.properties para central versionnumber / buildconfigurations ... 709
Mostrar información de firma ... 710
Definiendo tipos de compilación ... 711
Capítulo 121: GreenDAO ... 713
Introducción ... 713
Examples ... 713
Métodos de ayuda para las consultas SELECT, INSERT, DELETE, UPDATE ... 713
Creación de una entidad con GreenDAO 3.X que tiene una clave primaria compuesta ... 715
Empezando con GreenDao v3.X ... 716
Capítulo 122: GreenRobot EventBus ... 719
Sintaxis ... 719
Parámetros ... 719
Examples ... 719
Creando un objeto de evento ... 719
Recibir eventos ... 719
Enviando eventos ... 720
Pasando un evento simple ... 720
Capítulo 123: Gson ... 723
Introducción ... 723
Sintaxis ... 723
Examples ... 724
Analizando JSON con Gson ... 724
Analizar la propiedad JSON para enumerar con Gson ... 726
Analizar una lista con Gson ... 726
Serialización / Deserialización JSON con AutoValue y Gson ... 726
Análisis de JSON a objetos de clase genéricos con Gson ... 727
Añadiendo Gson a tu proyecto ... 728
Usando Gson para cargar un archivo JSON desde el disco ... 729
Agregar un convertidor personalizado a Gson ... 729

Usando Gson como serializador con Retrofit 730
Analizar la matriz json a una clase genérica usando Gson 730
Deserializador JSON personalizado utilizando Gson 731
Usando Gson con herencia 733
Capítulo 124: Herramientas Atributos 736
Observaciones 736
Examples 736
Atributos de diseño en tiempo de diseño 736
Capítulo 125: Herramientas de informes de bloqueo 738
Observaciones 738
Examples 738
Tejido - Crashlytics 738
Cómo configurar Fabric-Crashlytics 738
Usando el plugin IDE de tela 739
Informe de Accidentes con ACRA 743
Forzar un choque de prueba con tela 744
Captura bloqueos utilizando Sherlock 745
Capítulo 126: Hilandero 747
Examples 747
Añadiendo un spinner a tu actividad 747
Ejemplo de Spinner básico 747
Capítulo 127: Hilo 750
Examples 750
Ejemplo de hilo con su descripción 750
Actualización de la interfaz de usuario desde un hilo de fondo 750
Capítulo 128: Hojas inferiores 752
Introducción 752
Observaciones 752
Examples 752
BottomSheetBehavior como los mapas de Google 752
Configuración rápida 759
Hojas inferiores persistentes 759

Hojas inferiores modales con BottomSheetDialogFragment 761
Hojas de fondo modales con BottomSheetDialog 761
Abra BottomSheet DialogFragment en modo Expandido de forma predeterminada 761
Capítulo 129: HttpURLConnection 763
Sintaxis 763
Observaciones 763
Examples 763
Creando una conexión HttpURLC 763
Enviando una solicitud HTTP GET 764
Leyendo el cuerpo de una solicitud HTTP GET 765
Utilice HttpURLConnection para multipart / form-data 765
Enviando una solicitud HTTP POST con parámetros 768
Cargar archivo (POST) utilizando HttpURLConnection 769
Una clase HttpURLConnection multipropósito para manejar todos los tipos de solicitudes HTT 770
Uso 773
Capítulo 130: Huella digital API en Android 774
Observaciones 774
Examples 774
Añadiendo el escáner de huellas dactilares en la aplicación de Android 774
Cómo utilizar la API de huellas digitales de Android para guardar las contraseñas de los u 775
Capítulo 131: Imágenes de 9 parches 785
Observaciones 785
Examples 785
Esquinas redondeadas basicas 785
Hilandero basico 786
Líneas de relleno opcionales 787
Capítulo 132: ImageView 788
Introducción 788
Sintaxis 788
Parámetros 788
Examples 788
Establecer recurso de imagen 788

Establecer alfa ... 789
ImageView ScaleType - Centro ... 790
ImageView ScaleType - CenterCrop ... 791
ImageView ScaleType - CenterInside ... 791
ImageView ScaleType - FitStart y FitEnd ... 791
ImageView ScaleType - FitCenter ... 791
ImageView ScaleType - FitXy ... 791
Establecer el tipo de escala ... 791
Establecer tinte ... 796
MLRoundedImageView.java ... 797
Capítulo 133: Indexación de la aplicación Firebase ... 799
Observaciones ... 799
Examples ... 801
Apoyando URLs HTTP ... 801
Añadir API de AppIndexing ... 802
Capítulo 134: Instalando aplicaciones con ADB ... 805
Examples ... 805
Instalar una aplicación ... 805
Desinstalar una aplicación ... 805
Instalar todo el archivo apk en el directorio ... 805
Capítulo 135: Integración de Android Paypal Gateway ... 806
Observaciones ... 806
Examples ... 806
Configure paypal en su código de Android ... 806
Capítulo 136: Integración de inicio de sesión de Google en Android ... 808
Introducción ... 808
Examples ... 808
Integración de google Auth en tu proyecto. (Obtener un archivo de configuración) ... 808
Implementación de código de Google SignIn ... 808
Capítulo 137: Integrar el inicio de sesión de Google ... 810
Sintaxis ... 810
Parámetros ... 810

Examples ... 810
Google Inicia sesión con la clase de ayuda ... 810
Capítulo 138: Integrar OpenCV en Android Studio ... 814
Observaciones ... 814
Examples ... 814
Instrucciones ... 814
Capítulo 139: Intención ... 823
Introducción ... 823
Sintaxis ... 823
Parámetros ... 824
Observaciones ... 824
Advertencias de usar la intención implícita ... 824
Actividad de inicio que es una singleTask o una singleTask singleTop ... 824
Examples ... 825
Iniciar una actividad ... 825
Pasando datos entre actividades ... 825
OrigenActividad ... 826
DestinoActividad ... 826
Mandando correos electrónicos ... 827
Obtener un resultado de otra actividad ... 828
Actividad principal: ... 828
DetailActividad: ... 829
Algunas cosas que debes tener en cuenta: ... 829
Abre una URL en un navegador ... 830
Apertura con el navegador predeterminado ... 830
Pedir al usuario que seleccione un navegador ... 830
Mejores prácticas ... 831
Borrar una pila de actividades ... 831
Intención URI ... 832
Transmisión de mensajes a otros componentes ... 832
CustomTabsIntent para Chrome Custom Tabs ... 833

Compartiendo múltiples archivos a través de la intención ... 833
Patrón de arranque ... 834
Inicia el servicio Unbound usando una intención ... 835
Compartir intención ... 835
Iniciar el marcador ... 836
Abrir el mapa de Google con la latitud, longitud especificada ... 837
Pasando diferentes datos a través de Intención en Actividad ... 837
Mostrar un selector de archivos y leer el resultado ... 839
Iniciar una actividad de selección de archivos ... 839
Leyendo el resultado ... 840
Pasando objeto personalizado entre actividades ... 841
Parcelable ... 841
Serializable ... 843
Obteniendo un resultado de Actividad a Fragmentar ... 843
Capítulo 140: Intenciones implícitas ... 846
Sintaxis ... 846
Parámetros ... 846
Observaciones ... 846
Examples ... 846
Intenciones implícitas y explícitas ... 846
Intenciones implícitas ... 847
Capítulo 141: Inter-app UI testing con UIAutomator ... 848
Sintaxis ... 848
Observaciones ... 848
Examples ... 848
Prepara tu proyecto y escribe el primer test UIAutomator ... 848
Escribiendo pruebas más complejas usando el UIAutomatorViewer ... 849
Creación de un conjunto de pruebas de pruebas UIAutomator ... 850
Capítulo 142: Interfaces ... 851
Examples ... 851
Oyente personalizado ... 851
Definir interfaz ... 851

Crear oyente ... 851
Implementar oyente ... 851
Oyente del disparador ... 852
Oyente básico ... 853
Capítulo 143: Interfaz nativa de Java para Android (JNI) ... 855
Introducción ... 855
Examples ... 855
Cómo llamar a funciones en una biblioteca nativa a través de la interfaz JNI ... 855
Cómo llamar a un método Java desde código nativo ... 856
Método de utilidad en la capa JNI ... 857
Capítulo 144: Internacionalización y localización (I18N y L10N) ... 859
Introducción ... 859
Observaciones ... 859
Examples ... 859
Planificación para la localización: habilitar el soporte RTL en Manifiesto ... 859
Planificación para la localización: Añadir soporte RTL en diseños ... 860
Planificación para la localización: diseños de prueba para RTL ... 861
Codificación para localización: creación de cadenas y recursos predeterminados ... 861
Codificación para localización: Proporcionar cadenas alternativas ... 862
Codificación para localización: Proporcionar diseños alternativos ... 862
Capítulo 145: Jackson ... 864
Introducción ... 864
Examples ... 864
Ejemplo completo de enlace de datos ... 864
Capítulo 146: Java en Android ... 866
Introducción ... 866
Examples ... 866
Java 8 cuenta con subconjunto con Retrolambda ... 866
Capítulo 147: JCodec ... 869
Examples ... 869
Empezando ... 869

Obtención de fotograma de la película ... 869
Capítulo 148: JSON en Android con org.json ... 870
Sintaxis ... 870
Observaciones ... 870
Examples ... 870
Parse simple objeto JSON ... 870
Creando un objeto JSON simple ... 871
Añadir JSONArray a JSONObject ... 872
Crear una cadena JSON con valor nulo ... 872
Trabajando con una cadena nula al analizar json ... 872
Uso de JsonReader para leer JSON desde una secuencia ... 873
Crear objeto JSON anidado ... 875
Manejo de clave dinámica para respuesta JSON ... 875
Compruebe la existencia de campos en JSON ... 876
Actualizando los elementos en el JSON ... 877
Capítulo 149: Leakcanary ... 879
Introducción ... 879
Observaciones ... 879
Examples ... 879
Implementando una aplicación de Leak Canary en Android ... 879
Capítulo 150: Lectura de códigos de barras y códigos QR ... 880
Observaciones ... 880
Examples ... 880
Usando QRCodeReaderView (basado en Zxing) ... 880
Agregando la biblioteca a tu proyecto ... 880
Primer uso ... 880
Capítulo 151: Library Dagger 2: Inyección de dependencia en aplicaciones ... 882
Introducción ... 882
Observaciones ... 882
Dagger 2 API: ... 882
Links importantes: ... 882

Examples ... 883
Cree la clase @Module y la anotación @Singleton para el objeto ... 883
Solicitud de dependencias en objetos dependientes ... 883
Conectando @Modules con @Inject ... 883
Usando la interfaz @Component para obtener objetos ... 884
Capítulo 152: Lienzo de dibujo utilizando SurfaceView ... 885
Observaciones ... 885
Examples ... 885
SurfaceView con hilo de dibujo ... 885
Capítulo 153: Localización con recursos en Android. ... 891
Examples ... 891
Moneda ... 891
Añadiendo traducción a tu aplicación de Android ... 891
Tipo de directorios de recursos en la carpeta "res" ... 892
Tipos de configuración y nombres de calificadores para cada carpeta en el directorio "res" ... 893
Lista exhaustiva de todos los diferentes tipos de configuración y sus valores calificadore ... 893
Cambiar la configuración regional de la aplicación de Android programáticamente ... 896
Capítulo 154: Looper ... 901
Introducción ... 901
Examples ... 901
Crear un LooperThread simple ... 901
Ejecutar un bucle con un HandlerThread ... 901
Capítulo 155: LruCache ... 902
Observaciones ... 902
Examples ... 902
Inicializando el caché ... 902
Agregar un mapa de bits (recurso) a la caché ... 902
Obtención de un mapa de bits (respuesta) de la caché ... 903
Capítulo 156: Manejo de enlaces profundos ... 904
Introducción ... 904
Parámetros ... 904
Observaciones ... 904

El <intent-filter> 904
Múltiples etiquetas <data> 905
Recursos 905
Examples 905
Enlace profundo simple 905
Múltiples rutas en un solo dominio 905
Múltiples dominios y múltiples caminos 906
Tanto http como https para el mismo dominio 906
Recuperando parámetros de consulta 907
Usando pathPrefix 907
Capítulo 157: Manejo de eventos táctiles y de movimiento. 909
Introducción 909
Parámetros 909
Examples 909
Botones 909
Superficie 910
Manipulación multitáctil en una superficie 911
Capítulo 158: Mapeo de puertos usando la biblioteca Cling en Android 913
Examples 913
Añadiendo soporte de Cling a tu proyecto de Android 913
Mapeo de un puerto NAT 913
Capítulo 159: MediaSession 915
Sintaxis 915
Observaciones 915
Examples 915
Recepción y manejo de eventos de botones 915
Capítulo 160: Mediastore 918
Examples 918
Obtenga archivos de audio / MP3 de una carpeta específica del dispositivo o busque todos l 918
Ejemplo con Actividad 920
Capítulo 161: Mejora de los diálogos de alerta 922

Introducción ... 922
Examples ... 922
Cuadro de diálogo de alerta que contiene un enlace cliqueable ... 922
Capítulo 162: Mejora del rendimiento de Android utilizando fuentes de iconos ... 923
Observaciones ... 923
Examples ... 923
Cómo integrar fuentes de iconos ... 923
TabLayout con fuentes de iconos ... 926
Capítulo 163: Menú ... 928
Sintaxis ... 928
Parámetros ... 928
Observaciones ... 928
Examples ... 928
Menú de opciones con separadores ... 928
Aplicar fuente personalizada al menú ... 929
Creando un Menú en una Actividad ... 929
Paso 1: ... 929
Paso 2: ... 930
¡Terminando! ... 930
Captura de pantalla de cómo se ve tu propio menú: ... 931
Capítulo 164: Métricas de la pantalla del dispositivo ... 933
Examples ... 933
Consigue las dimensiones de las pantallas en píxeles ... 933
Obtener densidad de pantalla ... 933
Fórmula px a dp, dp a px conversación ... 933
Capítulo 165: Modo Doze ... 935
Observaciones ... 935
Examples ... 937
Excluir la aplicación del uso del modo dormido ... 937
Lista blanca de una aplicación de Android mediante programación ... 938
Capítulo 166: Modo PorterDuff ... 939

Introducción ... 939
Observaciones ... 939
Examples ... 940
Creando un PorterDuff ColorFilter ... 940
Creando un PorterDuff XferMode ... 941
Aplique una máscara radial (viñeta) a un mapa de bits usando PorterDuffXfermode ... 941
Capítulo 167: Moshi ... 942
Introducción ... 942
Observaciones ... 942
Examples ... 942
JSON en Java ... 942
serializar objetos Java como JSON ... 942
Construido en adaptadores de tipo ... 942
Capítulo 168: Multidex y el método Dex Limit ... 944
Introducción ... 944
Observaciones ... 944
¿Qué es dex? ... 944
El problema: ... 944
Qué hacer al respecto: ... 944
Cómo evitar el límite: ... 945
Examples ... 945
Multidex utilizando MultiDexApplication directamente ... 945
Multidex extendiendo la aplicación ... 946
Habilitando Multidex ... 947
Configuracion gradle ... 947
Habilite MultiDex en su aplicación ... 947
Referencias del método de conteo en cada compilación (Dexcount Gradle Plugin) ... 947
Multidex extendiendo MultiDexApplication ... 948
Capítulo 169: MVVM (Arquitectura) ... 950
Observaciones ... 950
Examples ... 951
Ejemplo de MVVM usando la biblioteca de DataBinding ... 951

Capítulo 170: NavigationView ... 959
Observaciones ... 959
Documentación oficial: ... 959
Especificaciones de materiales de diseño: ... 959
Examples ... 959
Cómo agregar el NavigationView ... 959
Añadir subrayado en los elementos del menú ... 964
Añadir separadores al menú ... 965
Agregue el menú Divider usando la opción predeterminada DividerItemDecoration. ... 966
Capítulo 171: Notificacion canal android o ... 968
Introducción ... 968
Sintaxis ... 968
Parámetros ... 968
Examples ... 968
Canal de notificaciones ... 968
Capítulo 172: Notificaciones ... 975
Examples ... 975
Creando una notificación simple ... 975
Especifique el contenido de la notificación: ... 975
Crea la intención de disparar al hacer clic: ... 975
Finalmente, construya la notificación y muéstrela. ... 975
Heads Up Notification with Ticker para dispositivos más antiguos ... 975
Esto es lo que parece en Android Marshmallow con la Notificación de Heads Up: ... 976
Aquí está lo que parece en Android KitKat con el Ticker: ... 977
Android 6.0 Marshmallow: ... 977
Android 4.4.x KitKat: ... 978
Establecer diferentes prioridades en la notificación ... 979
Programación de notificaciones ... 980
Establecer notificación personalizada: muestra el contenido completo del texto ... 981
Por ejemplo, tienes esto: ... 981
Pero deseas que tu texto se muestre completamente: ... 982
Establecer el icono de notificación personalizado usando la biblioteca `Picasso` ... 982

Obtener dinámicamente el tamaño de píxel correcto para el icono grande ... 983
Notificación continua con botón de acción ... 983
Capítulo 173: Obtención de dimensiones de vista calculadas ... 985
Observaciones ... 985
Examples ... 985
Cálculo de dimensiones iniciales de vista en una actividad ... 985
Capítulo 174: Obtención de nombres de fuentes del sistema y uso de las fuentes ... 987
Introducción ... 987
Examples ... 987
Obtención de nombres de fuentes del sistema ... 987
Aplicando una fuente del sistema a un TextView ... 987
Capítulo 175: OkHttp ... 988
Examples ... 988
Interceptor de registro ... 988
Reescritura de respuestas ... 988
Ejemplo de uso básico ... 988
Llamada sincrónica Get ... 989
Asynchronous Get Call ... 989
Parámetros de formulario ... 990
Publicar una solicitud multiparte ... 990
Configurando OkHttp ... 991
Capítulo 176: Okio ... 992
Examples ... 992
Descargar / Implementar ... 992
Decodificador PNG ... 992
ByteStrings y Buffers ... 993
Capítulo 177: Optimización del núcleo de Android ... 994
Examples ... 994
Configuración de RAM baja ... 994
Cómo agregar un regulador de CPU ... 994
Programadores de E / S ... 996
Capítulo 178: Optimización del rendimiento ... 998

Introducción ... 998
Examples ... 998
Guardar búsquedas de vista con el patrón ViewHolder ... 998
Capítulo 179: ORMLite en Android ... 999
Examples ... 999
Ejemplo de Android OrmLite sobre SQLite ... 999
Configuración de Gradle ... 999
Ayudante de base de datos ... 1000
Objeto persistente a SQLite ... 1001
Capítulo 180: Otto Event Bus ... 1004
Observaciones ... 1004
Examples ... 1004
Pasando un evento ... 1004
Recibiendo un evento ... 1005
Capítulo 181: Paginación en RecyclerView ... 1006
Introducción ... 1006
Examples ... 1006
MainActivity.java ... 1006
Capítulo 182: Pantallas de apoyo con diferentes resoluciones, tamaños ... 1011
Observaciones ... 1011
Tamaño de pantalla ... 1011
Densidad de pantalla ... 1011
Orientación ... 1011
Unidades ... 1012
px ... 1012
en ... 1012
mm ... 1012
pt ... 1012
dp o dip ... 1012
sp ... 1012
Examples ... 1013

Uso de calificadores de configuración ... 1013
Convertir dp y sp a píxeles ... 1014
Tamaño del texto y diferentes tamaños de pantalla de Android ... 1014
Capítulo 183: Parcelable ... 1016
Introducción ... 1016
Observaciones ... 1016
Examples ... 1016
Haciendo un objeto personalizado parcelable ... 1016
Objeto parcelable que contiene otro objeto parcelable ... 1017
Usando Enums con Parcelable ... 1018
Capítulo 184: Patrones de diseño ... 1020
Introducción ... 1020
Examples ... 1020
Ejemplo de clase Singleton ... 1020
Patrón observador ... 1021
Implementando el patrón observador. ... 1021
Capítulo 185: Pérdidas de memoria ... 1022
Examples ... 1022
Fugas de memoria comunes y cómo solucionarlos. ... 1022
1. Arregla tus contextos: ... 1022
2. Referencia estática al contexto. ... 1022
3. Comprueba que realmente estás terminando tus servicios. ... 1023
4. Comprobar el uso de la imagen y de los mapas de bits: ... 1023
5. Si está utilizando receptores de difusión, anúltelos. ... 1023
6. Si está utilizando java.util.Observer (patrón de observador): ... 1023
Evite las actividades con fugas con AsyncTask ... 1023
Callback anónimo en actividades ... 1024
Contexto de actividad en clases estáticas ... 1025
Detecta pérdidas de memoria con la biblioteca LeakCanary ... 1026
Evite las actividades de filtración con oyentes ... 1027
Alternativa 1: Eliminar oyentes ... 1029
Alternativa 2: Usar referencias débiles ... 1030

Evite las pérdidas de memoria con la clase anónima, el controlador, la tarea del temporiza....1033
Capítulo 186: Permisos de tiempo de ejecución en API-23 +....1034
Introducción....1034
Observaciones....1034
Examples....1035
Android 6.0 permisos múltiples....1035
Cumplimiento de permisos en difusiones, URI....1036
Permisos de tiempo de ejecución múltiples de los mismos grupos de permisos....1037
Usando PermissionUtil....1039
Incluya todo el código relacionado con permisos para una clase base abstracta y extienda l....1040
Ejemplo de uso en la actividad.....1041
Capítulo 187: Picasso....1043
Introducción....1043
Observaciones....1043
Examples....1043
Añadiendo la biblioteca de Picasso a tu proyecto de Android....1043
Gradle....1043
Maven....1043
Marcador de posición y manejo de errores....1043
Redimensionamiento y rotación....1044
Avatares circulares con Picasso....1044
Deshabilitar el caché en Picasso....1046
Cargando imagen desde almacenamiento externo....1046
Descargando imagen como Bitmap usando Picasso....1046
Cancelando solicitudes de imagen usando Picasso....1047
Usando Picasso como ImageGetter para Html.fromHtml....1047
Pruebe primero la memoria caché del disco sin conexión, luego conéctese y busque la imagen....1049
Capítulo 188: Ping ICMP....1051
Introducción....1051
Examples....1051
Realiza un solo ping....1051

Capítulo 189: Pintar ... 1052
Introducción ... 1052
Examples ... 1052
Creando una pintura ... 1052
Configuración de la pintura para el texto ... 1052
Ajustes de dibujo de texto ... 1052
Texto de medición ... 1053
Configuración de pintura para dibujar formas ... 1053
Poniendo banderas ... 1053
Capítulo 190: Pista de audio ... 1055
Examples ... 1055
Generar tono de una frecuencia específica ... 1055
Capítulo 191: Política de modo estricto: una herramienta para detectar el error en el tiem ... 1056
Introducción ... 1056
Observaciones ... 1056
Examples ... 1056
El siguiente fragmento de código es para configurar StrictMode para políticas de subproces ... 1056
El código siguiente trata las fugas de memoria, como las que se detectan cuando se llama a ... 1056
Capítulo 192: Preferencias compartidas ... 1057
Introducción ... 1057
Sintaxis ... 1057
Parámetros ... 1058
Observaciones ... 1058
Documentacion oficial ... 1058
Examples ... 1058
Leer y escribir valores en SharedPreferences ... 1058
Quitando llaves ... 1059
Implementando una pantalla de configuración usando SharedPreferences ... 1060
Recupere todas las entradas almacenadas de un archivo de SharedPreferences particular ... 1062
Escuchando cambios de SharedPreferences ... 1062
Lectura y escritura de datos en SharedPreferences con Singleton ... 1063
Diferentes formas de instanciar un objeto de SharedPreferences ... 1067

getPreferences (int) VS getSharedPreferences (String, int) 1068
Cometer vs. Aplicar 1068
Tipos de datos soportados en SharedPreferences 1069
Almacenar, recuperar, eliminar y borrar datos de SharedPreferences 1069
Soporte pre-Honeycomb con StringSet 1070
Añadir filtro para EditTextPreference 1071
Capítulo 193: Procesador de anotaciones 1073
Introducción 1073
Examples 1073
@NonNull Annotation 1073
Tipos de anotaciones 1073
Creación y uso de anotaciones personalizadas 1074
Capítulo 194: Programación de Android con Kotlin. 1076
Introducción 1076
Observaciones 1076
Examples 1076
Instalando el plugin de Kotlin 1076
Configurando un proyecto Gradle existente con Kotlin 1077
Creando una nueva actividad de Kotlin 1079
Convertir código Java existente a Kotlin 1081
Comenzando una nueva actividad 1081
Capítulo 195: Programación de trabajos 1082
Observaciones 1082
Examples 1082
Uso básico 1082
Crear un nuevo servicio de empleo 1082
Agregue el nuevo servicio de trabajo a su AndroidManifest.xml 1082
Configura y ejecuta el trabajo 1083
Capítulo 196: ProGuard - ofuscar y encoger su código 1085
Examples 1085
Reglas para algunas de las bibliotecas ampliamente utilizadas 1085
Habilita ProGuard para tu compilación 1087

Eliminar las declaraciones de registro de seguimiento (y otras) en el momento de la compil 1087
Protegiendo su código de hackers 1088
Habilitando ProGuard con un archivo de configuración de ofuscación personalizado 1089
Capítulo 197: Proveedor de contenido 1091
Observaciones 1091
Examples 1091
Implementando una clase de proveedor de contenido básico 1091
Capítulo 198: Prueba de interfaz de usuario con espresso 1096
Observaciones 1096
Café exprés 1096
Solución de problemas 1096
Examples 1096
Preparar espresso 1096
Crear clase de prueba de espresso 1097
Abrir Cerrar CajónDisposición 1097
Prueba de IU simple expreso 1098
Herramientas de prueba de interfaz de usuario 1098
Cómo agregar espresso al proyecto 1099
Configuración de dispositivo 1100
Escribiendo la prueba 1101
Arriba navegación 1103
Realizar una acción en una vista 1103
Encontrar una vista con onView 1104
Cafeteras personalizadas espresso 1104
Espresso general 1106
Introduzca texto en EditarTexto 1108
Realizar Clic en Vista 1108
Se muestra la vista de comprobación 1108
Agrupar una colección de clases de prueba en un conjunto de pruebas 1108
Capítulo 199: Pruebas unitarias en Android con JUnit. 1110
Observaciones 1110
Examples 1110

Creando pruebas unitarias locales ... 1110
Ejemplo de clase de prueba ... 1110
Descompostura ... 1110
Consejo: crea rápidamente clases de prueba en Android Studio ... 1111
Sugerencia: Ejecutar pruebas fácilmente en Android Studio ... 1111
Moviendo la lógica de negocios fuera de los componentes de Android ... 1112
Empezando con JUnit ... 1114
Preparar ... 1114
Escribiendo una prueba ... 1115
Haciendo una prueba ... 1116
Excepciones ... 1117
Importación estática ... 1118
Capítulo 200: Publicar el archivo .aar en Apache Archiva con Gradle ... 1120
Examples ... 1120
Ejemplo de implementación simple ... 1120
Capítulo 201: Publicar en Play Store ... 1122
Examples ... 1122
Guía de envío de aplicaciones mínimas ... 1122
Capítulo 202: Publicar una biblioteca en Repositorios Maven ... 1124
Examples ... 1124
Publicar archivo .aar a Maven ... 1124
Capítulo 203: Receptor de radiodifusión ... 1126
Introducción ... 1126
Examples ... 1126
Introducción al receptor de radiodifusión ... 1126
Fundamentos de BroadcastReceiver ... 1127
Usando LocalBroadcastManager ... 1127
Receptor Bluetooth Broadcast ... 1128
agrega permiso en tu archivo manifiesto ... 1128
En tu Fragmento (o Actividad) ... 1128
Registrar transmisión ... 1128
Anular el registro de transmisión ... 1129

Habilitar y deshabilitar un receptor de difusión programáticamente 1129
BroadcastReceiver para manejar eventos BOOT_COMPLETED 1129
Ejemplo de un LocalBroadcastManager 1130
Comunicar dos actividades a través del receptor Broadcast personalizado 1131
Transmisión pegajosa 1132
Usando transmisiones ordenadas 1132
Android detuvo el estado 1133
Capítulo 204: Recolectores de fecha y hora 1134
Examples 1134
Material DatePicker 1134
Cuadro de diálogo Selector de fecha 1136
Capítulo 205: Reconocimiento de actividad 1138
Introducción 1138
Examples 1138
Actividad de Google PlayReconocimientoAPI 1138
Reconocimiento de la actividad PathSense 1140
Capítulo 206: Recursos 1143
Examples 1143
Traducir una cadena 1143
Definir cuerdas 1144
Definir matriz de cadena 1145
Definir dimensiones 1145
Definir enteros 1146
Definir matriz de enteros 1146
Definir colores 1147
Obteniendo recursos sin advertencias "obsoletas" 1148
Defina un recurso de menú y utilícelo dentro de Actividad / Fragmento 1149
Formato de cadena en cadenas.xml 1150
Definir una lista de estados de color 1151
Definir plurales de cadena 1151
Importar matriz de objetos definidos en recursos 1152
9 parches 1154

GUÍA SENCILLA DE 9-PATCH PARA LA IU DE ANDROID 18 de mayo de 2011 1155
Nivel de transparencia de color (alfa) 1158
Trabajando con el archivo strings.xml 1158
Capítulo 207: RecyclerView 1161
Introducción 1161
Parámetros 1161
Observaciones 1161
Otros temas relacionados: 1162
Documentacion oficial 1162
Versiones anteriores: 1162
Examples 1163
Añadiendo un RecyclerView 1163
Carga más suave de artículos 1164
Arrastrar y soltar y deslizar con RecyclerView 1165
Añadir encabezado / pie de página a un RecyclerView 1166
Usando varios ViewHolders con ItemViewType 1168
Filtrar elementos dentro de RecyclerView con un SearchView 1170
Menú emergente con recyclerView 1170
Animar el cambio de datos 1172
Ejemplo usando SortedList 1174
RecyclerView con DataBinding 1176
Desplazamiento sin fin en Recycleview. 1178
Mostrar vista predeterminada hasta que los elementos se carguen o cuando los datos no esté 1179
Agregue líneas divisorias a los artículos RecyclerView 1181
Capítulo 208: RecyclerView Decoraciones 1184
Sintaxis 1184
Parámetros 1184
Observaciones 1184
Las decoraciones son estáticas. 1184
Decoraciones multiples 1184
Otros temas relacionados: 1184

Oficial javadoc ... 1184
Examples ... 1185
Dibujando un separador ... 1185
Márgenes por artículo con ItemDecoration ... 1186
Añadir divisor a RecyclerView ... 1187
Cómo agregar divisores usando y DividerItemDecoration ... 1189
ItemOffsetDecoration para GridLayoutManager en RecycleView ... 1189
Capítulo 209: RecyclerView onClickListeners ... 1191
Examples ... 1191
Nuevo ejemplo ... 1191
Ejemplo de Kotlin y RxJava ... 1192
Ejemplo fácil de OnLongClick y OnClick ... 1193
Demostración del adaptador ... 1194
Artículo Click Listeners ... 1196
Otra forma de implementar Item Click Listener ... 1197
RecyclerView Click listener ... 1199
Capítulo 210: RecyclerView y LayoutManagers ... 1202
Examples ... 1202
GridLayoutManager con recuento dinámico de span ... 1202
Agregar vista de encabezado a recyclerview con el administrador de gridlayout ... 1204
Lista simple con LinearLayoutManager ... 1206
Diseño de la actividad ... 1206
Definir el modelo de datos. ... 1206
Lista de elementos de diseño ... 1207
Crear un adaptador RecyclerView y ViewHolder ... 1207
(Generar datos aleatorios) ... 1209
Conecte el RecyclerView con el PlaceListAdapter y el conjunto de datos ... 1209
¡Hecho! ... 1210
StaggeredGridLayoutManager ... 1210
Capítulo 211: Registro y uso de Logcat ... 1213
Sintaxis ... 1213

Parámetros 1213
Observaciones 1213
Definición 1213
Cuándo usar 1214
Enlaces útiles 1214
Examples 1214
Filtrado de la salida logcat 1214
Explotación florestal 1216
Registro basico 1216
Niveles de registro 1217
Motivación para la tala 1217
Cosas a tener en cuenta al iniciar sesión: 1218
Legibilidad del registro: 1218
Actuación: 1218
Seguridad: 1218
Conclusión: 1218
Iniciar sesión con un enlace a la fuente directamente desde Logcat 1219
Usando el Logcat 1219
Generando código de registro 1220
Uso de Android Studio 1221
Borrar registros 1224
Capítulo 212: Reino 1225
Introducción 1225
Observaciones 1225
Examples 1225
Agregando Realm a tu proyecto 1225
Modelos de reino 1226
Lista de primitivas (RealmList) 1227
probar con recursos 1228
Consultas ordenadas 1228
Consultas asincrónicas 1229
Usando Realm con RxJava 1229

Uso básico 1230
Configurando una instancia 1230
Cerrando una instancia 1230
Modelos 1231
Inserción o actualización de datos. 1232
Consultar la base de datos 1232
Borrando un objeto 1233
Capítulo 213: RenderScript 1234
Introducción 1234
Examples 1234
Empezando 1234
Configurando tu proyecto 1234
Cómo funciona RenderScript 1235
Escribiendo tu primer RenderScript 1235
Plantilla de RenderScript 1236
Variables globales 1237
Kernels 1237
Kernels en general 1237
Métodos de la API de RenderScript Runtime 1238
Implementacion de Kernel 1238
Llamando a RenderScript en Java 1239
Lo esencial 1239
Creación de instancias de asignación 1240
Ejemplo completo 1241
Conclusión 1242
Desenfocar una imagen 1243
Desenfocar una vista 1245
BlurBitmapTask.java 1245
Uso: 1246
Capítulo 214: Reproductor multimedia 1247
Sintaxis 1247

Observaciones ... 1247
Examples ... 1249
Creación básica y juego ... 1249
Preparación asíncrona ... 1249
Obteniendo tonos del sistema ... 1250
Obtención y configuración del volumen del sistema ... 1251
Tipos de flujo de audio ... 1251
Ajuste de volumen ... 1251
Ajustando el volumen en un solo paso ... 1251
Configuración de MediaPlayer para utilizar un tipo de transmisión específico ... 1252
Reproductor multimedia con progreso de búfer y posición de juego ... 1252
Importar audio en androidstudio y reproducirlo ... 1254
Capítulo 215: RestricciónDisposición ... 1256
Introducción ... 1256
Sintaxis ... 1256
Parámetros ... 1257
Observaciones ... 1257
Para más información sobre el diseño de restricciones: ... 1257
Examples ... 1257
Agregando ConstraintLayout a su proyecto ... 1258
Las cadenas ... 1258
Capítulo 216: RestricciónSet ... 1260
Introducción ... 1260
Examples ... 1260
Restricción establecida con ContraintLayout mediante programación ... 1260
Capítulo 217: Retrofit2 ... 1261
Introducción ... 1261
Observaciones ... 1261
Examples ... 1261
Una simple solicitud GET ... 1261
Añadir registro a Retrofit2 ... 1264
Subiendo un archivo a través de Multipart ... 1265

Reequipamiento con interceptor OkHttp ... 1266
Encabezado y cuerpo: un ejemplo de autenticación ... 1266
Sube múltiples archivos usando Retrofit como multiparte ... 1267
Descarga un archivo del servidor usando Retrofit2 ... 1269
Depurando con stetho ... 1271
Retrofit 2 Custom Xml Converter ... 1272
Una simple solicitud POST con GSON ... 1274
Leyendo la URL del formulario XML con Retrofit 2 ... 1276
Capítulo 218: Retrofit2 con RxJava ... 1279
Examples ... 1279
Retrofit2 con RxJava ... 1279
Reequipamiento con RxJava para obtener datos de forma asíncrona ... 1280
Ejemplo de solicitudes anidadas: solicitudes múltiples, combinar resultados ... 1282
Capítulo 219: RoboGuice ... 1284
Examples ... 1284
Ejemplo simple ... 1284
Instalación para proyectos Gradle ... 1284
@ContentView anotación ... 1284
@InjectResource anotación ... 1284
@InjectView anotación ... 1285
Introducción a RoboGuice ... 1285
Capítulo 220: Robolectric ... 1288
Introducción ... 1288
Examples ... 1288
Prueba robolectrica ... 1288
Configuración ... 1288
Ejecutar con clase de aplicación personalizada ... 1288
Establecer objetivo SDK ... 1288
Ejecutar con manifiesto personalizado ... 1289
Usar calificadores ... 1289
Capítulo 221: SearchView ... 1290
Examples ... 1290

AppCompat SearchView con el observador de RxBindings 1290
SearchView en la barra de herramientas con fragmento 1292
Establecer tema para SearchView 1294
Capítulo 222: Secure SharedPreferences 1296
Introducción 1296
Sintaxis 1296
Parámetros 1296
Observaciones 1296
Examples 1296
Asegurar una preferencia compartida 1296
Capítulo 223: Secure SharedPreferences 1298
Introducción 1298
Sintaxis 1298
Parámetros 1298
Observaciones 1298
Examples 1298
Asegurar una preferencia compartida 1298
Capítulo 224: Seguridad 1300
Examples 1300
Verificación de la firma de la aplicación - Detección de sabotaje 1300
Capítulo 225: SensorManager 1302
Examples 1302
Recuperando eventos del sensor 1302
Transformación del sensor al sistema de coordenadas del mundo. 1303
Decide si tu dispositivo es estático o no, usando el acelerómetro 1303
Capítulo 226: Servicio 1305
Introducción 1305
Observaciones 1305
Examples 1305
Comenzando un servicio 1305
Ciclo de vida de un servicio 1305

Definiendo el proceso de un servicio. . . . 1306
Creando Servicio Bound con ayuda de Binder . . . 1307
Creación de servicio remoto (a través de AIDL) . . . 1308
Creación de un servicio independiente . . . 1310
Capítulo 227: Servicio de Intención . . . 1313
Sintaxis . . . 1313
Observaciones . . . 1313
Examples . . . 1313
Creando un IntentService . . . 1313
Ejemplo de servicio de intenciones . . . 1313
Ejemplo de IntentService Básico . . . 1314
Capítulo 228: shell adb . . . 1316
Introducción . . . 1316
Sintaxis . . . 1316
Parámetros . . . 1316
Examples . . . 1316
Envíe texto, tecla presionada y eventos táctiles al dispositivo Android a través de ADB . . . 1316
Listar paquetes . . . 1318
Otorgar y revocar permisos API 23+ . . . 1318
Imprimir datos de la aplicación . . . 1319
Grabando la pantalla . . . 1319
Cambio de permisos de archivos usando el comando chmod . . . 1320
Establecer fecha / hora a través de adb . . . 1321
Opciones de desarrollador abierto . . . 1322
Generando una transmisión "Boot Complete" . . . 1322
Ver contenido de almacenamiento externo / secundario . . . 1322
matar un proceso dentro de un dispositivo Android . . . 1322
Capítulo 229: ShortcutManager . . . 1324
Examples . . . 1324
Atajos de lanzadores dinámicos . . . 1324
Capítulo 230: Sincronización de datos con el adaptador de sincronización . . . 1325
Examples . . . 1325

Dummy Sync Adapter con proveedor de código auxiliar ... 1325
Capítulo 231: Snackbar ... 1331
Sintaxis ... 1331
Parámetros ... 1331
Observaciones ... 1331
Documentacion oficial ... 1331
Examples ... 1331
Creando un Snackbar simple ... 1332
Snack Bar personalizado ... 1332
Snackbar con devolución de llamada ... 1333
Snackbar personalizado ... 1333
Snackbar vs Tostadas: ¿Cuál debo usar? ... 1334
Snackbar personalizado (no hay necesidad de ver) ... 1335
Capítulo 232: Sonido y Medios Android ... 1336
Examples ... 1336
Cómo escoger imagen y video para api> 19 ... 1336
Reproducir sonidos a través de SoundPool ... 1337
Capítulo 233: SpannableString ... 1339
Sintaxis ... 1339
Examples ... 1339
Añadir estilos a un TextView ... 1339
Multi cadena, con multi color ... 1342
Capítulo 234: SQLite ... 1344
Introducción ... 1344
Observaciones ... 1344
Examples ... 1344
Usando la clase SQLiteOpenHelper ... 1344
Insertar datos en la base de datos ... 1345
Método onUpgrade () ... 1346
Leyendo datos de un cursor ... 1346
Cree un contrato, asistente y proveedor para SQLite en Android ... 1348
Actualizar una fila en una tabla ... 1352

Realizando una Transacción ... 1353
Eliminar fila (s) de la tabla ... 1353
Almacenar imagen en SQLite ... 1354
Crear base de datos desde la carpeta de activos ... 1356
Exportando e importando una base de datos ... 1358
Inserto a granel ... 1359
Capítulo 235: SyncAdapter con periódicamente hacer sincronización de datos ... 1361
Introducción ... 1361
Examples ... 1361
Adaptador de sincronización con cada valor mínimo de solicitud del servidor ... 1361
Capítulo 236: TabLayout ... 1371
Examples ... 1371
Usando un TabLayout sin un ViewPager ... 1371
Capítulo 237: Tarjeta electrónica ... 1372
Examples ... 1372
Tarjeta inteligente de envío y recepción ... 1372
Capítulo 238: Teclado ... 1375
Examples ... 1375
Oculta el teclado cuando el usuario toca cualquier otro lugar en la pantalla ... 1375
Registrar una devolución de llamada para abrir y cerrar el teclado ... 1375
Capítulo 239: Tema DayNight (AppCompat v23.2 / API 14+) ... 1377
Examples ... 1377
Adición del tema DayNight a una aplicación ... 1377
Capítulo 240: Tema, Estilo, Atributo ... 1379
Examples ... 1379
Usa un tema personalizado a nivel mundial ... 1379
Definir colores primarios, primarios oscuros y de acento ... 1379
Usar tema personalizado por actividad ... 1379
Color de desplazamiento superior (API 21+) ... 1380
Color de ondulación (API 21+) ... 1380
Barra de estado de luz (API 23+) ... 1380
Navegación translúcida y barras de estado (API 19+) ... 1381

Color de la barra de navegación (API 21+) 1381
Herencia del tema 1381
Temas múltiples en una aplicación 1382
Cambio de tema para todas las actividades a la vez. 1383
Capítulo 241: TensorFlow 1385
Introducción 1385
Observaciones 1385
Examples 1385
Cómo utilizar 1385
Capítulo 242: TextInputLayout 1387
Introducción 1387
Observaciones 1387
Examples 1387
Uso básico 1387
Errores de manejo 1387
Agregando el conteo de personajes 1388
La visibilidad de la contraseña cambia 1388
TextInputEditText 1389
Personalizando la apariencia de TextInputLayout 1389
Capítulo 243: Texto a voz (TTS) 1391
Examples 1391
Base de texto a voz 1391
Implementación de TextToSpeech en las APIs 1393
Capítulo 244: tostada 1396
Introducción 1396
Sintaxis 1396
Parámetros 1396
Observaciones 1396
Documentación oficial: 1397
Examples 1397
Establecer posición de una tostada 1397
Mostrando un mensaje de brindis 1397

Creando un brindis personalizado ... 1398
Forma segura de subprocesos de mostrar Toast (aplicación amplia) ... 1399
Mostrar mensaje de tostada sobre el teclado suave ... 1400
Hilo seguro de mostrar un mensaje de Toast (para AsyncTask) ... 1400
Capítulo 245: Transiciones de elementos compartidos ... 1401
Introducción ... 1401
Sintaxis ... 1401
Examples ... 1401
Transición de elementos compartidos entre dos fragmentos ... 1401
Capítulo 246: TransitionDrawable ... 1404
Examples ... 1404
Añadir transición o cross-fade entre dos imágenes ... 1404
Paso 1: Crea una transición dibujable en XML ... 1404
Paso 2: Agregue el código para ImageView en su diseño XML para mostrar el dibujo anterior ... 1404
Paso 3: Acceda a la transición XML dibujable en el método onCreate () de su Actividad e in ... 1404
Animar vistas de color de fondo (cambio de color) con TransitionDrawable ... 1405
Capítulo 247: Ubicación ... 1406
Introducción ... 1406
Observaciones ... 1406
Gerente de locación ... 1406
FusedLocationProviderApi ... 1407
Solución de problemas ... 1409
Examples ... 1416
API de ubicación fusionada ... 1416
Ejemplo de uso de la actividad con LocationRequest ... 1416
Ejemplo de uso de Service w / PendingIntent y BroadcastReceiver ... 1418
Solicitando actualizaciones de ubicación usando LocationManager ... 1421
Solicitar actualizaciones de ubicación en un subproceso separado utilizando LocationManage ... 1422
Registrar geofence ... 1424
Obtener la dirección de la ubicación utilizando Geocoder ... 1427
Obteniendo actualizaciones de ubicación en un BroadcastReceiver ... 1427

Capítulo 248: Una forma rápida de configurar Retrolambda en un proyecto de Android. 1429
Introducción 1429
Examples 1429
Configuración y ejemplo de uso: 1429
Capítulo 249: URL de devolución de llamada 1431
Examples 1431
Ejemplo de URL de devolución de llamada con Instagram OAuth 1431
Capítulo 250: Utilidades de tiempo 1433
Examples 1433
Convertir formato de fecha en milisegundos 1433
Para comprobar dentro de un plazo 1434
GetCurrentRealTime 1434
Capítulo 251: Validación de correo electrónico 1436
Examples 1436
Validación de la dirección de correo electrónico 1436
Validación de la dirección de correo electrónico con el uso de patrones 1436
Capítulo 252: VectorDrawable y AnimatedVectorDrawable 1437
Examples 1437
Básico VectorDrawable 1437
Utilizando 1437
etiquetas 1438
AnimatedVectorDrawable básico 1439
Utilizando trazos 1440
Compatibilidad de vectores a través de AppCompat 1442
Capítulo 253: Versiones de android 1444
Observaciones 1444
Examples 1445
Comprobación de la versión de Android en el dispositivo en tiempo de ejecución 1445
Capítulo 254: Versiones de Project SDK 1447
Introducción 1447
Parámetros 1447

Observaciones ... 1447
Examples ... 1448
Definir versiones de proyecto SDK ... 1448
Capítulo 255: Vibración ... 1449
Examples ... 1449
Empezando con la vibración ... 1449
Vibrar indefinidamente ... 1449
Patrones de vibracion ... 1449
Dejar de vibrar ... 1450
Vibrar por una vez ... 1450
Capítulo 256: VideoView ... 1451
Examples ... 1451
VideoView Crear ... 1451
Reproducir video desde la URL con el uso de VideoView ... 1451
Capítulo 257: VideoView optimizado ... 1453
Introducción ... 1453
Examples ... 1453
VideoView optimizado en ListView ... 1453
Capítulo 258: ViewFlipper ... 1465
Introducción ... 1465
Examples ... 1465
ViewFlipper con imagen deslizante ... 1465
Capítulo 259: ViewPager ... 1467
Introducción ... 1467
Observaciones ... 1467
Examples ... 1467
Uso básico de ViewPager con fragmentos ... 1467
ViewPager con TabLayout ... 1469
ViewPager con PreferenceFragment ... 1471
Agregar un ViewPager ... 1472
ViewPager con un indicador de puntos ... 1473
TabLayout anidado en ViewPager ... 1473

TabLayout separado ... 1474
selected_dot.xml ... 1474
default_dot.xml ... 1475
tab_selector.xml ... 1475
Configurar OnPageChangeListener ... 1475
Capítulo 260: Vista de la lista ... 1477
Introducción ... 1477
Observaciones ... 1477
Examples ... 1477
Filtrado con CursorAdapter ... 1477
ArrayAdapter personalizado ... 1478
Un ListView básico con un ArrayAdapter ... 1479
Capítulo 261: Vista de texto ... 1481
Introducción ... 1481
Sintaxis ... 1481
Observaciones ... 1481
Examples ... 1481
Textview con diferentes tamaños de textos ... 1481
Personalización de TextView ... 1481
TextView de Spannable ... 1484
TextView con imagen ... 1486
Strikethrough TextView ... 1486
Tachar todo el texto. ... 1486
Tachar solo partes del texto ... 1486
Personalización de temas y estilos. ... 1487
Hacer que RelativeSizeSpan se alinee hacia arriba ... 1489
Pinchzoom en TextView ... 1491
TextView único con dos colores diferentes ... 1492
Capítulo 262: Vista inferior de la navegación ... 1494
Introducción ... 1494
Observaciones ... 1494

Campo de golf: 1494
Examples 1494
Implementacion basica 1494
Personalización de BottomNavigationView 1495
Manejo de estados habilitados / deshabilitados 1496
Permitiendo más de 3 menús 1496
Capítulo 263: Visualización de anuncios de Google 1498
Examples 1498
Configuración básica de anuncios 1498
Añadiendo anuncio intersticial 1498
Capítulo 264: Voleo 1501
Introducción 1501
Sintaxis 1501
Observaciones 1501
Instalación 1501
Documentacion oficial 1501
Examples 1502
StringRequest básico utilizando el método GET 1502
Cancelar una solicitud 1502
Agregar atributos de tiempo de diseño personalizados a NetworkImageView 1503
Solicita JSON 1504
Agregar encabezados personalizados a sus solicitudes [por ejemplo, para autenticación básí 1504
Clase de ayuda para manejar los errores de volea 1505
Autenticación del servidor remoto usando StringRequest a través del método POST 1506
Usando Volley para peticiones HTTP 1508
Respuesta variable booleana del servidor con solicitud json en volea 1510
Usa JSONArray como cuerpo de solicitud 1511
Capítulo 265: WebView 1513
Introducción 1513
Observaciones 1513
Examples 1513
Los diálogos de alerta de JavaScript en WebView - Cómo hacer que funcionen 1513

Comunicación de Javascript a Java (Android) 1513
Comunicación de Java a Javascript 1515
Ejemplo de marcador abierto 1515
Solución de problemas de WebView mediante la impresión de los mensajes de la consola o la 1516
Imprimiendo mensajes de consola webview a logcat 1516
Depuración remota de dispositivos Android con Chrome 1516
Habilitar la depuración USB en su dispositivo Android 1516
Conecta y descubre tu dispositivo Android 1517
Abrir archivo local / Crear contenido dinámico en Webview 1517
Capítulo 266: Widgets 1518
Observaciones 1518
Examples 1518
Declaración Manifiesta - 1518
Metadatos 1518
Clase AppWidgetProvider 1518
Dos widgets con diferentes diseños de declaración 1519
Crear / Integrar Widget básico utilizando Android Studio 1520
Justo en su aplicación ==> Nuevo ==> Widget ==> Widget de aplicación 1520
Capítulo 267: XMPP registro de inicio de sesión y chat simple ejemplo 1522
Examples 1522
Registro XMPP inicio de sesión y ejemplo básico de chat 1522
Capítulo 268: Xposed 1531
Examples 1531
Creando un Módulo Xposed 1531
Enganchando un método 1531
Creditos 1534

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: android

It is an unofficial and free Android ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Android.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con Android

Observaciones

Si desea obtener más información sobre la configuración de Android Gradle Plugin, consulte la documentación de `android-gradle` .

Si estás interesado en emuladores alternativos, puedes mirar Genymotion . Proporciona un plan gratuito y requiere una menor cantidad de RAM.

Versiones

Versión	Nivel de API	Código de versión	Fecha de lanzamiento
1.0	1	`BASE`	2008-09-23
1.1	2	`BASE_1_1`	2009-02-09
1.5	3	`CUPCAKE`	2009-04-27
1.6	4	`DONUT`	2009-09-15
2.0	5	`ECLAIR`	2009-10-26
2.0.1	6	`ECLAIR_0_1`	2009-12-03
2.1.x	7	`ECLAIR_MR1`	2010-01-12
2.2.x	8	`FROYO`	2010-05-20
2.3	9	`GINGERBREAD`	2010-12-06
2.3.3	10	`GINGERBREAD_MR1`	2011-02-09
3.0.x	11	`HONEYCOMB`	2011-02-22
3.1.x	12	`HONEYCOMB_MR1`	2011-05-10
3.2.x	13	`HONEYCOMB_MR2`	2011-07-15
4.0	14	`ICE_CREAM_SANDWICH`	2011-10-18
4.0.3	15	`ICE_CREAM_SANDWICH_MR1`	2011-12-16
4.1	dieciséis	`JELLY_BEAN`	2012-07-09
4.2	17	`JELLY_BEAN_MR1`	2012-11-13

Versión	Nivel de API	Código de versión	Fecha de lanzamiento
4.3	18	`JELLY_BEAN_MR2`	2013-07-24
4.4	19	`KITKAT`	2013-10-31
4.4W	20	`KITKAT_WATCH`	2014-06-25
5.0	21	`LOLLIPOP`	2014-11-12
5.1	22	`LOLLIPOP_MR1`	2015-03-09
6.0	23	`M` (malvavisco)	2015-10-05
7.0	24	`N` (Turrón)	2016-08-22
7.1	25	`N_MR1` (turrón MR1)	2016-10-04
8.0	26	`O` (Vista previa del desarrollador 4)	2017-07-24

Examples

Configuración de Android Studio

Android Studio es el IDE de desarrollo de Android que es oficialmente compatible y recomendado por Google. Android Studio incluye el Android SDK Manager , que es una herramienta para descargar los componentes de `Android SDK` necesarios para comenzar a desarrollar aplicaciones.

Instalar Android Studio y `Android SDK` herramientas del `Android SDK` :

1. Descarga e instala Android Studio .
2. Descargue las herramientas SDK Tools y SDK Platform más recientes abriendo Android Studio y luego siga las instrucciones de actualización de las herramientas SDK de Android . Debe instalar los últimos paquetes estables disponibles.

   Si necesita trabajar en proyectos antiguos que se crearon con versiones anteriores del SDK, es posible que deba descargar estas versiones también.

Desde Android Studio 2.2, una copia del último OpenJDK viene con la instalación y es el JDK (Java Development Kit) recomendado para todos los proyectos de Android Studio. Esto elimina el requisito de tener instalado el paquete JDK de Oracle. Para utilizar el SDK incluido, proceda de la siguiente manera;

1. Abra su proyecto en Android Studio y seleccione **Archivo> Estructura del proyecto** en la barra de menú.
2. En la página de **ubicación** del **SDK** y en la **ubicación de JDK** , marque la casilla de verificación **Usar JDK incorporado** .
3. Haga clic en **Aceptar** .

Configurar Android Studio

Android Studio proporciona acceso a dos archivos de configuración a través del menú **Ayuda** :

- studio.vmoptions : Personalice las opciones para la Máquina Virtual Java (JVM) de Studio, como el tamaño del almacenamiento dinámico y el tamaño del caché. Tenga en cuenta que en las máquinas con Linux, este archivo puede llamarse *studio64.vmoptions* , dependiendo de su versión de Android Studio.
- idea.properties : Personalice las propiedades de Android Studio, como la ruta de la carpeta de complementos o el tamaño máximo de archivo admitido.

Cambiar / agregar tema

Puedes cambiarlo como prefieras. `File->Settings->Editor->Colors & Fonts->` y seleccione un tema. También puede descargar nuevos temas desde http://color-themes.com/ Una vez que haya descargado el archivo `.jar.zip` , vaya a `File -> Import Settings...` y elegir el archivo descargado.

Compilando apps

Crea un nuevo proyecto o abre un proyecto existente en Android Studio y presiona el botón verde Play  en la barra de herramientas superior para ejecutarlo. Si está en gris, debe esperar un segundo para permitir que Android Studio indexe correctamente algunos archivos, cuyo progreso se puede ver en la barra de estado inferior.

Si desea crear un proyecto desde el shell, asegúrese de tener un archivo `local.properties` , que Android Studio crea automáticamente. Si necesita crear el proyecto sin Android Studio, necesita una línea que comience con `sdk.dir=` seguida de la ruta a su instalación de SDK.

Abra un shell y vaya al directorio del proyecto. Ingrese `./gradlew aR` y presione enter. `aR` es un acceso directo para `assembleRelease` , que descargará todas las dependencias por ti y creará la aplicación. El archivo final de APK estará en `ProjectName/ModuleName/build/outputs/apk` y se llamará `ModuleName-release.apk` .

Creando un Nuevo Proyecto

Configurar Android Studio

Comience por configurar Android Studio y luego ábralo. ¡Ahora, estás listo para hacer tu primera aplicación de Android!

> **Nota:** esta guía se basa en Android Studio 2.2, pero el proceso en otras versiones es principalmente el mismo.

Configure su proyecto

Configuracion basica

Puedes comenzar un nuevo proyecto de dos maneras:

- Haga clic en `Start a New Android Studio Project` de `Start a New Android Studio Project` en la pantalla de bienvenida.
- Vaya a `File` → `New Project` si ya tiene un proyecto abierto.

A continuación, debe describir su solicitud completando algunos campos:

1. **Nombre de la aplicación** : este nombre se mostrará al usuario.

   Ejemplo: `Hello World` . Siempre puedes cambiarlo más tarde en el archivo `AndroidManifest.xml` .

2. **Dominio de la empresa** : este es el calificador para el nombre del paquete de su proyecto.

   Ejemplo: `stackoverflow.com` .

3. **Nombre del paquete** (también conocido como `applicationId`): este es el nombre *completo del* paquete del proyecto.

   Debe seguir la *Notación de nombre de dominio inversa* (también conocido como *DNS inverso*): *Dominio de nivel superior* **.** *Dominio de la empresa* **.** [*Segmento de la empresa* **.**] *Nombre de la aplicación* .

   Ejemplo: `com.stackoverflow.android.helloworld` o `com.stackoverflow.helloworld` . Siempre puede cambiar su ID de *aplicación* anulando en su archivo de gradle .

   No use el prefijo predeterminado "com.example" a menos que no tenga la intención de enviar su solicitud a Google Play Store. El nombre del paquete será su **aplicación** únicaId en Google Play.

4. **Ubicación del proyecto** : este es el directorio donde se almacenará su proyecto.

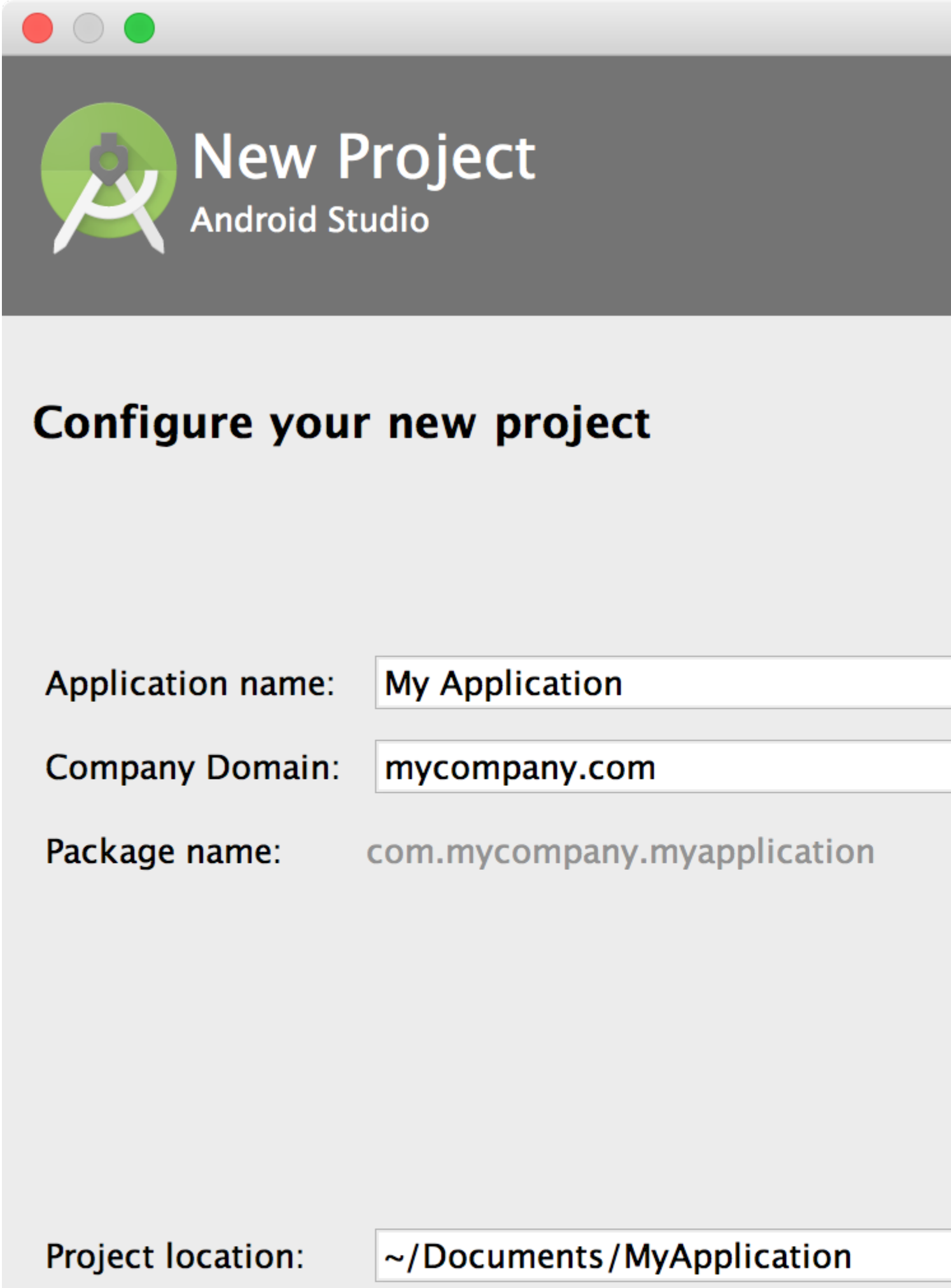
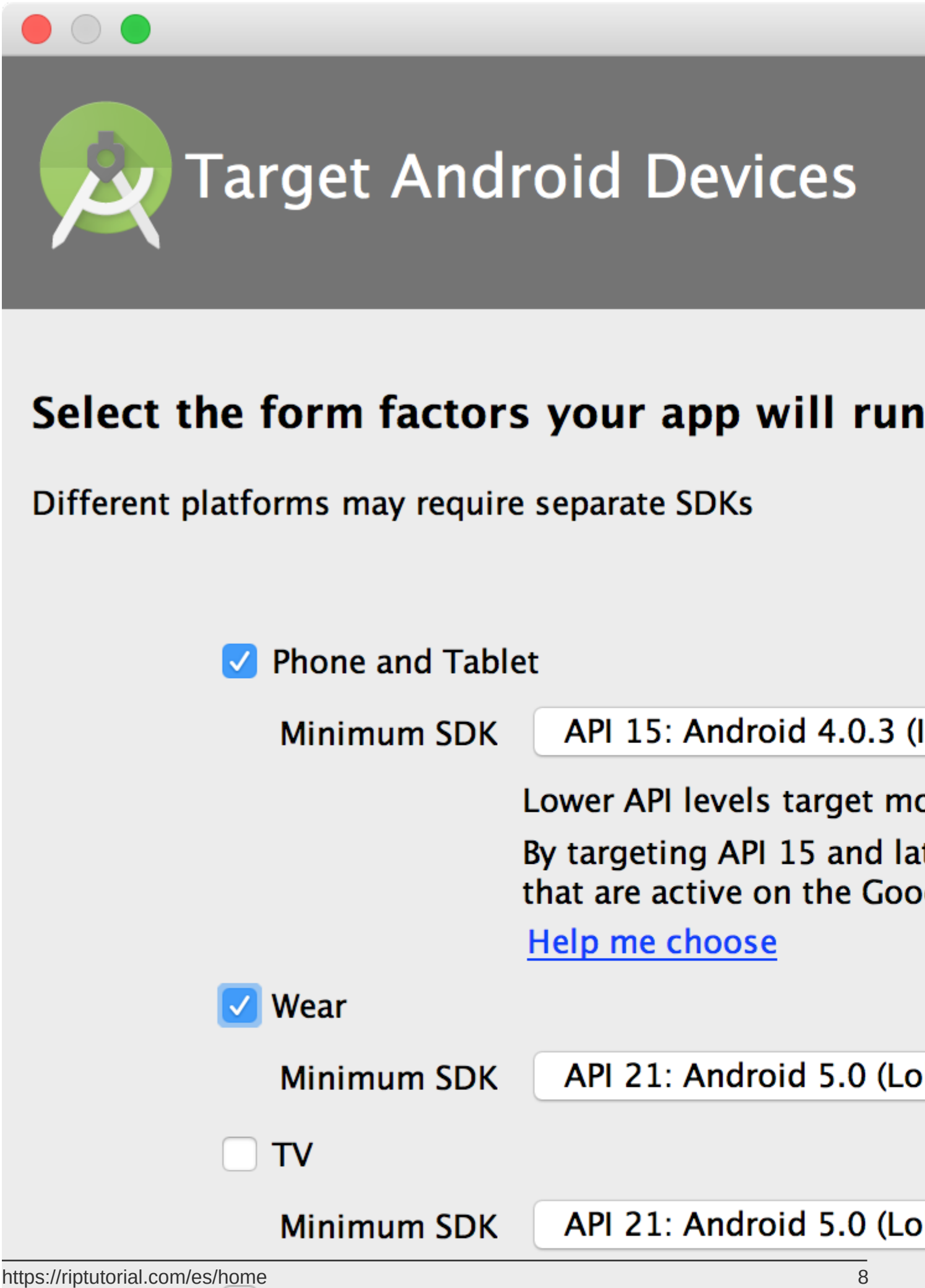

New Project
Android Studio
Configure your new project
Application name:
My Application
Company Domain:
mycompany.com
Package name:
com.mycompany.myapplication
Project location:
~/Documents/MyApplication

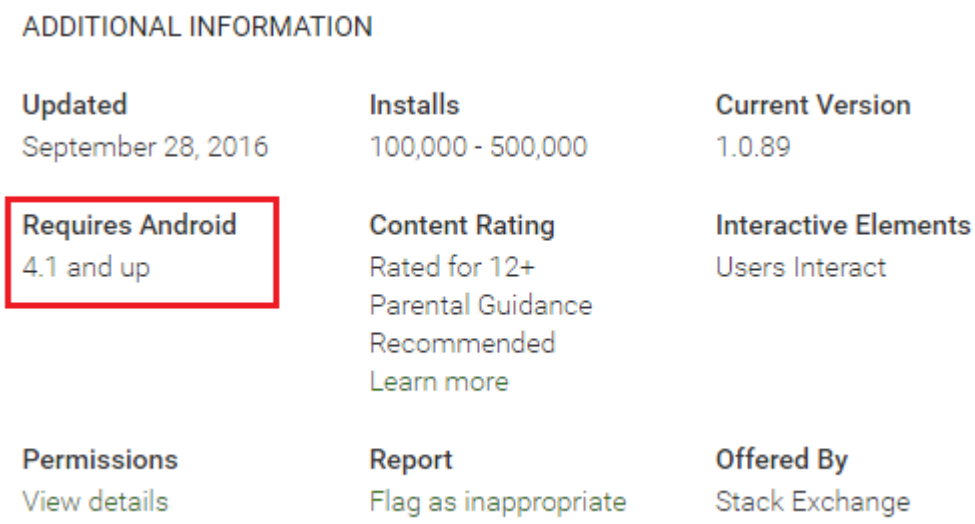
Gráfico de las distribuciones actuales de la versión de Android, que se muestra al hacer clic en Ayudarme a elegir.

La ventana de Distribución de la plataforma de Android muestra la distribución de dispositivos móviles que ejecutan cada versión de Android, como se muestra en la Figura 2. Haga clic en un nivel de API para ver una lista de características introducidas en la versión correspondiente de Android. Esto le ayuda a elegir el nivel de API mínimo que tiene todas las funciones que sus aplicaciones necesitan, para que pueda alcanzar la mayor cantidad de dispositivos posible. Luego haga **clic** en **Aceptar** .

Ahora, elija qué plataformas y versión de Android SDK será compatible con la aplicación.


Target Android Devices
Select the form factors your app will run
Different platforms may require separate SDKs
Phone and Tablet
Minimum SDK
API 15: Android 4.0.3 (I
Lower API levels target mo
By targeting API 15 and lat
that are active on the Goo
Help me choose
Wear
Minimum SDK
API 21: Android 5.0 (Lo
TV
Minimum SDK
API 21: Android 5.0 (Lo
Android Auto

`Google Play Store` para determinar en qué dispositivos se puede instalar una aplicación. Por ejemplo, la aplicación Stack Exchange es compatible con Android 4.1+.



Android Studio le dirá (aproximadamente) qué porcentaje de dispositivos será compatible dado el SDK mínimo especificado.

> Los niveles de API más bajos se dirigen a más dispositivos, pero tienen menos funciones disponibles.

Al decidir sobre el **SDK mínimo** , debe considerar las estadísticas de Dashboards , que le brindarán información sobre la versión de los dispositivos que visitaron la tienda de Google Play a nivel mundial durante la última semana.



Desde: Dashboards en el sitio web del desarrollador de Android.

Añadir una actividad

Ahora vamos a seleccionar una actividad por defecto para nuestra aplicación. En Android, una `Activity` es una pantalla única que se presentará al usuario. Una aplicación puede albergar múltiples actividades y navegar entre ellas. Para este ejemplo, elija `Empty Activity` y haga clic en siguiente.

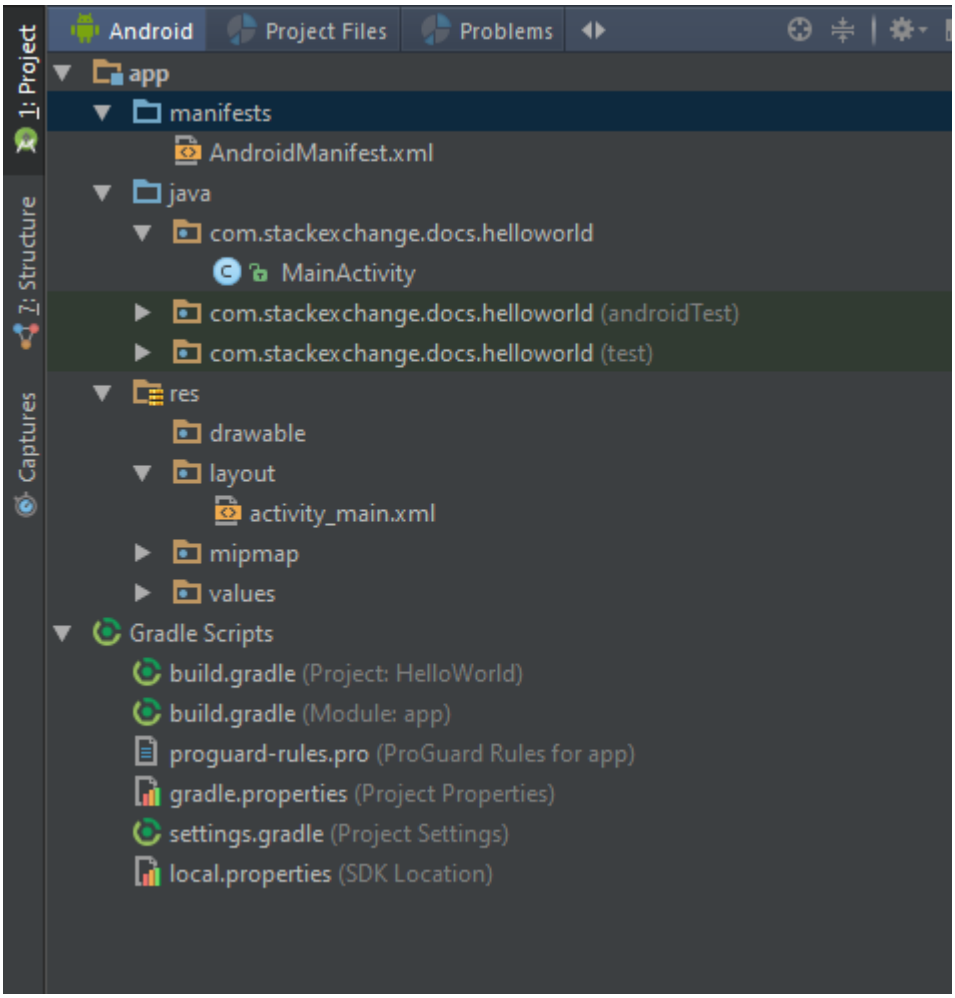
Aquí, si lo desea, puede cambiar el nombre de la actividad y el diseño. Una buena práctica es mantener la `Activity` como un sufijo para el nombre de la actividad, y la `activity_` como un prefijo para el nombre del diseño. Si dejamos esto como predeterminado, Android Studio generará una actividad para nosotros llamada `MainActivity` y un archivo de diseño llamado `activity_main` . Ahora haga clic en `Finish` .

Android Studio creará y configurará nuestro proyecto, lo que puede llevar algún tiempo dependiendo del sistema.

Inspeccionando el proyecto

Para comprender cómo funciona Android, echemos un vistazo a algunos de los archivos que se crearon para nosotros.

En el panel izquierdo de Android Studio, podemos ver la estructura de nuestra aplicación de Android .



Primero, abramos `AndroidManifest.xml` haciendo doble clic en él. El archivo de manifiesto de Android describe parte de la información básica sobre una aplicación de Android. Contiene la declaración de nuestras actividades, así como algunos componentes más avanzados.

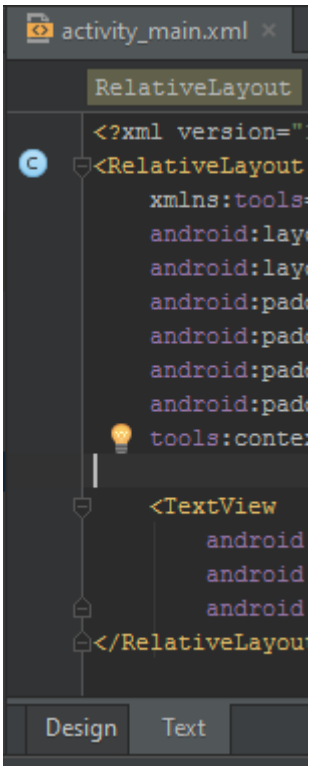
Si una aplicación necesita acceso a una característica protegida por un permiso, debe declarar

que requiere ese permiso con un elemento `<uses-permission>` en el manifiesto. Luego, cuando la aplicación se instala en el dispositivo, el instalador determina si otorga o no el permiso solicitado mediante la verificación de las autoridades que firmaron los certificados de la aplicación y, en algunos casos, le pregunta al usuario. Una aplicación también puede proteger sus propios componentes (actividades, servicios, receptores de difusión y proveedores de contenido) con permisos. Puede emplear cualquiera de los permisos definidos por Android (listados en android.Manifest.permission) o declarados por otras aplicaciones. O puede definir su propia cuenta.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.stackoverflow.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

A continuación, abramos `activity_main.xml` que se encuentra en `app/src/main/res/layout/` . Este archivo contiene declaraciones para los componentes visuales de nuestra MainActivity. Verás diseñador visual. Esto le permite arrastrar y soltar elementos en el diseño seleccionado.

También puede cambiar al diseñador de diseño xml haciendo clic en "Texto" en la parte inferior de Android Studio, como se ve aquí:




```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.stackexchange.docs.helloworld.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</RelativeLayout>
```

Verá un widget llamado `TextView` dentro de este diseño, con la propiedad de `android:text` establecida en "¡Hola mundo!". Este es un bloque de texto que se mostrará al usuario cuando ejecute la aplicación.

Puedes leer más sobre Diseños y atributos .

A continuación, echemos un vistazo a `MainActivity` . Este es el código Java que se ha generado para `MainActivity` .

```
public class MainActivity extends AppCompatActivity {

    // The onCreate method is called when an Activity starts
    // This is where we will set up our layout
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        // setContentView sets the Activity's layout to a specified XML layout
        // In our case we are using the activity_main layout
        setContentView(R.layout.activity_main);
    }
}
```

Como se define en nuestro manifiesto de Android, `MainActivity` se iniciará de forma predeterminada cuando un usuario inicie la aplicación `HelloWorld` .

Por último, abra el archivo llamado `build.gradle` ubicado en `app/` .
Android Studio utiliza el sistema de compilación **Gradle** para compilar y construir bibliotecas y aplicaciones de Android.

```
apply plugin: 'com.android.application'

android {
signingConfigs {
         applicationName {
             keyAlias 'applicationName'
             keyPassword 'password'
             storeFile file('../key/applicationName.jks')
             storePassword 'anotherPassword'
         }
     }
    compileSdkVersion 26
    buildToolsVersion "26.0.0"

    defaultConfig {
        applicationId "com.stackexchange.docs.helloworld"
        minSdkVersion 16
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        signingConfig signingConfigs.applicationName
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:26.0.0'
}
```

Este archivo contiene información sobre la compilación y la versión de su aplicación, y también puede usarla para agregar dependencias a bibliotecas externas. Por ahora, no hagamos ningún cambio.

Es recomendable seleccionar siempre la última versión disponible para las dependencias:

- buildToolsVersion : 26.0.0

- com.android.support:appcompat-v7 : 26.0.0 (julio de 2017)
- base de fuego : 11.0.4 (agosto de 2017)

compileSdkVersion

`compileSdkVersion` es su forma de decirle a *Gradle con* qué versión del SDK de Android debe compilar su aplicación. El uso del nuevo SDK de Android es un requisito para usar cualquiera de las nuevas API agregadas en ese nivel.

Se debe enfatizar que cambiar su `compileSdkVersion` no cambia el comportamiento del tiempo de ejecución. Si bien pueden aparecer nuevos avisos / errores de compilación al cambiar su `compileSdkVersion` , su `compileSdkVersion` no está incluido en su APK: se utiliza únicamente en tiempo de compilación.

Por lo tanto, se recomienda encarecidamente que siempre compile con el último SDK. Obtendrá todos los beneficios de las nuevas comprobaciones de compilación en el código existente, evitará las API recientemente obsoletas y estará listo para usar nuevas API.

minSdkVersion

Si `compileSdkVersion` establece las API más nuevas disponibles para usted, `minSdkVersion` es el *límite inferior* para su aplicación. `minSdkVersion` es una de las señales que utiliza Google Play Store para determinar en qué dispositivos de un usuario se puede instalar una aplicación.

También juega un papel importante durante el desarrollo: de manera predeterminada, la pelusa se ejecuta contra su proyecto, advirtiéndole cuando use cualquier API por encima de `minSdkVersion` , lo que lo ayuda a evitar el problema de tiempo de ejecución al intentar llamar a una API que no existe. Verificar la versión del sistema en tiempo de ejecución es una técnica común cuando se usan API solo en las versiones más nuevas de la plataforma.

targetSdkVersion

`targetSdkVersion` es la principal forma en que Android proporciona compatibilidad hacia adelante al no aplicar cambios de comportamiento a menos que se actualice `targetSdkVersion` . Esto le permite utilizar nuevas API antes de trabajar a través de los cambios de comportamiento. La actualización para apuntar al último SDK debe ser una alta prioridad para cada aplicación. Eso no significa que tenga que usar cada nueva función introducida ni debería actualizar ciegamente su `targetSdkVersion` sin probar.

targetSDKVersion es la versión de Android que es el límite superior para las herramientas disponibles. Si targetSDKVersion es menor que 23, la aplicación no necesita solicitar permisos en tiempo de ejecución para una instancia, incluso si la aplicación se está ejecutando en API 23+. TargetSDKVersion **no** impide que las versiones de Android sobre la versión seleccionada de Android ejecuten la aplicación.

Puedes encontrar más información sobre el plugin Gradle:

- Un ejemplo basico
- Introducción al plugin Gradle para Android y la envoltura

- Introducción a la configuración de los métodos build.gradle y DSL.

Ejecutando la aplicación

Ahora, vamos a ejecutar nuestra aplicación HelloWorld. Puede ejecutar un dispositivo virtual de Android (que puede configurar utilizando AVD Manager en Android Studio, como se describe en el siguiente ejemplo) o conectar su propio dispositivo Android a través de un cable USB.

Configuración de un dispositivo Android

Para ejecutar una aplicación desde Android Studio en su dispositivo Android, debe habilitar la `USB Debugging` en las `Developer Options` en la configuración de su dispositivo.

`Settings > Developer options > USB debugging`

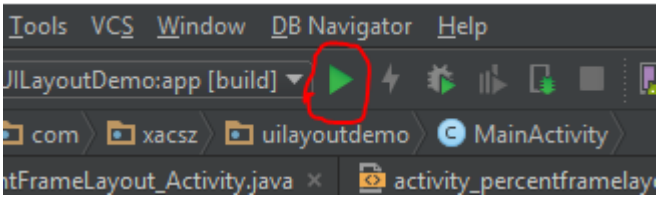
Si las `Developer Options` no están visibles en la configuración, navegue hasta `About Phone` y toque el `Build Number` siete veces. Esto permitirá que las `Developer Options` aparezcan en tu configuración.

`Settings > About phone > Build number`

También es posible que deba cambiar la configuración de `build.gradle` para construir en una versión que tenga su dispositivo.

Ejecutando desde Android Studio

Haga clic en el botón verde `Run` de la barra de herramientas en la parte superior de Android Studio. En la ventana que aparece, seleccione el dispositivo en el que desea ejecutar la aplicación (inicie un dispositivo virtual de Android si es necesario, o consulte Configuración de un AVD (dispositivo virtual de Android) si necesita configurar uno) y haga `OK` en `OK` .



En dispositivos con Android 4.4 (KitKat) y posiblemente superior, se mostrará una ventana emergente para autorizar la depuración USB. Haga `OK` en `OK` para aceptar.

La aplicación ahora se instalará y ejecutará en su dispositivo o emulador de Android.

Ubicación del archivo APK

Cuando prepara su aplicación para el lanzamiento, configura, crea y prueba una versión de lanzamiento de su aplicación. Las tareas de configuración son sencillas, e involucran tareas

básicas de limpieza de código y modificación de código que ayudan a optimizar su aplicación. El proceso de compilación es similar al proceso de compilación de depuración y se puede hacer usando las herramientas JDK y Android SDK. Las tareas de prueba sirven como una verificación final, asegurando que su aplicación se desempeña como se espera en condiciones reales. Cuando haya terminado de preparar su aplicación para el lanzamiento, tiene un archivo APK firmado, que puede distribuir directamente a los usuarios o distribuir a través de un mercado de aplicaciones como Google Play.

Android Studio

Como en los ejemplos anteriores se usa Gradle, la ubicación del archivo APK generado es: `<Your Project Location>/app/build/outputs/apk/app-debug.apk`

IntelliJ

Si usted es un usuario de IntelliJ antes de cambiar a Studio y está importando su proyecto de IntelliJ directamente, entonces nada cambió. La ubicación de la salida será la misma en:

```
out/production/...
```

Nota: esto será desaprobado a veces alrededor de 1.0

Eclipse

Si está importando directamente el proyecto Eclipse de Android, ¡no lo haga! Tan pronto como tenga dependencias en su proyecto (archivos jar o proyectos de biblioteca), esto no funcionará y su proyecto no se configurará correctamente. Si no tiene dependencias, entonces el apk se encontraría en la misma ubicación que lo encontraría en Eclipse:

```
bin/...
```

Programación de Android sin un IDE.

Este es un ejemplo minimalista de Hello World que usa solo las herramientas más básicas de Android.

Requisitos y suposiciones

- Oracle JDK 1.7 o posterior
- Herramientas de Android SDK (solo las herramientas de línea de comandos)

Este ejemplo asume Linux. Puede que tenga que ajustar la sintaxis para su propia plataforma.

Configurando el SDK de Android

Después de desempacar la versión SDK:

1. Instalar paquetes adicionales utilizando el administrador de SDK. No use la `android update sdk --no-ui` como se indica en el `android update sdk --no-ui` Readme.txt; Descarga unos 30 GB de archivos innecesarios. En su lugar, use el administrador de SDK interactivo para `android sdk` para obtener los paquetes mínimos recomendados.

2. Agregue los siguientes directorios JDK y SDK a su PATH de ejecución. Esto es opcional, pero las instrucciones a continuación lo asumen.

    - JDK / bin
    - SDK / plataforma-herramientas
    - SDK / herramientas
    - SDK / build-tools / LATEST *(como se instaló en el paso 1)*

3. Crea un dispositivo virtual Android. Utilice el AVD Manager interactivo (`android avd` AVD). Puede que tenga que juguetear un poco y buscar consejo; Las instrucciones en el sitio no siempre son útiles.

    (También puedes usar tu propio dispositivo)

4. Ejecuta el dispositivo:

    ```
    emulator -avd DEVICE
    ```

5. Si la pantalla del dispositivo parece estar bloqueada, desliza para desbloquearla.

    Deja que se ejecute mientras codificas la aplicación.

Codificando la aplicación

6. Cambiar a un directorio de trabajo vacío.

7. Haz el archivo fuente:

    ```
    mkdir --parents src/dom/domain
    touch src/dom/domain/SayingHello.java
    ```

    Contenido:

    ```
    package dom.domain;
    import android.widget.TextView;

    public final class SayingHello extends android.app.Activity
    {
        protected @Override void onCreate( final android.os.Bundle activityState )
        {
            super.onCreate( activityState );
            final TextView textV = new TextView( SayingHello.this );
            textV.setText( "Hello world" );
            setContentView( textV );
        }
    ```

```
}
```

8. Añadir un manifiesto:

```
touch AndroidManifest.xml
```

Contenido:

```
<?xml version='1.0'?>
<manifest xmlns:a='http://schemas.android.com/apk/res/android'
 package='dom.domain' a:versionCode='0' a:versionName='0'>
    <application a:label='Saying hello'>
        <activity a:name='dom.domain.SayingHello'>
             <intent-filter>
                <category a:name='android.intent.category.LAUNCHER'/>
                <action a:name='android.intent.action.MAIN'/>
                </intent-filter>
            </activity>
        </application>
    </manifest>
```

9. Haga un subdirectorio para los recursos declarados:

```
mkdir res
```

Déjalo vacío por ahora.

Construyendo el código

10. Generar la fuente para las declaraciones de recursos. Sustituya aquí la ruta correcta a su **SDK** y la **API** instalada contra la que construir (por ejemplo, "android-23"):

```
aapt package -f \
  -I SDK/platforms/android-API/android.jar \
  -J src -m \
  -M AndroidManifest.xml -S res -v
```

Las declaraciones de recursos (que se describen más adelante) son en realidad opcionales. Mientras tanto, la llamada anterior no hace nada si res / todavía está vacío.

11. Compile el código fuente en el bytecode de Java (.java → .class):

```
javac \
  -bootclasspath SDK/platforms/android-API/android.jar \
  -classpath src -source 1.7 -target 1.7 \
  src/dom/domain/*.java
```

12. Traduzca el código de bytes de Java a Android (.class → .dex):

Primero usando Jill (.class → .jayce):

```
java -jar SDK/build-tools/LATEST/jill.jar \
  --output classes.jayce src
```

Entonces Jack (.jayce → .dex):

```
java -jar SDK/build-tools/LATEST/jack.jar \
  --import classes.jayce --output-dex .
```

El código de bytes de Android solía llamarse "código ejecutable de Dalvik", y por lo tanto "dex".

Podría reemplazar los pasos 11 y 12 con una sola llamada a Jack si lo desea; puede compilar directamente desde la fuente Java (.java → .dex). Pero hay ventajas de compilar con `javac` . Es una herramienta más conocida, mejor documentada y más ampliamente aplicable.

13. Empaquetar los archivos de recursos, incluido el manifiesto:

```
aapt package -f \
  -F app.apkPart \
  -I SDK/platforms/android-API/android.jar \
  -M AndroidManifest.xml -S res -v
```

Eso resulta en un archivo APK parcial (paquete de aplicación de Android).

14. Haz la APK completa usando la herramienta `ApkBuilder` :

```
java -classpath SDK/tools/lib/sdklib.jar \
  com.android.sdklib.build.ApkBuilderMain \
  app.apkUnalign \
  -d -f classes.dex -v -z app.apkPart
```

Advierte, "ESTA HERRAMIENTA ESTÁ DEPRECTA. Vea --help para obtener más información". Si `--help` falla con una `ArrayIndexOutOfBoundsException` , en su lugar no pase ningún argumento:

```
java -classpath SDK/tools/lib/sdklib.jar \
  com.android.sdklib.build.ApkBuilderMain
```

Explica que la CLI (`ApkBuilderMain`) está en desuso a favor de llamar directamente a la API de Java (`ApkBuilder`). (Si sabe cómo hacerlo desde la línea de comandos, actualice este ejemplo).

15. Optimizar la alineación de datos de la APK (práctica recomendada):

```
zipalign -f -v 4 app.apkUnalign app.apk
```

Instalación y ejecución

16. Instala la aplicación en el dispositivo Android:

```
adb install -r app.apk
```

17. Inicia la aplicación:

```
adb shell am start -n dom.domain/.SayingHello
```

    Debería correr y saludar.

Eso es todo. Eso es lo que se necesita para saludar con las herramientas básicas de Android.

Declarar un recurso

Esta sección es opcional. No se requieren declaraciones de recursos para una aplicación simple "hello world". Si tampoco son necesarios para su aplicación, entonces podría simplificar un poco la compilación omitiendo el paso 10 y eliminando la referencia al directorio res / del paso 13.

De lo contrario, aquí hay un breve ejemplo de cómo declarar un recurso y cómo hacer referencia a él.

18. Agrega un archivo de recursos:

```
mkdir res/values
touch res/values/values.xml
```

    Contenido:

```
<?xml version='1.0'?>
<resources>
    <string name='appLabel'>Saying hello</string>
</resources>
```

19. Referencia el recurso desde el manifiesto XML. Este es un estilo declarativo de referencia:

```
<!-- <application a:label='Saying hello'> -->
     <application a:label='@string/appLabel'>
```

20. Referencia el mismo recurso desde la fuente de Java. Esta es una referencia imperativa:

```
// v.setText( "Hello world" );
   v.setText( "This app is called "
     + getResources().getString( R.string.appLabel ));
```

21. Pruebe las modificaciones anteriores reconstruyendo, reinstalando y volviendo a ejecutar la aplicación (pasos 10-17).

    Debería reiniciarse y decir: "Esta aplicación se llama Decir hola".

Desinstalando la aplicación

```
adb uninstall dom.domain
```

Ver también

- Pregunta original - La pregunta original que motivó este ejemplo.
- ejemplo de trabajo : un script de compilación de trabajo que utiliza los comandos anteriores

Fundamentos de la aplicación

Las aplicaciones de Android están escritas en Java. Las herramientas de Android SDK compilan los archivos de código, datos y recursos en un APK (paquete de Android). En general, un archivo APK contiene todo el contenido de la aplicación.

Cada aplicación se ejecuta en su propia máquina virtual (VM) para que la aplicación pueda ejecutarse aislada de otras aplicaciones. El sistema Android funciona con el principio de privilegio mínimo. Cada aplicación solo tiene acceso a los componentes que requiere para hacer su trabajo, y no más. Sin embargo, hay formas para que una aplicación comparta datos con otras aplicaciones, como compartir la identificación de usuario de Linux entre aplicaciones, o las aplicaciones pueden solicitar permiso para acceder a datos de dispositivos como tarjetas SD, contactos, etc.

Componentes de la aplicación

Los componentes de la aplicación son los componentes básicos de una aplicación de Android. Cada componente desempeña un papel específico en una aplicación de Android que tiene un propósito distinto y tiene ciclos de vida distintos (el flujo de cómo y cuándo se crea y destruye el componente). Aquí están los cuatro tipos de componentes de la aplicación:

1. **Actividades:** una actividad representa una única pantalla con una interfaz de usuario (UI). Una aplicación de Android puede tener más de una actividad. (por ejemplo, una aplicación de correo electrónico puede tener una actividad para enumerar todos los correos electrónicos, otra para mostrar el contenido de cada correo electrónico y otra para redactar un nuevo correo electrónico). Todas las actividades en una Aplicación trabajan juntas para crear una experiencia de usuario (UX).
2. **Servicios:** un servicio se ejecuta en segundo plano para realizar operaciones de larga ejecución o para realizar un trabajo en procesos remotos. Un servicio no proporciona ninguna IU, se ejecuta solo en segundo plano con la entrada del usuario. (Por ejemplo, un

servicio puede reproducir música en segundo plano mientras el usuario está en una aplicación diferente, o puede descargar datos de Internet sin bloquear la interacción del usuario con el dispositivo Android).

3. **Proveedores de contenido:** un proveedor de contenido administra los datos compartidos de la aplicación. Hay cuatro formas de almacenar datos en una aplicación: puede escribirse en un archivo y almacenarse en el sistema de archivos, insertarse o actualizarse en una base de datos SQLite, publicarse en la web o guardarse en cualquier otra ubicación de almacenamiento persistente a la que la aplicación pueda acceder. . A través de los proveedores de contenido, otras aplicaciones pueden consultar o incluso modificar los datos. (por ejemplo, el sistema Android proporciona un proveedor de contenido que administra la información de contacto del usuario para que cualquier aplicación que tenga permiso pueda consultar a los contactos). Los proveedores de contenido también se pueden usar para guardar los datos privados de la aplicación para una mejor integridad de los datos.
4. **Receptores de transmisión:** un receptor de transmisión responde a las transmisiones de anuncios de todo el sistema (p. Ej., Una transmisión que anuncia que la pantalla se ha apagado, que la batería está baja, etc.) o desde Aplicaciones (p. Ej., Para que otras aplicaciones sepan que se han detectado algunos datos). descargado al dispositivo y está disponible para su uso). Los receptores de transmisión no tienen interfaces de usuario, pero pueden mostrar notificaciones en la barra de estado para alertar al usuario. Por lo general, los receptores de difusión se utilizan como puerta de entrada a otros componentes de la aplicación, que consisten principalmente en actividades y servicios.

Un aspecto único del sistema Android es que cualquier aplicación puede iniciar el componente de otra aplicación (por ejemplo, si desea hacer una llamada, enviar SMS, abrir una página web o ver una foto, hay una aplicación que ya lo hace y su aplicación puede hacer uso de él, en lugar de desarrollar una nueva actividad para la misma tarea).

Cuando el sistema inicia un componente, inicia el proceso para esa aplicación (si aún no se está ejecutando, es decir, solo un proceso de primer plano por aplicación puede ejecutarse en un momento dado en un sistema Android) y crea una instancia de las clases necesarias para ese componente. Por lo tanto, el componente se ejecuta en el proceso de esa aplicación a la que pertenece. Por lo tanto, a diferencia de las aplicaciones en otros sistemas, las aplicaciones de Android no tienen un solo punto de entrada (no hay un método `main()`).

Debido a que el sistema ejecuta cada aplicación en un proceso separado, una aplicación no puede activar directamente los componentes de otra aplicación, como puede hacerlo el sistema Android. Por lo tanto, para iniciar el componente de otra aplicación, una aplicación debe enviar un mensaje al sistema que especifique la intención de iniciar ese componente, luego el sistema iniciará ese componente.

Contexto

Las instancias de la clase `android.content.Context` proporcionan la conexión al sistema Android que ejecuta la aplicación. Se requiere Instance of Context para obtener acceso a los recursos del proyecto y la información global sobre el entorno de la aplicación.

Pongamos un ejemplo fácil de digerir: considera que estás en un hotel y quieres comer algo.

Llama al servicio de habitaciones y les pide que le traigan cosas o que limpien cosas para usted. Ahora piense en este hotel como una aplicación de Android, usted mismo como una actividad, y la persona de servicio de habitación es su contexto, que le brinda acceso a los recursos del hotel, como servicio de habitaciones, alimentos, etc.

Sin embargo, otro ejemplo: usted está en un restaurante sentado en una mesa, cada mesa tiene un asistente, cuando quiera pedir alimentos, le pide al asistente que lo haga. Luego, el asistente hace su pedido y sus alimentos se sirven en su mesa. Nuevamente, en este ejemplo, el restaurante es una aplicación de Android, las mesas o los clientes son componentes de la aplicación, los alimentos son sus recursos de la aplicación y el asistente es su contexto, lo que le brinda una manera de acceder a los recursos como alimentos.

La activación de cualquiera de los componentes anteriores requiere la instancia del contexto. No solo lo anterior, sino también casi todos los recursos del sistema: creación de la IU mediante vistas (que se analiza más adelante), creación de instancias de servicios del sistema, inicio de nuevas actividades o servicios, todo requiere un contexto.

Una descripción más detallada se escribe aquí .

Configuración de un AVD (dispositivo virtual de Android)

TL; DR Básicamente, nos permite simular dispositivos reales y probar nuestras aplicaciones sin un dispositivo real.

Según la documentación del desarrollador de Android ,

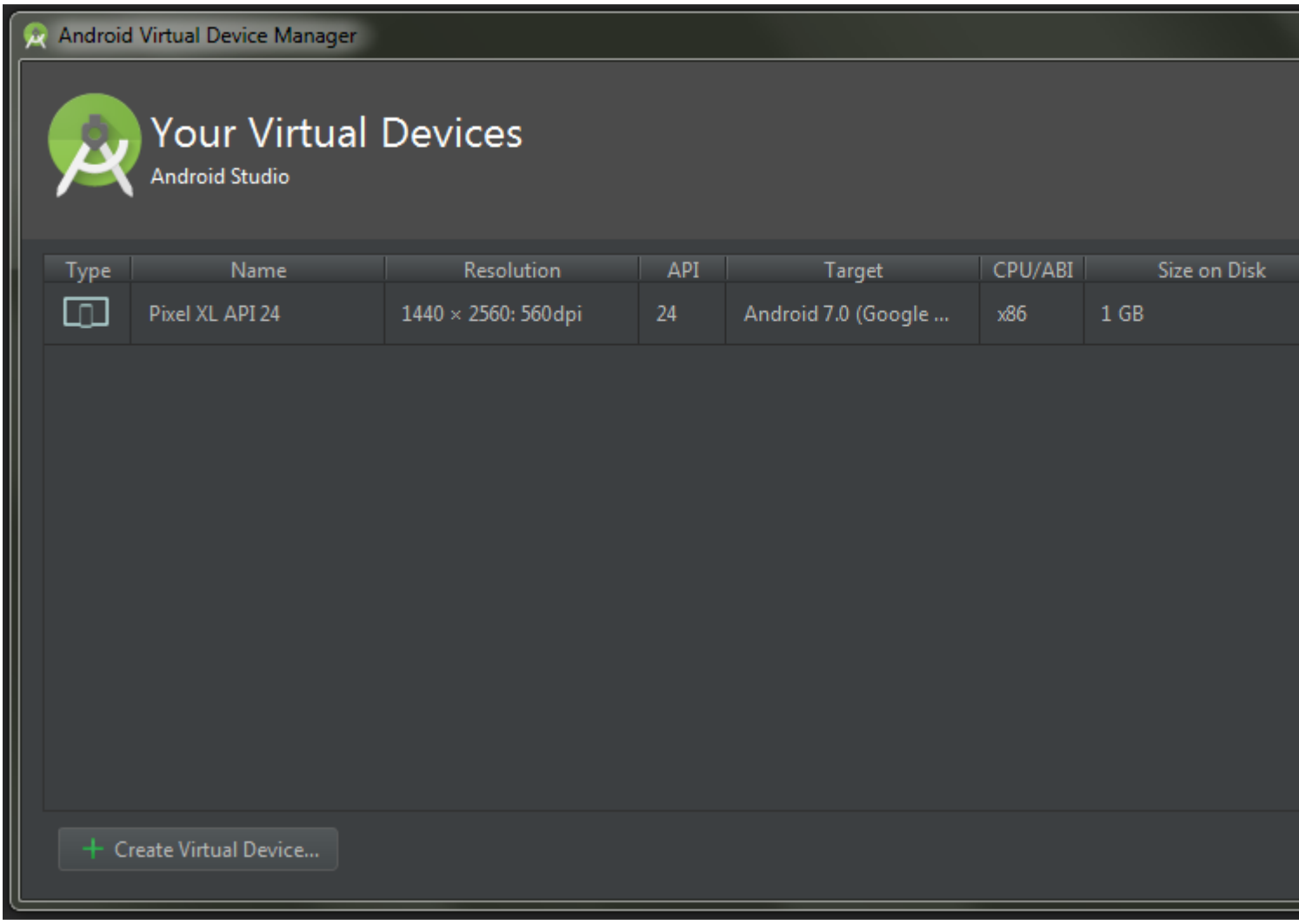
> una *definición de* ***dispositivo virtual de Android (AVD) le*** permite definir las características de un teléfono, tableta, Android Wear o dispositivo de TV Android que desee simular en el emulador de Android. AVD Manager lo ayuda a crear y administrar AVD fácilmente.

Para configurar un AVD, siga estos pasos:

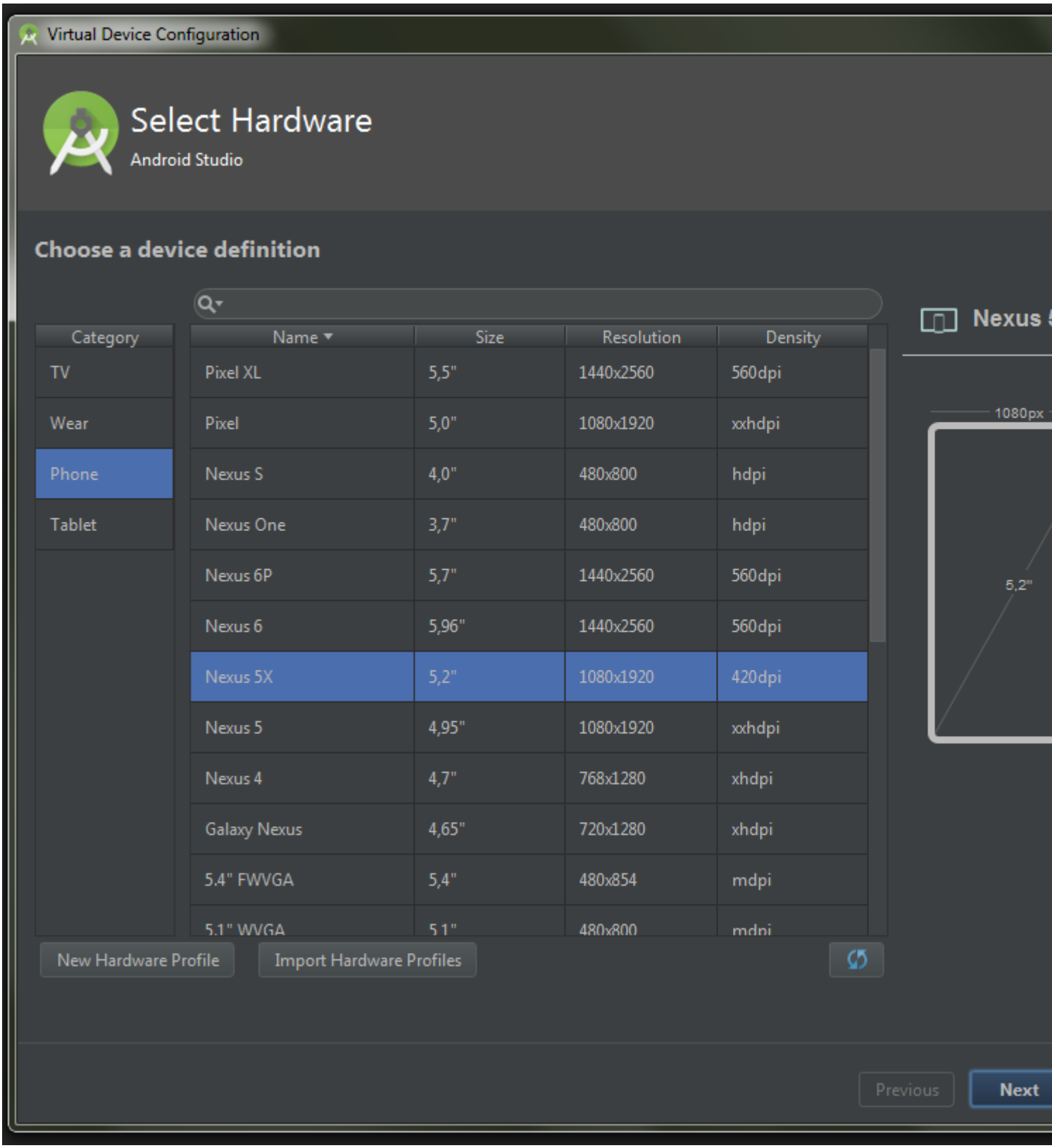
1. Haga clic en este botón para abrir el Administrador de AVD:



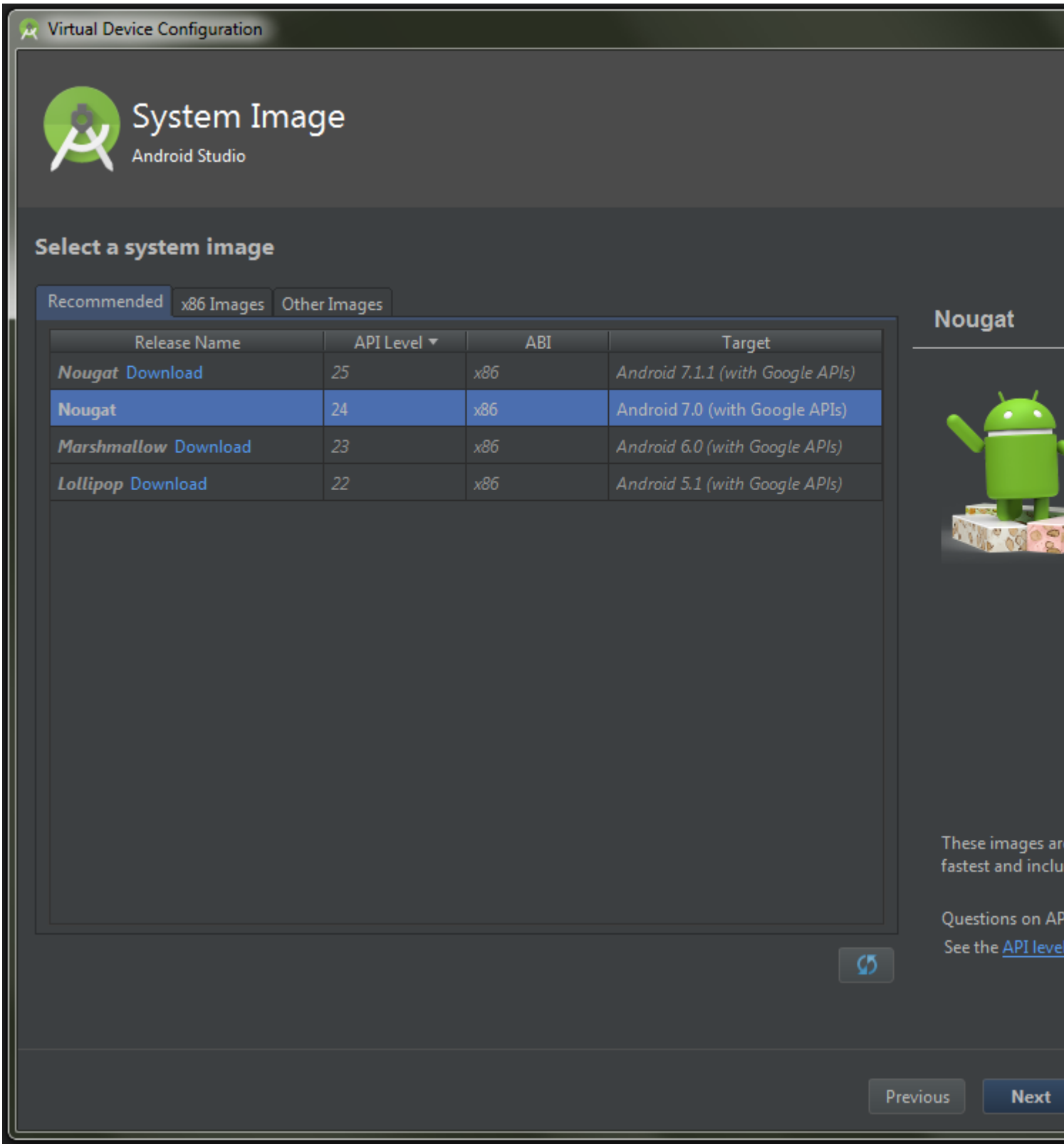
2. Deberías ver un diálogo como este:



3. Ahora haga clic en el botón `+ Create Virtual Device...` Esto abrirá el diálogo de configuración del dispositivo virtual:



4. Seleccione el dispositivo que desee y haga clic en `Next` :

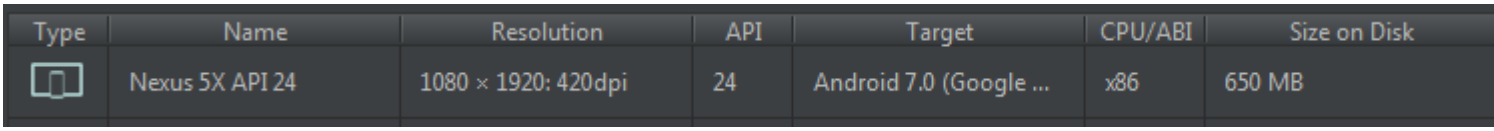


5. Aquí debes elegir una versión de Android para tu emulador. Es posible que también necesite descargarlo primero haciendo clic en `Download` . Después de haber elegido una versión, haga clic en `Next` .



6. Aquí, ingrese un nombre para su emulador, orientación inicial y si desea mostrar un marco a su alrededor. Después de haber elegido todos estos, haga clic en `Finish` .

7. Ahora tienes un nuevo AVD listo para lanzar tus aplicaciones en él.



Lea Empezando con Android en línea: https://riptutorial.com/es/android/topic/85/empezando-con-android

Capítulo 2: ¿Qué es ProGuard? ¿Qué es el uso en Android?

Introducción

Proguard es un reductor, optimizador, ofuscador y preverificador de archivos de clase Java. Detecta y elimina clases, campos, métodos y atributos no utilizados. Optimiza el bytecode y elimina las instrucciones no utilizadas. Renombra las clases, campos y métodos restantes utilizando nombres cortos sin significado.

Examples

Reduce tu código y recursos con proguard

Para hacer que su archivo APK sea lo más pequeño posible, debe habilitar la reducción para eliminar el código y los recursos no utilizados en su versión de lanzamiento. Esta página describe cómo hacerlo y cómo especificar qué código y recursos mantener o descartar durante la compilación.

La reducción de código está disponible con ProGuard, que detecta y elimina las clases, campos, métodos y atributos no utilizados de su aplicación empaquetada, incluidos los de las bibliotecas de códigos incluidas (lo que la convierte en una herramienta valiosa para trabajar alrededor del límite de referencia de 64k). ProGuard también optimiza el código de bytes, elimina las instrucciones de código no utilizadas y confunde las clases, campos y métodos restantes con nombres cortos. El código confuso dificulta la ingeniería inversa de su APK, lo que es especialmente valioso cuando su aplicación utiliza características sensibles a la seguridad, como la verificación de licencias.

La reducción de recursos está disponible con el complemento de Android para Gradle, que elimina los recursos no utilizados de su aplicación empaquetada, incluidos los recursos no utilizados en las bibliotecas de códigos. Funciona junto con la reducción de código, de modo que una vez que se ha eliminado el código no utilizado, cualquier recurso que ya no se hace referencia también se puede eliminar de forma segura.

Encoge tu código

Para habilitar la reducción de código con `ProGuard` , agregue `minifyEnabled` true al tipo de compilación apropiado en su archivo `build.gradle` .

Tenga en cuenta que la reducción de código ralentiza el tiempo de compilación, por lo que debe evitar usarlo en su compilación de depuración si es posible. Sin embargo, es importante que habilite la reducción de código en su APK final utilizado para las pruebas, ya que podría introducir errores si no personaliza suficientemente qué código mantener.

Por ejemplo, el siguiente fragmento de código de un archivo `build.gradle` permite la reducción de

código para la versión de lanzamiento:

```
android {
    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
                    'proguard-rules.pro'
        }
    }
    ...
}
```

Además de la propiedad `minifyEnabled` , la propiedad `proguardFiles` define las `ProGuard rules` :

El método getDefaultProguardFile ('proguard-android.txt') obtiene la configuración predeterminada de ProGuard de la `tools/proguard/ folder` Android SDK. Consejo: para reducir aún más el código, pruebe el `proguard-android-optimize.txt` que se encuentra en la misma ubicación. Incluye las mismas reglas de ProGuard, pero con otras optimizaciones que realizan análisis en el nivel de bytecode, dentro y en todos los métodos, para reducir aún más el tamaño de su APK y ayudarlo a correr más rápido. El archivo `proguard-rules.pro` es donde puede agregar reglas personalizadas de ProGuard. De forma predeterminada, este archivo se encuentra en la raíz del módulo (junto al archivo build.gradle). Para añadir más reglas ProGuard que son específicas para cada variante de construcción, agregar otra propiedad proguardFiles en el correspondiente `productFlavor` bloque. Por ejemplo, el siguiente archivo de Gradle agrega flavor2-rules.pro al sabor de producto flavour2. Ahora flavor2 usa las tres reglas de ProGuard porque también se aplican las del bloque de publicación.

```
android {
    ...
    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
                   'proguard-rules.pro'
        }
    }
    productFlavors {
        flavor1 {
        }
        flavor2 {
            proguardFile 'flavor2-rules.pro'
        }
    }
}
```

Lea ¿Qué es ProGuard? ¿Qué es el uso en Android? en línea:
https://riptutorial.com/es/android/topic/9205/-que-es-proguard---que-es-el-uso-en-android-

Capítulo 3: Accediendo a bases de datos SQLite usando la clase ContentValues

Examples

Insertar y actualizar filas en una base de datos SQLite

Primero, necesita abrir su base de datos SQLite, que se puede hacer de la siguiente manera:

```
SQLiteDatabase myDataBase;
String mPath = dbhelper.DATABASE_PATH + dbhelper.DATABASE_NAME;
myDataBase = SQLiteDatabase.openDatabase(mPath, null, SQLiteDatabase.OPEN_READWRITE);
```

Después de abrir la base de datos, puede insertar o actualizar filas fácilmente usando la clase `ContentValues` . Los siguientes ejemplos asumen que un primer nombre es dado por `str_edtfname` y un último nombre `str_edtlname` . También debe reemplazar `table_name` por el nombre de la tabla que desea modificar.

Insertando datos

```
ContentValues values = new ContentValues();
values.put("First_Name", str_edtfname);
values.put("Last_Name", str_edtlname);
myDataBase.insert("table_name", null, values);
```

Actualización de datos

```
ContentValues values = new ContentValues();
values.put("First_Name", str_edtfname);
values.put("Last_Name", str_edtlname);
myDataBase.update("table_name", values, "id" + " = ?", new String[] {id});
```

Lea Accediendo a bases de datos SQLite usando la clase ContentValues en línea: https://riptutorial.com/es/android/topic/10154/accediendo-a-bases-de-datos-sqlite-usando-la-clase-contentvalues

Capítulo 4: ACRA

Sintaxis

- android: name = ". ACRAHandler"
- ACRA.init (esto, config);
- clase pública ACRAHandler extiende aplicación {

Parámetros

Parámetro	Descripción
@ReportCrashes	Define la configuración de ACRA, por ejemplo, dónde se debe informar, el contenido personalizado, etc.
formUri	la ruta al archivo que informa del fallo

Observaciones

- ACRA ya no admite formularios de Google, por lo que necesita un servidor: https://github.com/ACRA/acra/wiki/Backends

Examples

ACRAHandler

Ejemplo de clase que extiende la aplicación para manejar el informe:

```
@ReportsCrashes(

        formUri = "https://backend-of-your-choice.com/",//Non-password protected.
        customReportContent = { /* */ReportField.APP_VERSION_NAME,
ReportField.PACKAGE_NAME,ReportField.ANDROID_VERSION,
ReportField.PHONE_MODEL,ReportField.LOGCAT },
        mode = ReportingInteractionMode.TOAST,
        resToastText = R.string.crash

)
public class ACRAHandler extends Application {
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);

        final ACRAConfiguration config = new ConfigurationBuilder(this)

                .build();

        // Initialise ACRA
```

```
        ACRA.init(this, config);


    }

}
```

Ejemplo manifiesto

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    <!-- etc -->

>



    <!-- Internet is required. READ_LOGS are to ensure that the Logcat is transmitted-->
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.READ_LOGS"/>

    <application
        android:allowBackup="true"
        android:name=".ACRAHandler"<!-- Activates ACRA on startup -->
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >



        <!-- Activities -->
    </application>

</manifest>
```

Instalación

Maven

```
<dependency>
    <groupId>ch.acra</groupId>
    <artifactId>acra</artifactId>
    <version>4.9.2</version>
    <type>aar</type>
</dependency>
```

Gradle

```
compile 'ch.acra:acra:4.9.2'
```

Lea ACRA en línea: https://riptutorial.com/es/android/topic/1324/acra

Capítulo 5: Actividad

Introducción

Una Actividad representa una sola pantalla con una **interfaz de** usuario **(UI)** . Una aplicación de Android puede tener más de una actividad, por ejemplo, una aplicación de correo electrónico puede tener una actividad para enumerar todos los correos electrónicos, otra actividad para mostrar el contenido del correo electrónico, y otra actividad para redactar un nuevo correo electrónico. Todas las actividades en una aplicación trabajan juntas para crear una experiencia de usuario perfecta.

Sintaxis

- void onCreate (Bundle savedInstanceState) // Se invoca cuando se inicia la actividad.
- void onPostCreate (Bundle savedInstanceState) // Llamado cuando se completa el inicio de la actividad (después de que se haya llamado a onStart () y onRestoreInstanceState (Bundle)).
- void onStart () // Llamado después de onCreate (Bundle) - o después de onRestart () cuando se detuvo la actividad, pero ahora se muestra nuevamente al usuario.
- void onResume () // Llamado después de onRestoreInstanceState (Bundle), onRestart () o onPause (), para que su actividad comience a interactuar con el usuario.
- void onPostResume () // Llamado cuando se completa la reanudación de la actividad (después de que se haya llamado a onResume ()).
- void onRestart () // Llamado después de onStop () cuando la actividad actual se muestra nuevamente al usuario (el usuario ha regresado a ella).
- void onPause () // Llamado como parte del ciclo de vida de la actividad cuando una actividad se pone en segundo plano, pero no se ha eliminado (todavía).
- void onStop () // Llamado cuando ya no eres visible para el usuario.
- void onDestroy () // Realice cualquier limpieza final antes de que se destruya una actividad.
- void onNewIntent (Intención de intención) // Esto se llama para actividades que configuran launchMode en "singleTop" en su paquete, o si un cliente usó el indicador FLAG_ACTIVITY_SINGLE_TOP al llamar a startActivity (Intent).
- void onSaveInstanceState (Bundle outState) // Llamado para recuperar el estado por instancia de una actividad antes de eliminarse para que el estado se pueda restaurar en onCreate (Bundle) o onRestoreInstanceState (Bundle) (el Bundle completado por este método se pasará a ambos).

- void onRestoreInstanceState (Bundle savedInstanceState) // Este método se llama después de onStart () cuando la actividad se está reinicializando desde un estado previamente guardado, que se proporciona aquí en savedInstanceState.

Parámetros

Parámetro	Detalles
Intención	Se puede usar con startActivity para lanzar una actividad
Haz	Una asignación de claves de cadena a varios valores parcelables .
Contexto	Interfaz con información global sobre un entorno de aplicación.

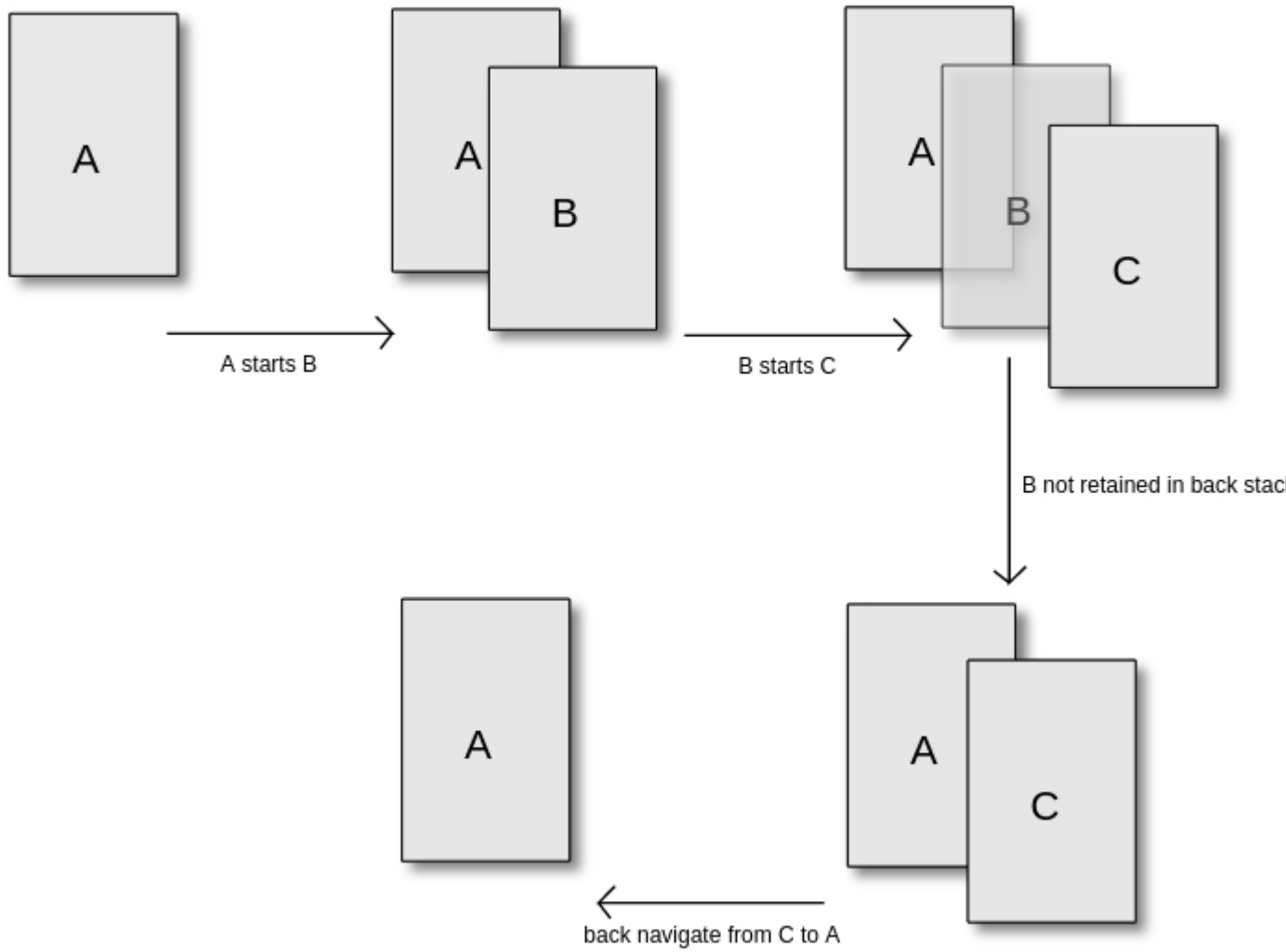
Observaciones

Una Actividad es un componente de la aplicación que proporciona una pantalla con la que los usuarios pueden interactuar para hacer algo, como marcar el teléfono, tomar una foto, enviar un correo electrónico o ver un mapa. A cada actividad se le da una ventana en la que dibujar su interfaz de usuario. La ventana normalmente llena la pantalla, pero puede ser más pequeña que la pantalla y flotar sobre otras ventanas.

Examples

Excluir una actividad del historial de back-stack

Deje que haya una Actividad `B` que pueda abrirse y que pueda iniciar más Actividades. Pero, el usuario no debe encontrarlo cuando navega hacia atrás en las actividades de tareas.



La solución más sencilla es establecer el atributo `noHistory` en `true` para esa etiqueta `<activity>` en `AndroidManifest.xml` :

```
<activity
     android:name=".B"
     android:noHistory="true">
```

Este mismo comportamiento también es posible desde el código si `B` llama a `finish()` antes de comenzar la siguiente actividad:

```
finish();
startActivity(new Intent(context, C.class));
```

El uso típico de la bandera `noHistory` es con "Pantalla de inicio" o Actividades de inicio de sesión.

Actividad de Android LifeCycle explicó

Supongamos una aplicación con MainActivity que puede llamar a la siguiente actividad con un clic

del botón.

```
public class MainActivity extends AppCompatActivity {

    private final String LOG_TAG = MainActivity.class.getSimpleName();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(LOG_TAG, "calling onCreate from MainActivity");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.d(LOG_TAG, "calling onStart from MainActivity");
    }
    @Override
    protected void onResume() {
        super.onResume();
        Log.d(LOG_TAG, "calling onResume  from MainActivity");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.d(LOG_TAG, "calling onPause  from MainActivity");
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.d(LOG_TAG, "calling onStop  from MainActivity");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d(LOG_TAG, "calling onDestroy  from MainActivity");
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Log.d(LOG_TAG, "calling onRestart  from MainActivity");
    }
    public void toNextActivity(){
        Log.d(LOG_TAG, "calling Next Activity");
        Intent intent = new Intent(this, NextActivity.class);
        startActivity(intent);
    } }
```

y

```
public class NextActivity extends AppCompatActivity {
    private final String LOG_TAG = NextActivity.class.getSimpleName();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_next);
```

```
        Log.d(LOG_TAG, "calling onCreate from Next Activity");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.d(LOG_TAG, "calling onStart from Next Activity");
    }
    @Override
    protected void onResume() {
        super.onResume();
        Log.d(LOG_TAG, "calling onResume  from Next Activity");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.d(LOG_TAG, "calling onPause  from Next Activity");
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.d(LOG_TAG, "calling onStop  from Next Activity");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d(LOG_TAG, "calling onDestroy  from Next Activity");
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Log.d(LOG_TAG, "calling onRestart  from Next Activity");
    } }
```

Cuando la aplicación se crea por primera vez
D / MainActivity: llamando a onCreate desde MainActivity
D / MainActivity: llamar a OnStart desde MainActivity
D / MainActivity: llamada onResume desde MainActivity
son llamados

Cuando la pantalla duerme
08: 11: 03.142 D / MainActivity: llamada onPause desde MainActivity
08: 11: 03.192 D / MainActivity: llamando a Stop desde MainActivity
son llamados. Y otra vez cuando se despierta.
08: 11: 55.922 D / MainActivity: llamando onRestart desde MainActivity
08: 11: 55.962 D / MainActivity: llamar a OnStart desde MainActivity
08: 11: 55.962 D / MainActivity: llamada onResume desde MainActivity
son llamados

Caso 1: cuando se llama a la siguiente actividad desde la actividad principal
D / MainActivity: llamando a la siguiente actividad
D / MainActivity: llamada onPause desde MainActivity

D / NextActivity: llamando a Crear desde la próxima actividad
D / NextActivity: llamar a OnStart desde la siguiente actividad
D / NextActivity: llamando a Currículum de la siguiente actividad
D / MainActivity: llamando a onStop desde MainActivity

Cuando regrese a la actividad principal de la siguiente actividad con el botón de retroceso
D / NextActivity: llamar en pausa desde la siguiente actividad
D / MainActivity: llamando a onRestart desde MainActivity
D / MainActivity: llamar a OnStart desde MainActivity
D / MainActivity: llamada onResume desde MainActivity
D / Próxima actividad: llamar a Stop desde la próxima actividad
D / Próxima actividad: llamar a destruir en la próxima actividad

Caso 2: cuando la actividad está parcialmente oculta (cuando se presiona el botón de vista general) o cuando la aplicación pasa al fondo y otra aplicación la oculta por completo
D / MainActivity: llamada onPause desde MainActivity
D / MainActivity: llamando a onStop desde MainActivity
y cuando la aplicación vuelva a estar en primer plano, lista para aceptar entradas de usuario,
D / MainActivity: llamando a onRestart desde MainActivity
D / MainActivity: llamar a OnStart desde MainActivity
D / MainActivity: llamada onResume desde MainActivity
son llamados

Caso3: cuando se llama a una actividad para cumplir una intención implícita y el usuario ha realizado una selección. Por ejemplo, cuando se presiona el botón Compartir y el usuario tiene que seleccionar una aplicación de la lista de aplicaciones que se muestra
D / MainActivity: llamada onPause desde MainActivity

La actividad es visible pero no está activa ahora. Cuando se realiza la selección y la aplicación está activa.
D / MainActivity: llamada onResume desde MainActivity
se llama

Caso4:
Cuando la aplicación se elimine en segundo plano (para liberar recursos para otra aplicación en primer plano), *onPause* (para el dispositivo anterior al panal) o *onStop* (ya que se trata de un dispositivo con forma de panal) será la última llamada antes de que finalice la aplicación.

onCreate y onDestroy se llamarán mayor cada vez que se ejecute la aplicación. Pero el onPause, onStop, onRestart, onStart, onResume puede ser llamado muchas veces durante el ciclo de vida.

Actividad launchMode

El modo de inicio define el comportamiento de la actividad nueva o existente en la tarea.
Hay posibles modos de lanzamiento:

- estándar
- singleTop

- sola tarea
- única instancia

Se debe definir en el manifiesto de Android en el elemento `<activity/>` como atributo `android:launchMode` .

```
<activity
    android:launchMode=["standard" | "singleTop" | "singleTask" | "singleInstance"] />
```

Estándar:

Valor por defecto. Si se establece este modo, siempre se creará una nueva actividad para cada nuevo intento. Así que es posible realizar muchas actividades del mismo tipo. La nueva actividad se colocará en la parte superior de la tarea. Hay algunas diferencias para diferentes versiones de Android: si la actividad se inicia desde otra aplicación, en androides <= 4.4 se colocará en la misma tarea que la aplicación de inicio, pero en> = 5.0 se creará una nueva tarea.

SingleTop:

Este modo es casi el mismo que el `standard` . Se podrían crear muchas instancias de actividad singleTop. La diferencia es que, si ya existe una instancia de actividad en la parte superior de la pila actual, se `onNewIntent()` lugar de crear una nueva instancia.

SingleTask:

La actividad con este modo de inicio solo puede tener una instancia **en el sistema** . Se creará una nueva tarea para la actividad, si no existe. De lo contrario, la tarea con actividad se moverá al frente y se `onNewIntent` .

Única instancia:

Este modo es similar a `singleTask` . La diferencia es que la tarea que contiene una actividad con `singleInstance` podría tener solo esta actividad y nada más. Cuando la actividad `singleInstance` crea otra actividad, se creará una nueva tarea para colocar esa actividad.

Presentando UI con setContentView

La clase de actividad se encarga de crear una ventana para ti en la que puedes colocar tu IU con `setContentView` .
Hay tres métodos `setContentView` :

- `setContentView(int layoutResID)` : establece el contenido de la actividad a partir de un recurso de diseño.
- `setContentView(View view)` : establece el contenido de la actividad en una vista explícita.
- `setContentView(View view, ViewGroup.LayoutParams params)` : establece el contenido de la actividad en una vista explícita con los parámetros proporcionados.

Cuando se llama a `setContentView` , esta vista se coloca directamente en la jerarquía de vistas de la actividad. Puede ser una jerarquía de vista compleja.

Ejemplos

Establecer contenido desde el archivo de recursos:

Agregue el archivo de recursos (main.xml en este ejemplo) con la jerarquía de vista:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello" />

</FrameLayout>
```

Establézcalo como contenido en actividad:

```
public final class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // The resource will be inflated,
        // adding all top-level views to the activity.
        setContentView(R.layout.main);
    }
}
```

Establecer contenido a una vista explícita:

```
public final class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Creating view with container
```

```
        final FrameLayout root = new FrameLayout(this);
        final TextView text = new TextView(this);
        text.setText("Hello");
        root.addView(text);

        // Set container as content view
        setContentView(root);
    }
}
```

Borra tu pila de actividades actual y lanza una nueva actividad

Si desea borrar su pila de actividades actual e iniciar una nueva actividad (por ejemplo, cerrar la sesión de la aplicación e iniciar un inicio de sesión en la actividad), parece haber dos enfoques.

1. Destino (API> = 16)

Llamando a `finishAffinity()` desde una actividad

2. Objetivo (11 <= API <16)

```
Intent intent = new Intent(this, LoginActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK
|Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
finish();
```

Finalizar la aplicación con excluir de Recientes

Primero defina una ExitActivity en el AndroidManifest.xml

```
<activity
        android:name="com.your_example_app.activities.ExitActivity"
        android:autoRemoveFromRecents="true"
        android:theme="@android:style/Theme.NoDisplay" />
```

Después la clase ExitActivity

```
/**
 * Activity to exit Application without staying in the stack of last opened applications
 */
public class ExitActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (Utils.hasLollipop()) {
            finishAndRemoveTask();
        } else if (Utils.hasJellyBean()) {
            finishAffinity();
        } else {
            finish();
        }
    }
```

```
   /**
    * Exit Application and Exclude from Recents
    *
    * @param context Context to use
    */
    public static void exitApplication(ApplicationContext context) {
        Intent intent = new Intent(context, ExitActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NO_ANIMATION | Intent.FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS);
        context.startActivity(intent);
    }
}
```

Navegación para actividades

La navegación hacia arriba se realiza en Android agregando `android:parentActivityName=""` en Manifest.xml a la etiqueta de actividad. Básicamente, con esta etiqueta usted le dice al sistema sobre la actividad principal de una actividad.

Como se hace

```
<uses-permission android:name="android.permission.INTERNET" />

<application
    android:name=".SkillSchoolApplication"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity
        android:name=".ui.activities.SplashActivity"
        android:theme="@style/SplashTheme">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".ui.activities.MainActivity" />
    <activity android:name=".ui.activities.HomeActivity"
     android:parentActivityName=".ui.activities.MainActivity/> // HERE I JUST TOLD THE SYSTEM
THAT MainActivity is the parent of HomeActivity
</application>
```

Ahora, cuando haga clic en la flecha dentro de la barra de herramientas de HomeActivity, volveré a la actividad principal.

Código Java

Aquí escribiré el código java apropiado requerido para esta funcionalidad.

```
public class HomeActivity extends AppCompatActivity {
    @BindView(R.id.toolbar)
    Toolbar toolbar;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        ButterKnife.bind(this);
        //Since i am using custom tool bar i am setting refernce of that toolbar to Actionbar.
If you are not using custom then you can simple leave this and move to next line
        setSupportActionBar(toolbar);
        getSupportActionBar.setDisplayHomeAsUpEnabled(true); // this will show the back arrow
in the tool bar.
}
}
```

Si ejecuta este código, verá que cuando presiona el botón Atrás, volverá a MainActivity. Para una mayor comprensión de la navegación hacia arriba recomendaría leer documentos

Puede personalizar más este comportamiento según sus necesidades al anular

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    // Respond to the action bar's Up/Home button
    case android.R.id.home:
        NavUtils.navigateUpFromSameTask(this); // Here you will write your logic for handling
up navigation
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

Hack simple

Este es un truco simple que se usa principalmente para navegar a la actividad principal si el padre está en backstack. Al llamar a `onBackPressed()` si id es igual a `android.R.id.home`

```
@Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        switch (id) {
            case android.R.id.home:
                onBackPressed();
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
```

Lea Actividad en línea: https://riptutorial.com/es/android/topic/1481/actividad

Capítulo 6: Actividades de pantalla dividida / multipantalla

Examples

Pantalla dividida introducida en Android Nougat implementado.

Establezca este atributo en su manifiesto o elemento para habilitar o deshabilitar la visualización de ventanas múltiples:

```
android:resizeableActivity=["true" | "false"]
```

Si este atributo se establece en verdadero, la actividad se puede iniciar en los modos de pantalla dividida y de forma libre. Si el atributo se establece en falso, la actividad no admite el modo de ventanas múltiples. Si este valor es falso, y el usuario intenta iniciar la actividad en el modo de ventanas múltiples, la actividad asume toda la pantalla.

Si su aplicación apunta al nivel de API 24, pero no especifica un valor para este atributo, el valor del atributo por defecto es verdadero.

El siguiente código muestra cómo especificar el tamaño y la ubicación predeterminados de una actividad, y su tamaño mínimo, cuando la actividad se muestra en modo libre:

```
<--These are default values suggested by google.-->
<activity android:name=".MyActivity">
<layout android:defaultHeight="500dp"
      android:defaultWidth="600dp"
      android:gravity="top|end"
      android:minHeight="450dp"
      android:minWidth="300dp" />
</activity>
```

Funciones deshabilitadas en modo multi-ventana

Ciertas funciones se deshabilitan o ignoran cuando un dispositivo está en modo de múltiples ventanas, porque no tienen sentido para una actividad que puede estar compartiendo la pantalla del dispositivo con otras actividades o aplicaciones. Tales características incluyen:

1. Algunas opciones de personalización de la IU del sistema están deshabilitadas; por ejemplo, las aplicaciones no pueden ocultar la barra de estado si no se ejecutan en modo de pantalla completa.

2. El sistema ignora los cambios en el atributo **android: screenOrientation** .

Si su aplicación apunta al nivel de API 23 o inferior

Si su aplicación apunta a un nivel de API 23 o inferior y el usuario intenta usar la aplicación en

modo de ventanas múltiples, el sistema redimensiona la aplicación a la fuerza a menos que la aplicación declare una orientación fija.

Si su aplicación no declara una orientación fija, debe iniciarla en un dispositivo con Android 7.0 o superior e intentar poner la aplicación en modo de pantalla dividida. Verifique que la experiencia del usuario sea aceptable cuando la aplicación se redimensione por la fuerza.

Si la aplicación declara una orientación fija, debe intentar poner la aplicación en modo de ventanas múltiples. Verifique que al hacerlo, la aplicación permanezca en modo de pantalla completa.

Lea Actividades de pantalla dividida / multipantalla en línea:
https://riptutorial.com/es/android/topic/7130/actividades-de-pantalla-dividida---multipantalla

Capítulo 7: ADB (Android Debug Bridge)

Introducción

ADB (Android Debug Bridge) es una herramienta de línea de comandos que se utiliza para comunicarse con una instancia de emulador o dispositivo Android conectado.

Descripción general de ADB

Una gran parte de este tema se dividió en adb shell

Observaciones

Lista de ejemplos movidos a adb shell :

- Otorgar y revocar permisos API 23+
- Envíe texto, tecla presionada y eventos táctiles al dispositivo Android a través de ADB
- Listar paquetes
- Grabando la pantalla
- Opciones de desarrollador abierto
- Establecer fecha / hora a través de adb
- Cambio de permisos de archivos usando el comando chmod
- Generando una transmisión "Boot Complete"
- Imprimir datos de la aplicación
- Ver contenido de almacenamiento externo / secundario
- http://stackoverflow.com/documentation/android/9408/adb-shell/29140/adb-shell
- matar un proceso dentro de un dispositivo Android

Examples

Imprimir lista detallada de dispositivos conectados

Para obtener una lista detallada de todos los dispositivos conectados a `adb` , escriba el siguiente comando en su terminal:

```
adb devices -l
```

Ejemplo de salida

```
List of devices attached
ZX1G425DC6             device usb:336592896X product:shamu model:Nexus_6 device:shamu
013e4e127e59a868       device usb:337641472X product:bullhead model:Nexus_5X device:bullhead
ZX1D229KCN             device usb:335592811X product:titan_retde model:XT1068
device:titan_umtsds
A50PL                  device usb:331592812X
```

- La primera columna es el número de serie del dispositivo. Si comienza con el `emulator-` , este dispositivo es un emulador.
- `usb:` la ruta del dispositivo en el subsistema USB.
- `product:` el código del producto del dispositivo. Esto es muy específico del fabricante, y como puede ver en el caso del dispositivo Archos `A50PL` anterior, puede estar en blanco.
- `model:` el modelo de dispositivo. Como `product` , puede estar vacío.
- `device:` el código del dispositivo. Esto también es muy específico del fabricante y puede estar vacío.

Leer información del dispositivo

Escribe el siguiente comando en tu terminal:

```
adb shell getprop
```

Esto imprimirá toda la información disponible en forma de pares clave / valor.

Solo puede leer información específica agregando el nombre de una clave específica al comando. Por ejemplo:

```
adb shell getprop ro.product.model
```

Aquí hay algunos datos interesantes que obtienes:

- `ro.product.model` : nombre del modelo del dispositivo (por ejemplo, Nexus 6P)
- `ro.build.version.sdk` : Nivel de API del dispositivo (por ejemplo, 23)
- `ro.product.brand` : marca del dispositivo (por ejemplo, Samsung)

Ejemplo completo de salida

```
[dalvik.vm.dex2oat-Xms]: [64m]
[dalvik.vm.dex2oat-Xmx]: [512m]
[dalvik.vm.heapsize]: [384m]
[dalvik.vm.image-dex2oat-Xms]: [64m]
[dalvik.vm.image-dex2oat-Xmx]: [64m]
[dalvik.vm.isa.x86.variant]: [dalvik.vm.isa.x86.features=default]
[dalvik.vm.isa.x86_64.features]: [default]
[dalvik.vm.isa.x86_64.variant]: [x86_64]
[dalvik.vm.lockprof.threshold]: [500]
[dalvik.vm.stack-trace-file]: [/data/anr/traces.txt]
[debug.atrace.tags.enableflags]: [0]
[debug.force_rtl]: [0]
[dev.bootcomplete]: [1]
[gsm.current.phone-type]: [1]
[gsm.defaultpdpcontext.active]: [true]
[gsm.network.type]: [UMTS]
[gsm.nitz.time]: [1469106902492]
[gsm.operator.alpha]: [Android]
[gsm.operator.iso-country]: [us]
[gsm.operator.isroaming]: [false]
[gsm.operator.numeric]: [310260]
[gsm.sim.operator.alpha]: [Android]
```

```
[gsm.sim.operator.iso-country]: [us]
[gsm.sim.operator.numeric]: [310260]
[gsm.sim.state]: [READY]
[gsm.version.ril-impl]: [android reference-ril 1.0]
[init.svc.adbd]: [running]
[init.svc.bootanim]: [stopped]
[init.svc.console]: [running]
[init.svc.debuggerd]: [running]
[init.svc.debuggerd64]: [running]
[init.svc.drm]: [running]
[init.svc.fingerprintd]: [running]
[init.svc.gatekeeperd]: [running]
[init.svc.goldfish-logcat]: [stopped]
[init.svc.goldfish-setup]: [stopped]
[init.svc.healthd]: [running]
[init.svc.installd]: [running]
[init.svc.keystore]: [running]
[init.svc.lmkd]: [running]
[init.svc.logd]: [running]
[init.svc.logd-reinit]: [stopped]
[init.svc.media]: [running]
[init.svc.netd]: [running]
[init.svc.perfprofd]: [running]
[init.svc.qemu-props]: [stopped]
[init.svc.ril-daemon]: [running]
[init.svc.servicemanager]: [running]
[init.svc.surfaceflinger]: [running]
[init.svc.ueventd]: [running]
[init.svc.vold]: [running]
[init.svc.zygote]: [running]
[init.svc.zygote_secondary]: [running]
[net.bt.name]: [Android]
[net.change]: [net.dns2]
[net.dns1]: [10.0.2.3]
[net.dns2]: [10.0.2.4]
[net.eth0.dns1]: [10.0.2.3]
[net.eth0.dns2]: [10.0.2.4]
[net.eth0.gw]: [10.0.2.2]
[net.gprs.local-ip]: [10.0.2.15]
[net.hostname]: [android-5e1af924d72dc578]
[net.qtaguid_enabled]: [1]
[net.tcp.default_init_rwnd]: [60]
[persist.sys.dalvik.vm.lib.2]: [libart.so]
[persist.sys.profiler_ms]: [0]
[persist.sys.timezone]: [Europe/Vienna]
[persist.sys.usb.config]: [adb]
[qemu.gles]: [1]
[qemu.hw.mainkeys]: [0]
[qemu.sf.fake_camera]: [none]
[qemu.sf.lcd_density]: [560]
[rild.libargs]: [-d /dev/ttyS0]
[rild.libpath]: [/system/lib/libreference-ril.so]
[ro.allow.mock.location]: [0]
[ro.baseband]: [unknown]
[ro.board.platform]: []
[ro.boot.hardware]: [ranchu]
[ro.bootimage.build.date]: [Thu Jul 7 15:56:30 UTC 2016]
[ro.bootimage.build.date.utc]: [1467906990]
[ro.bootimage.build.fingerprint]:
[Android/sdk_google_phone_x86_64/generic_x86_64:6.0/MASTER/3038907:userdebug/test-keys]
[ro.bootloader]: [unknown]
```

```
[ro.bootmode]: [unknown]
[ro.build.characteristics]: [emulator]
[ro.build.date]: [Thu Jul  7 15:55:30 UTC 2016]
[ro.build.date.utc]: [1467906930]
[ro.build.description]: [sdk_google_phone_x86_64-userdebug 6.0 MASTER 3038907 test-keys]
[ro.build.display.id]: [sdk_google_phone_x86_64-userdebug 6.0 MASTER 3038907 test-keys]
[ro.build.fingerprint]:
[Android/sdk_google_phone_x86_64/generic_x86_64:6.0/MASTER/3038907:userdebug/test-keys]
[ro.build.flavor]: [sdk_google_phone_x86_64-userdebug]
[ro.build.host]: [vpak15.mtv.corp.google.com]
[ro.build.id]: [MASTER]
[ro.build.product]: [generic_x86_64]
[ro.build.tags]: [test-keys]
[ro.build.type]: [userdebug]
[ro.build.user]: [android-build]
[ro.build.version.all_codenames]: [REL]
[ro.build.version.base_os]: []
[ro.build.version.codename]: [REL]
[ro.build.version.incremental]: [3038907]
[ro.build.version.preview_sdk]: [0]
[ro.build.version.release]: [6.0]
[ro.build.version.sdk]: [23]
[ro.build.version.security_patch]: [2015-10-01]
[ro.com.google.locationfeatures]: [1]
[ro.config.alarm_alert]: [Alarm_Classic.ogg]
[ro.config.nocheckin]: [yes]
[ro.config.notification_sound]: [OnTheHunt.ogg]
[ro.crypto.state]: [unencrypted]
[ro.dalvik.vm.native.bridge]: [0]
[ro.debuggable]: [1]
[ro.hardware]: [ranchu]
[ro.hardware.audio.primary]: [goldfish]
[ro.kernel.android.checkjni]: [1]
[ro.kernel.android.qemud]: [1]
[ro.kernel.androidboot.hardware]: [ranchu]
[ro.kernel.clocksource]: [pit]
[ro.kernel.console]: [0]
[ro.kernel.ndns]: [2]
[ro.kernel.qemu]: [1]
[ro.kernel.qemu.gles]: [1]
[ro.opengles.version]: [131072]
[ro.product.board]: []
[ro.product.brand]: [Android]
[ro.product.cpu.abi]: [x86_64]
[ro.product.cpu.abilist]: [x86_64,x86]
[ro.product.cpu.abilist32]: [x86]
[ro.product.cpu.abilist64]: [x86_64]
[ro.product.device]: [generic_x86_64]
[ro.product.locale]: [en-US]
[ro.product.manufacturer]: [unknown]
[ro.product.model]: [Android SDK built for x86_64]
[ro.product.name]: [sdk_google_phone_x86_64]
[ro.radio.use-ppp]: [no]
[ro.revision]: [0]
[ro.runtime.firstboot]: [1469106908722]
[ro.secure]: [1]
[ro.serialno]: []
[ro.wifi.channels]: []
[ro.zygote]: [zygote64_32]
[selinux.reload_policy]: [1]
[service.bootanim.exit]: [1]
```

```
[status.battery.level]: [5]
[status.battery.level_raw]: [50]
[status.battery.level_scale]: [9]
[status.battery.state]: [Slow]
[sys.boot_completed]: [1]
[sys.sysctl.extra_free_kbytes]: [43200]
[sys.sysctl.tcp_def_init_rwnd]: [60]
[sys.usb.config]: [adb]
[sys.usb.state]: [adb]
[vold.has_adoptable]: [1]
[wlan.driver.status]: [unloaded]
[xmpp.auto-presence]: [true]
```

Conecta ADB a un dispositivo a través de WiFi

La configuración estándar de ADB implica una conexión USB a un dispositivo físico.
Si lo prefiere, puede cambiar al modo TCP / IP y, en su lugar, conectar ADB a través de WiFi.

Dispositivo no rooteado

1. **Entrar en la misma red:**
    - Asegúrese de que su dispositivo y su computadora estén en la misma red.
2. **Conecte el dispositivo a la computadora host con un cable USB.**
3. **Conecte `adb` al dispositivo a través de la red:**

    Mientras su dispositivo está conectado a `adb` través de USB, realice el siguiente comando para escuchar una conexión TCP / IP en un puerto (predeterminado 5555):

    - Escriba `adb tcpip <port>` (cambie al modo TCP / IP).
    - Desconecte el cable USB del dispositivo de destino.
    - Escriba `adb connect <ip address>:<port>` (el puerto es opcional; predeterminado 5555).

    Por ejemplo:

    ```
    adb tcpip 5555
    adb connect 192.168.0.101:5555
    ```

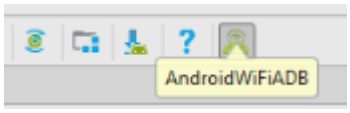
    Si no conoce la IP de su dispositivo, puede:

    - Compruebe la IP en la configuración de WiFi de su dispositivo.
    - use ADB para descubrir IP (a través de USB):
        1. Conecta el dispositivo a la computadora a través de USB
        2. En una línea de comando, escriba `adb shell ifconfig` y copie la dirección IP de su dispositivo

    Para **volver** a la depuración a través de **USB,** use el siguiente comando:

```
adb usb
```

También puede conectar ADB a través de WiFi mediante la instalación de un complemento para Android Studio. Para hacerlo, vaya a *Configuración> Complementos* y repositorios de navegación, busque *ADB WiFi* , instálelo y vuelva a abrir Android Studio. Verá un nuevo icono en su barra de herramientas como se muestra en la siguiente imagen. Conecte el dispositivo al ordenador host mediante USB y haga clic en este icono de *AndroidWiFiADB* . Mostrará un mensaje si su dispositivo está conectado o no. Una vez que se conecta puede desconectar su USB.



Dispositivo rooteado

Nota: Algunos dispositivos que **están rooteados** pueden usar la aplicación WiFi ADB de Play Store para habilitar esto de una manera simple. Además, para ciertos dispositivos (especialmente aquellos con ROM de CyanogenMod), esta opción está presente en las Opciones de Desarrollador entre las Configuraciones. Habilitarlo le dará la dirección IP y el número de puerto necesarios para conectarse a `adb` simplemente ejecutando `adb connect <ip address>:<port>` .

Cuando tienes un dispositivo rooteado pero no tienes acceso a un cable USB

El proceso se explica en detalle en la siguiente respuesta: http://stackoverflow.com/questions/2604727/how-can-i-connect-to-android-with-adb-over-tcp/3623727#3623727 Los comandos más importantes se muestran a continuación.

Abra un terminal en el dispositivo y escriba lo siguiente:

```
su
setprop service.adb.tcp.port <a tcp port number>
stop adbd
start adbd
```

Por ejemplo:

```
setprop service.adb.tcp.port 5555
```

Y en tu computadora:

```
adb connect <ip address>:<a tcp port number>
```

Por ejemplo:

```
adb connect 192.168.1.2:5555
```

Para apagarlo:

```
setprop service.adb.tcp.port -1
stop adbd
start adbd
```

Evitar el tiempo de espera

Por defecto, `adb` se agotará después de 5000 ms. Esto puede suceder en algunos casos, como WiFi lento o APK grande.

Un simple cambio en la configuración de Gradle puede hacer el truco:

```
android {
    adbOptions {
        timeOutInMs 10 * 1000
    }
}
```

Tire de (empuje) archivos desde (hacia) el dispositivo

Puede extraer (descargar) archivos del dispositivo ejecutando el siguiente comando:

```
adb pull <remote> <local>
```

Por ejemplo:

```
adb pull /sdcard/ ~/
```

También puede enviar (cargar) archivos desde su computadora al dispositivo:

```
adb push <local> <remote>
```

Por ejemplo:

```
adb push ~/image.jpg /sdcard/
```

Ejemplo para recuperar la base de datos del dispositivo

```
sudo adb -d shell "run-as com.example.name cat /data/da/com.example.name
/databases/DATABASE_NAME  > /sdcard/file
```

Reiniciar dispositivo

Puede reiniciar su dispositivo ejecutando el siguiente comando:

```
adb reboot
```

Ejecuta este comando para reiniciar en el gestor de arranque:

```
adb reboot bootloader
```

Reinicie al modo de recuperación:

```
adb reboot recovery
```

¡Tenga en cuenta que el dispositivo no se apagará primero!

Encender / apagar Wifi

Encender:

```
adb shell svc wifi enable
```

Apagar:

```
adb shell svc wifi disable
```

Ver dispositivos disponibles

Mando:

```
 adb devices
```

Ejemplo de resultado:

```
List of devices attached
emulator-5554  device
PhoneRT45Fr54  offline
123.454.67.45  no device
```

Primera columna - número de serie del dispositivo

Segunda columna - estado de conexión

Documentación de Android

Conectar dispositivo por IP

Ingrese estos comandos en el terminal de dispositivo Android

```
su
setprop service.adb.tcp.port 5555
stop adbd
start adbd
```

Después de esto, puede usar **CMD** y **ADB** para conectarse usando el siguiente comando

```
adb connect 192.168.0.101.5555
```

Y puede deshabilitarlo y volver a ADB a escuchar en USB con

```
setprop service.adb.tcp.port -1
stop adbd
start adbd
```

Desde una computadora, si ya tiene acceso USB (no se requiere root)

Es incluso más fácil cambiar a usar Wi-Fi, si ya tiene USB. Desde una línea de comandos en la computadora que tiene el dispositivo conectado a través de USB, ejecute los comandos

```
adb tcpip 5555
adb connect 192.168.0.101:5555
```

Reemplace 192.168.0.101 con el dispositivo IP

Iniciar / detener adb

Iniciar ADB:

```
adb kill-server
```

Detener ADB:

```
adb start-server
```

Ver logcat

Puede ejecutar `logcat` como un comando adb o directamente en un indicador de shell de su emulador o dispositivo conectado. Para ver la salida del registro usando `adb` , navegue a su plataforma-herramientas / directorio SDK y ejecute:

```
$ adb logcat
```

Alternativamente, puede crear una conexión de shell a un dispositivo y luego ejecutar:

```
$ adb shell
$ logcat
```

Un comando útil es:

```
adb logcat -v threadtime
```

Esto muestra la fecha, la hora de invocación, la prioridad, la etiqueta y el PID y TID del hilo que emite el mensaje en un formato de mensaje largo.

Filtración

Logcat logs obtuvo los llamados niveles de registro:

> **V** - Verbosa, **D** - Depuración, **I** - Información, **W** - Advertencia, **E** - Error, **F** - Fatal, **S** - Silencio

También puede filtrar logcat por nivel de registro. Por ejemplo, si solo desea generar un nivel de depuración:

```
adb logcat *:D
```

Logcat se puede filtrar por un nombre de paquete, por supuesto, puede combinarlo con el filtro de nivel de registro:

```
adb logcat <package-name>:<log level>
```

También puede filtrar el registro usando grep (más información sobre cómo filtrar la salida logcat aquí):

```
adb logcat | grep <some text>
```

En Windows, el filtro se puede usar usando findstr, por ejemplo:

```
adb logcat | findstr <some text>
```

Para ver el búfer de registro alternativo [main | events | radio], ejecute el `logcat` con la opción `-b` :

```
adb logcat -b radio
```

Guardar salida en el archivo:

```
adb logcat > logcat.txt
```

Guarda la salida en el archivo mientras también lo miras:

```
adb logcat | tee logcat.txt
```

Limpiando los troncos:

```
adb logcat -c
```

Dirigir el comando ADB a un dispositivo específico en una configuración de

múltiples dispositivos

1. Dirigir un dispositivo por número de serie

Use la opción `-s` seguida de un nombre de dispositivo para seleccionar en qué dispositivo debe ejecutarse el comando `adb` . Las opciones `-s` deben ser las primeras en la línea, **antes** del comando.

```
adb -s <device> <command>
```

Ejemplo:

```
adb devices

List of devices attached
emulator-5554       device
02157df2d1faeb33    device

adb -s emulator-5554 shell
```

Ejemplo # 2:

```
adb devices -l

List of devices attached
06157df65c6b2633    device usb:1-3 product:zerofltexx model:SM_G920F device:zeroflte
LC62TB413962        device usb:1-5 product:a50mgp_dug_htc_emea model:HTC_Desire_820G_dual_sim
device:htc_a50mgp_dug

adb -s usb:1-3 shell
```

2. Dirigirse a un dispositivo, cuando solo hay conectado un tipo de dispositivo

Puedes apuntar al único emulador en ejecución con -e

```
adb -e <command>
```

O puede apuntar al único dispositivo USB conectado con -d

```
adb -d <command>
```

Captura de pantalla y video (solo para kitkat) desde la pantalla del dispositivo

Captura de pantalla: Opción 1 (adb puro)

El comando `shell` adb nos permite ejecutar comandos utilizando el shell integrado de un dispositivo. El comando de shell `screencap` captura el contenido actualmente visible en un dispositivo y lo guarda en un archivo de imagen dado, por ejemplo, `/sdcard/screen.png` :

```
adb shell screencap /sdcard/screen.png
```

Luego puede usar el comando de extracción para descargar el archivo desde el dispositivo al directorio actual en su computadora:

```
adb pull /sdcard/screen.png
```

Captura de pantalla: Opción 2 (más rápido)

Ejecutar el siguiente de una sola línea:

(Marshmallow y anteriores):

```
adb shell screencap -p | perl -pe 's/\x0D\x0A/\x0A/g' > screen.png
```

(Turrón y posteriores):

```
adb shell screencap -p > screen.png
```

El indicador `-p` redirige la salida del comando `screencap` a la salida estándar. La expresión de Perl en la que se canaliza limpia algunos problemas de final de línea en Marshmallow y versiones anteriores. La secuencia se escribe en un archivo llamado `screen.png` dentro del directorio actual. Vea este artículo y este artículo para más información.

Vídeo

Esto solo funciona en KitKat y solo a través de ADB. Esto no funciona debajo de Kitkat Para comenzar a grabar la pantalla de su dispositivo, ejecute el siguiente comando:

`adb shell screenrecord /sdcard/example.mp4` , este comando comenzará a grabar la pantalla de su dispositivo usando la configuración predeterminada y guardará el video resultante en un archivo en `/sdcard/example.mp4` en su dispositivo.

Cuando haya terminado de grabar, presione Ctrl + C (z en Linux) en la ventana del símbolo del sistema para detener la grabación de la pantalla. Luego puede encontrar el archivo de grabación de pantalla en la ubicación que especificó. Tenga en cuenta que la grabación de la pantalla se guarda en el almacenamiento interno de su dispositivo, no en su computadora.

La configuración predeterminada es usar la resolución de pantalla estándar de su dispositivo, codificar el video a una tasa de bits de 4 Mbps y establecer el tiempo máximo de grabación de pantalla en 180 segundos. Para obtener más información sobre las opciones de línea de comandos que puede usar, ejecute el siguiente comando:

`adb shell screenrecord -help` , esto funciona sin enraizar el dispositivo. Espero que esto ayude.

Borrar datos de la aplicación

Uno puede borrar los datos de usuario de una aplicación específica usando `adb` :

```
adb shell pm clear <package>
```

Esto es lo mismo que para navegar por la configuración del teléfono, seleccionar la aplicación y presionar el botón de borrar datos.

- `pm` invoca el gestor de paquetes en el dispositivo
- `clear` borra todos los datos asociados con un paquete

Enviando transmisión

Es posible enviar transmisión a `BroadcastReceiver` con `adb` .

En este ejemplo, estamos enviando difusión con la acción `com.test.app.ACTION` y la cadena extra en el paquete `'foo'='bar'` :

```
adb shell am broadcast -a action com.test.app.ACTION --es foo "bar"
```

Puede incluir cualquier otro tipo compatible en el paquete, no solo las cadenas:

--ez - booleano
--ei - entero
--el - largo
--ef - flotar
--eu - uri
--eia - int array (separado por ',')
--ela - matriz larga (separada por ',')
--efa - matriz flotante (separada por ',')
--esa - cadena de cadenas (separadas por ',')

Para enviar la intención a un paquete específico / clase `-n` o `-p` se puede usar el parámetro. Enviando al paquete:

```
-p com.test.app
```

Envío a un componente específico (`SomeReceiver` clase en `com.test.app package`):

```
-n com.test.app/.SomeReceiver
```

Ejemplos utiles:

- Enviando una transmisión de "arranque completo"
- Enviar una transmisión de "hora modificada" después de configurar la hora mediante el comando adb

Instalar y ejecutar una aplicación

Para instalar un archivo APK , use el siguiente comando:

```
adb install path/to/apk/file.apk
```

O si la aplicación ya existe y queremos reinstalarla.

```
adb install -r path/to/apk/file.apk
```

Para desinstalar una aplicación , tenemos que especificar su paquete.

```
adb uninstall application.package.name
```

Use el siguiente comando para iniciar una aplicación con un nombre de paquete provisto (o una actividad específica en una aplicación):

```
adb shell am start -n adb shell am start <package>/<activity>
```

Por ejemplo, para iniciar Waze:

```
adb shell am start -n adb shell am start com.waze/com.waze.FreeMapAppActivity
```

Apoyo

Puede usar el comando `adb backup` para hacer una copia de seguridad de su dispositivo.

```
adb backup [-f <file>] [-apk|-noapk] [-obb|-noobb] [-shared|-noshared] [-all]
           [-system|nosystem] [<packages...>]
```

`-f <filename>` especifique el nombre de archivo **predeterminado:** *crea backup.ab en el directorio actual*

`-apk|noapk` habilita / deshabilita la copia de seguridad de .apks ellos mismos por **defecto:** *-noapk*

`-obb|noobb` habilita / deshabilita la copia de seguridad de archivos adicionales por **defecto:** *-noobb*

`-shared|noshared` compartido de la tarjeta SD / almacenamiento compartido del dispositivo de copia de seguridad no compartidos por **defecto:** *-noshared*

`-all` respaldan todas las aplicaciones instaladas.

`-system|nosystem` incluye las aplicaciones del sistema por **defecto:** *-system*

`<packages>` una lista de paquetes para realizar copias de seguridad (por ejemplo com.example.android.myapp) (no es necesario si `-all` se especifica)

Para una copia de seguridad completa del dispositivo, incluyendo todo, use

```
adb backup -apk -obb -shared -all -system -f fullbackup.ab
```

> **Nota:** Hacer una copia de seguridad completa puede llevar mucho tiempo.

Para restaurar una copia de seguridad, utilice

```
adb restore backup.ab
```

Instalar ADB en el sistema Linux

Cómo instalar el Puente de depuración de Android (ADB) en un sistema Linux con el terminal utilizando los repositorios de su distro.

Instalar en el sistema Ubuntu / Debian a través de apt:

```
sudo apt-get update
sudo apt-get install adb
```

Instalar en el sistema Fedora / CentOS a través de yum:

```
sudo yum check-update
sudo yum install android-tools
```

Instalar en el sistema Gentoo con portage:

```
sudo emerge --ask dev-util/android-tools
```

Instalar en el sistema openSUSE con zypper:

```
sudo zypper refresh
sudo zypper install android-tools
```

Instalar en el sistema Arch con pacman:

```
sudo pacman -Syyu
sudo pacman -S android-tools
```

Listar todos los permisos que requieren la concesión de tiempo de ejecución de los usuarios en Android 6.0

```
adb shell pm list permissions -g -d
```

Ver los datos internos de una aplicación (datos / datos /) en un dispositivo

Primero, asegúrese de que se pueda hacer una copia de seguridad de su aplicación en `AndroidManifest.xml` , es decir, `android:allowBackup` no es `false` .

Comando de copia de seguridad:

```
adb -s <device_id> backup -noapk <sample.package.id>
```

Crea un tar con el comando dd:

```
dd if=backup.ab bs=1 skip=24 | python -c "import
zlib,sys;sys.stdout.write(zlib.decompress(sys.stdin.read()))" > backup.tar
```

Extraer el alquitrán:

```
tar -xvf backup.tar
```

A continuación, puede ver el contenido extraído.

Ver pila de actividades

```
adb -s <serialNumber> shell dumpsys activity activities
```

Muy útil cuando se usa junto con el comando `watch` unix:

```
watch -n 5 "adb -s <serialNumber> shell dumpsys activity activities | sed -En -e '/Stack #/p'
-e '/Running activities/,/Run #0/p'"
```

Ver y extraer archivos de caché de una aplicación

Puede usar este comando para enumerar los archivos para su propia apk debuggable:

```
adb shell run-as <sample.package.id> ls /data/data/sample.package.id/cache
```

Y esta secuencia de comandos para extraer de la memoria caché, primero copia el contenido a sdcard, extrae y luego lo elimina al final:

```
#!/bin/sh
adb shell "run-as <sample.package.id> cat '/data/data/<sample.package.id>/$1' > '/sdcard/$1'"
adb pull "/sdcard/$1"
adb shell "rm '/sdcard/$1'"
```

Luego puedes extraer un archivo de la memoria caché de esta manera:

```
./pull.sh cache/someCachedData.txt
```

Obtener archivo de base de datos a través de ADB

```
sudo adb -d shell "run-as com.example.name cat /data/da/com.example.name
/databases/STUDENT_DATABASE  > /sdcard/file
```

Lea ADB (Android Debug Bridge) en línea: https://riptutorial.com/es/android/topic/1051/adb--android-debug-bridge-

Capítulo 8: AdMob

Sintaxis

- compile 'com.google.firebase: firebase-ads: 10.2.1' // NOTA: CONFIGURAR LA VERSIÓN MÁS NUEVA SI ESTÁ DISPONIBLE
- `<uses-permission android:name="android.permission.INTERNET" />` Necesario para recuperar el anuncio
- AdRequest adRequest = new AdRequest.Builder (). Build (); // Banner publicitario
- AdView mAdView = (AdView) findViewById (R.id.adView); // Banner publicitario
- mAdView.loadAd (adRequest); // Banner publicitario

Parámetros

Param	Detalles
ads: adUnitId = "@ string / main_screen_ad"	La identificación de su anuncio. Obtenga su ID del sitio admob. *"Si bien no es un requisito, almacenar los valores de ID de su bloque de anuncios en un archivo de recursos es una buena práctica. A medida que su aplicación crezca y sus necesidades de publicación de anuncios maduren, puede ser necesario cambiar los valores de ID. Si los mantiene en un recurso archivo, nunca tendrá que buscar a través de su código buscándolos ".* [1]

Observaciones

- Requiere una cuenta Admob válida
- Lea la política de admob . Asegúrese de no hacer nada que pueda suspender su cuenta admob

Examples

Implementar

Nota: este ejemplo requiere una cuenta Admob válida y un código de anuncio Admob válido.

Build.gradle en el nivel de aplicación

Cambie a la última versión si existe:

```
compile 'com.google.firebase:firebase-ads:10.2.1'
```

Manifiesto

Se requiere permiso de Internet para acceder a los datos del anuncio. Tenga en cuenta que este permiso no tiene que ser solicitado (usando API 23+) ya que es un permiso normal y no peligroso:

```
<uses-permission android:name="android.permission.INTERNET" />
```

XML

El siguiente ejemplo XML muestra un anuncio de banner:

```
<com.google.android.gms.ads.AdView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/adView"
    ads:adSize="BANNER"
    ads:adUnitId="@string/main_screen_ad" />
```

Para el código de otros tipos, consulte la Ayuda de Google AdMob .

Java

El siguiente código es para la integración de anuncios de banner. Tenga en cuenta que otros tipos de anuncios pueden requerir una integración diferente:

```
// Alternative for faster initialization.
// MobileAds.initialize(getApplicationContext(), "AD_UNIT_ID");

AdView mAdView = (AdView) findViewById(R.id.adView);
// Add your device test ID if you are doing testing before releasing.
// The device test ID can be found in the admob stacktrace.
AdRequest adRequest = new AdRequest.Builder().build();
mAdView.loadAd(adRequest);
```

Agregue los métodos del ciclo de vida de `AdView` en los métodos `onResume()` , `onPause()` y `onDestroy()` de su actividad:

```
@Override
public void onPause() {
    if (mAdView != null) {
        mAdView.pause();
    }
    super.onPause();
}

@Override
public void onResume() {
    super.onResume();
```

```
    if (mAdView != null) {
        mAdView.resume();
    }
}

@Override
public void onDestroy() {
    if (mAdView != null) {
        mAdView.destroy();
    }
    super.onDestroy();
}
```

Lea AdMob en línea: https://riptutorial.com/es/android/topic/5334/admob

Capítulo 9: Advertencias de la pelusa

Observaciones

La **herramienta Lint** comprueba los archivos de origen de su proyecto Android para detectar posibles errores y mejoras de optimización para la corrección, seguridad, rendimiento, facilidad de uso, accesibilidad e internacionalización. Puede ejecutar Lint desde la línea de comandos o desde Android Studio.

Documentación oficial:

https://developer.android.com/studio/write/lint.html

Examples

Usando herramientas: ignorar en archivos xml

Las `tools:ignore` atributos `tools:ignore` se pueden usar en archivos xml para descartar las advertencias de pelusas.

PERO descartar advertencias de pelusas con esta técnica es la mayoría de las veces la forma incorrecta de proceder.

Una advertencia de pelusas debe ser entendida y reparada ... puede ignorarse solo si tiene un entendimiento completo de su significado y una razón importante para ignorarlo.

Aquí hay un caso de uso donde es legítimo ignorar una advertencia de pelusa:

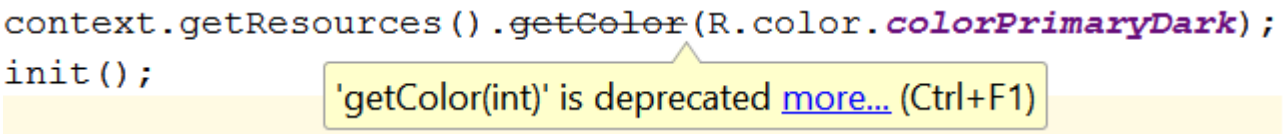
- Está desarrollando una aplicación de sistema (firmada con la clave del fabricante del dispositivo)
- Su aplicación necesita cambiar la fecha del dispositivo (o cualquier otra acción protegida)

Entonces puede hacer esto en su manifiesto: (es decir, solicitar el permiso protegido e ignorar la advertencia de pelusa porque sabe que en su caso se otorgará el permiso)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      ...>
    <uses-permission android:name="android.permission.SET_TIME"
        tools:ignore="ProtectedPermissions"/>
```

Importando recursos sin error "En desuso"

Usando la API de Android 23 o superior, muy a menudo tal situación se puede ver:



Esta situación es causada por el cambio estructural de la API de Android con respecto a obtener los recursos. Ahora la función:

```
public int getColor(@ColorRes int id, @Nullable Theme theme) throws NotFoundException
```

debería ser usado. Pero la biblioteca android.support.v4 tiene otra solución.

Agregue la siguiente dependencia al archivo build.gradle:

```
com.android.support:support-v4:24.0.0
```

Entonces todos los métodos de la biblioteca de soporte están disponibles:

```
ContextCompat.getColor(context, R.color.colorPrimaryDark);
ContextCompat.getDrawable(context, R.drawable.btn_check);
ContextCompat.getColorStateList(context, R.color.colorPrimary);
DrawableCompat.setTint(drawable);
ContextCompat.getColor(context,R.color.colorPrimaryDark));
```

Además, se pueden utilizar más métodos de la biblioteca de soporte:

```
ViewCompat.setElevation(textView, 1F);
ViewCompat.animate(textView);
TextViewCompat.setTextAppearance(textView, R.style.AppThemeTextStyle);
...
```

Configurar LintOptions con gradle

Puede configurar la pelusa agregando una sección `lintOptions` en el archivo `build.gradle` :

```
android {

    //.....

    lintOptions {
        // turn off checking the given issue id's
        disable 'TypographyFractions','TypographyQuotes'

        // turn on the given issue id's
        enable 'RtlHardcoded','RtlCompat', 'RtlEnabled'

        // check *only* the given issue id's
        check 'NewApi', 'InlinedApi'

       // set to true to turn off analysis progress reporting by lint
       quiet true

       // if true, stop the gradle build if errors are found
       abortOnError false
```

```
        // if true, only report errors
        ignoreWarnings true
    }
}
```

Puede ejecutar lint para una variante específica (ver más abajo), por ejemplo `./gradlew lintRelease` , o para todas las variantes (`./gradlew lint`), en cuyo caso produce un informe que describe a qué variantes específicas se aplica un problema determinado.

Consulte aquí la referencia DSL para todas las opciones disponibles .

Cómo configurar el archivo lint.xml

Puede especificar sus preferencias de control de pelusa en el archivo `lint.xml` . Si está creando este archivo manualmente, colóquelo en el directorio raíz de su proyecto de Android. Si está configurando las preferencias de Lint en Android Studio, el archivo lint.xml se crea automáticamente y se agrega a su proyecto de Android para usted.

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
    <lint>
        <!-- list of issues to configure -->
</lint>
```

Al establecer el valor del atributo de severidad en la etiqueta, puede deshabilitar la verificación de Lint para un problema o cambiar el nivel de severidad para un problema.

El siguiente ejemplo muestra el contenido de un archivo `lint.xml` .

```
<?xml version="1.0" encoding="UTF-8"?>
<lint>
    <!-- Disable the given check in this project -->
    <issue id="IconMissingDensityFolder" severity="ignore" />

    <!-- Ignore the ObsoleteLayoutParam issue in the specified files -->
    <issue id="ObsoleteLayoutParam">
        <ignore path="res/layout/activation.xml" />
        <ignore path="res/layout-xlarge/activation.xml" />
    </issue>

    <!-- Ignore the UselessLeaf issue in the specified file -->
    <issue id="UselessLeaf">
        <ignore path="res/layout/main.xml" />
    </issue>

    <!-- Change the severity of hardcoded strings to "error" -->
    <issue id="HardcodedText" severity="error" />
</lint>
```

Configuración de la comprobación de pelusas en archivos fuente de Java y XML

Puede deshabilitar la comprobación de Lint desde sus archivos de origen Java y XML.

Configurando la comprobación de pelusas en Java

Para deshabilitar la verificación de Lint específicamente para una **clase** o método **Java** en su proyecto de Android, agregue la **anotación** `@SuppressLint` a ese código Java.

Ejemplo:

```
@SuppressLint("NewApi")
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
```

Para deshabilitar la comprobación de todos los problemas de Lint:

```
@SuppressLint("all")
```

Configurando la comprobación de pelusas en XML

Puede usar las `tools:ignore` atributo para deshabilitar la verificación de Lint para secciones específicas de sus **archivos XML** .

Por ejemplo:

```
tools:ignore="NewApi,StringFormatInvalid"
```

Para suprimir la comprobación de todos los problemas de Lint en el elemento XML, use

```
tools:ignore="all"
```

Marca suprimir advertencias

Es una buena práctica marcar algunas advertencias en su código. Por ejemplo, algunos métodos en desuso son necesarios para su prueba, o versión de soporte anterior. Pero la comprobación de pelusa marcará ese código con advertencias. Para evitar este problema, necesita usar la anotación @SuppressWarnings.

Por ejemplo, agregue ignorar las advertencias a métodos en desuso. También hay que poner la descripción de las advertencias en la anotación:

```
@SuppressWarnings("deprecated");
public void setAnotherColor (int newColor) {
    getApplicationContext().getResources().getColor(newColor)
}
```

Usando esta anotación puede ignorar todas las advertencias, incluyendo Lint, Android y otras. Usando Suppress Warnings, ayuda a entender el código correctamente!

Lea Advertencias de la pelusa en línea: https://riptutorial.com/es/android/topic/129/advertencias-de-la-pelusa

Capítulo 10: AIDL

Introducción

AIDL es el lenguaje de definición de la interfaz de Android.

¿Qué? ¿Por qué? Cómo ?

¿Qué? Es un servicio acotado. Este servicio AIDL estará activo hasta que al menos exista uno de los clientes. Funciona basándose en el concepto de cálculo de referencias y desvinculación.

¿Por qué? Las aplicaciones remotas pueden acceder a su servicio + Multi Threading (solicitud de aplicación remota).

¿Cómo? Crear el archivo .aidl Implementar la interfaz Exponer la interfaz a los clientes

Examples

Servicio AIDL

ICalculator.aidl

```
// Declare any non-default types here with import statements

    interface ICalculator {
        int add(int x,int y);
        int sub(int x,int y);
    }
```

AidlService.java

```
public class AidlService extends Service {

    private static final String TAG = "AIDLServiceLogs";
    private static final String className = " AidlService";

    public AidlService() {
        Log.i(TAG, className+" Constructor");
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        Log.i(TAG, className+" onBind");
        return iCalculator.asBinder();
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.i(TAG, className+" onCreate");
```

```
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.i(TAG, className+" onDestroy");
    }


    ICalculator.Stub iCalculator = new ICalculator.Stub() {
        @Override
        public int add(int x, int y) throws RemoteException {
            Log.i(TAG, className+" add Thread Name: "+Thread.currentThread().getName());
            int z = x+y;
            return z;
        }

        @Override
        public int sub(int x, int y) throws RemoteException {
            Log.i(TAG, className+" add Thread Name: "+Thread.currentThread().getName());
            int z = x-y;
            return z;
        }
    };

}
```

Conexión de servicio

```
 // Return the stub as interface
ServiceConnection serviceConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            Log.i(TAG, className + " onServiceConnected");
            iCalculator = ICalculator.Stub.asInterface(service);
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {

            unbindService(serviceConnection);
        }
    };
```

Lea AIDL en línea: https://riptutorial.com/es/android/topic/9504/aidl

Capítulo 11: AlarmManager

Examples

Ejecutar una intención en un momento posterior

1. Crea un receptor. Esta clase recibirá la intención y la manejará como desee.

```
public class AlarmReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        // Handle intent
        int reqCode = intent.getExtras().getInt("requestCode");
        ...
    }
}
```

2. Dar un intento de AlarmManager. Este ejemplo activará la intención de ser enviado a AlarmReceiver después de 1 minuto.

```
final int requestCode = 1337;
AlarmManager am = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
Intent intent = new Intent(context, AlarmReceiver.class);
PendingIntent pendingIntent = PendingIntent.getBroadcast(context, requestCode, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
am.set( AlarmManager.RTC_WAKEUP, System.currentTimeMillis() + 60000 , pendingIntent );
```

Cómo cancelar una alarma

Si desea cancelar una alarma y no tiene una referencia al PendingIntent original utilizado para configurar la alarma, debe volver a crear un PendingIntent exactamente como estaba cuando se creó originalmente.

El Administrador de alarmas considera que una intención es igual :

> si su acción, datos, tipo, clase y categorías son iguales. Esto no compara ningún dato adicional incluido en los intentos.

Normalmente, el código de solicitud para cada alarma se define como una constante:

```
public static final int requestCode = 9999;
```

Entonces, para una alarma tan simple como esta:

```
Intent intent  = new Intent(this, AlarmReceiver.class);
intent.setAction("SomeAction");
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, requestCode, intent,
```

```
PendingIntent.FLAG_UPDATE_CURRENT);
AlarmManager alarmManager = (AlarmManager)getSystemService(Context.ALARM_SERVICE);
alarmManager.setExact(AlarmManager.RTC_WAKEUP, targetTimeInMillis, pendingIntent);
```

Aquí es cómo crearía una nueva referencia PendingIntent que puede usar para cancelar la alarma con una nueva referencia de AlarmManager:

```
Intent intent  = new Intent(this, AlarmReceiver.class);
intent.setAction("SomeAction");
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, requestCode, intent,
PendingIntent.FLAG_NO_CREATE);
AlarmManager alarmManager = (AlarmManager)getSystemService(Context.ALARM_SERVICE);
if(pendingIntent != null) {
    alarmManager.cancel(pendingIntent);
}
```

Creando alarmas exactas en todas las versiones de Android.

A `AlarmManager` se `AlarmManager` cada vez más optimizaciones de la batería en el sistema Android, los métodos del `AlarmManager` también han cambiado significativamente (para permitir un tiempo más indulgente). Sin embargo, para algunas aplicaciones todavía se requiere que sea lo más exacto posible en todas las versiones de Android. El siguiente asistente utiliza el método más preciso disponible en todas las plataformas para programar un `PendingIntent` :

```
public static void setExactAndAllowWhileIdle(AlarmManager alarmManager, int type, long
triggerAtMillis, PendingIntent operation) {
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M){
        alarmManager.setExactAndAllowWhileIdle(type, triggerAtMillis, operation);
    } else if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT){
        alarmManager.setExact(type, triggerAtMillis, operation);
    } else {
        alarmManager.set(type, triggerAtMillis, operation);
    }
}
```

El modo API23 + Doze interfiere con AlarmManager

Android 6 (API23) introdujo el modo Doze que interfiere con AlarmManager. Utiliza ciertas ventanas de mantenimiento para manejar las alarmas, por lo que incluso si usó `setExactAndAllowWhileIdle()` no puede asegurarse de que su alarma se active en el momento deseado.

Puede desactivar este comportamiento para su aplicación usando la configuración de su teléfono (`Settings/General/Battery & power saving/Battery usage/Ignore optimizations` o similares)

Dentro de tu aplicación puedes comprobar esta configuración ...

```
String packageName = getPackageName();
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
if (pm.isIgnoringBatteryOptimizations(packageName)) {
   // your app is ignoring Doze battery optimization
}
```

... y, finalmente, mostrar el cuadro de diálogo de configuración respectiva:

```
Intent intent = new Intent();
String packageName = getPackageName();
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
intent.setAction(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);
intent.setData(Uri.parse("package:" + packageName));
startActivity(intent);
```

Lea AlarmManager en línea: https://riptutorial.com/es/android/topic/1361/alarmmanager

Capítulo 12: Almacenamiento de archivos en almacenamiento interno y externo

Sintaxis

- FileOutputStream openFileInput (nombre de cadena)
- FileOutputStream openFileOutput (nombre de cadena, modo int)
- Archivo (Archivo dir, Nombre de la cadena)
- Archivo (ruta de la cadena)
- Archivo getExternalStoragePublicDirectory (tipo de cadena)
- Archivo getExternalFilesDir (tipo de cadena)

Parámetros

Parámetro	Detalles
nombre	El nombre del archivo a abrir. NOTA: No puede contener separadores de ruta
modo	Modo operativo. Use `MODE_PRIVATE` para la operación predeterminada, y `MODE_APPEND` para agregar un archivo existente. Otros modos incluyen `MODE_WORLD_READABLE` y `MODE_WORLD_WRITEABLE` , ambos en desuso en la API 17.
dir	Directorio del archivo para crear un nuevo archivo en.
camino	Ruta para especificar la ubicación del nuevo archivo
tipo	Tipo de directorio de archivos para recuperar. Puede ser `null` o alguno de los siguientes: `DIRECTORY_MUSIC` , `DIRECTORY_PODCASTS` , `DIRECTORY_RINGTONES` , `DIRECTORY_ALARMS` , `DIRECTORY_NOTIFICATIONS` , `DIRECTORY_PICTURES` o `DIRECTORY_MOVIES`

Examples

Uso de almacenamiento interno

De forma predeterminada, todos los archivos que guarde en Almacenamiento interno son privados para su aplicación. No se puede acceder a ellos por otras aplicaciones, ni al usuario en circunstancias normales. **Estos archivos se eliminan cuando el usuario desinstala la aplicación** .

Para escribir texto en un archivo

```
String fileName= "helloworld";
String textToWrite = "Hello, World!";
```

```
FileOutputStream fileOutputStream;

try {
  fileOutputStream = openFileOutput(fileName, Context.MODE_PRIVATE);
  fileOutputStream.write(textToWrite.getBytes());
  fileOutputStream.close();
} catch (Exception e) {
  e.printStackTrace();
}
```

Para agregar texto a un archivo existente

Use `Context.MODE_APPEND` para el parámetro de modo de `openFileOutput`

```
fileOutputStream = openFileOutput(fileName, Context.MODE_APPEND);
```

Uso de almacenamiento externo

El almacenamiento "externo" es otro tipo de almacenamiento que podemos usar para guardar archivos en el dispositivo del usuario. Tiene algunas diferencias clave con respecto al almacenamiento "interno", a saber:

- No siempre está disponible. En el caso de un medio extraíble (tarjeta SD), el usuario simplemente puede quitar el almacenamiento.
- No es privado. El usuario (y otras aplicaciones) tienen acceso a estos archivos.
- Si el usuario desinstala la aplicación, los archivos que guarde en el directorio recuperado con `getExternalFilesDir()` se eliminarán.

Para usar almacenamiento externo, primero debemos obtener los permisos adecuados. Necesitará usar:

- `android.permission.WRITE_EXTERNAL_STORAGE` para leer y escribir
- `android.permission.READ_EXTERNAL_STORAGE` para solo leer

Para otorgar estos permisos, deberá identificarlos en su `AndroidManifest.xml` como tal

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

> NOTA: ya que son permisos peligrosos si está utilizando el **nivel de API 23** o superior, deberá solicitar los **permisos en tiempo de ejecución** .

Antes de intentar escribir o leer desde el almacenamiento externo, siempre debe verificar que el medio de almacenamiento esté disponible.

```
String state = Environment.getExternalStorageState();
if (state.equals(Environment.MEDIA_MOUNTED)) {
    // Available to read and write
}
if (state.equals(Environment.MEDIA_MOUNTED) ||
        state.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
    // Available to at least read
```

```
}
```

Al escribir archivos en el almacenamiento externo, debe decidir si el archivo debe ser reconocido como Público o Privado. Si bien ambos tipos de archivos aún son accesibles para el usuario y otras aplicaciones en el dispositivo, existe una distinción clave entre ellos.

Los archivos públicos deben permanecer en el dispositivo cuando el usuario desinstala la aplicación. Un ejemplo de un archivo que debe guardarse como Público sería fotos tomadas a través de su aplicación.

Todos los archivos privados deben eliminarse cuando el usuario desinstala la aplicación. Estos tipos de archivos serían específicos de la aplicación y no serían de utilidad para el usuario u otras aplicaciones. Ex. Archivos temporales descargados / utilizados por su aplicación.

A continuación, se explica cómo obtener acceso al directorio `Documents` para archivos públicos y privados.

Público

```
// Access your app's directory in the device's Public documents directory
File docs = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_DOCUMENTS), "YourAppDirectory");
// Make the directory if it does not yet exist
myDocs.mkdirs();
```

Privado

```
// Access your app's Private documents directory
File file = new File(context.getExternalFilesDir(Environment.DIRECTORY_DOCUMENTS),
        "YourAppDirectory");
// Make the directory if it does not yet exist
myDocs.mkdirs();
```

Android: Almacenamiento interno y externo - Aclaración de terminología

Los desarrolladores de Android (principalmente principiantes) se han confundido con la terminología de almacenamiento interno y externo. Hay muchas preguntas sobre Stackoverflow sobre el mismo. Esto se debe principalmente al hecho de que la *terminología de* acuerdo con la documentación de Google / oficial de Android es bastante diferente a la de un usuario normal del sistema operativo Android. Por lo tanto, pensé que documentar esto ayudaría.

Lo que pensamos - Terminología del usuario (UT)

Almacenamiento interno (UT)	Almacenamiento externo (UT)
memoria interna incorporada del teléfono	Tarjeta Secure Digital (SD) extraíble o almacenamiento micro SD
Ejemplo: memoria interna	**Ejemplo:** espacio de almacenamiento en tarjetas SD

Almacenamiento interno (UT)	Almacenamiento externo (UT)
de 32 GB del Nexus 6P.	extraíbles proporcionadas por proveedores como Samsung, Sandisk, Strontium, Transcend y otros

Pero, de **acuerdo con la documentación / guía de Android - Terminología de Google (GT)**

Almacenamiento interno (GT):

> De forma predeterminada, los archivos guardados en el almacenamiento interno son privados para su aplicación y otras aplicaciones no pueden acceder a ellos (ni puede hacerlo el usuario).

Almacenamiento externo (GT):

> Puede ser un medio de almacenamiento extraíble (como una tarjeta SD) o un almacenamiento interno (no extraíble).

El almacenamiento externo (GT) se puede clasificar en dos tipos:

Almacenamiento externo primario	Almacenamiento externo secundario o almacenamiento extraíble (GT)
Esto es lo mismo que la memoria interna incorporada del teléfono (o) Almacenamiento interno (UT)	Esto es lo mismo que el almacenamiento extraíble de tarjeta micro SD (o) Almacenamiento externo (UT)
Ejemplo: memoria interna de 32 GB del Nexus 6P.	**Ejemplo:** espacio de almacenamiento en tarjetas SD extraíbles proporcionadas por proveedores como Samsung, Sandisk, Strontium, Transcend y otros
Se puede acceder a este tipo de almacenamiento en la PC con Windows conectando su teléfono a la PC mediante un cable USB y seleccionando *Cámara (PTP)* en la notificación de opciones de USB.	Se puede acceder a este tipo de almacenamiento en PC con Windows conectando su teléfono a PC mediante un cable USB y seleccionando *Transferencia de archivos* en la notificación de opciones de USB.

En una palabra,

Almacenamiento externo (GT) = Almacenamiento interno (UT) y Almacenamiento externo (UT)

Almacenamiento extraíble (GT) = Almacenamiento externo (UT)

Almacenamiento interno (GT) no tiene un término en UT.

Déjame explicarte claramente,

Almacenamiento interno (GT): de forma predeterminada, los archivos guardados en el almacenamiento interno son privados para su aplicación y otras aplicaciones no pueden acceder a ellos. El usuario de la aplicación tampoco puede acceder a ellos mediante el administrador de archivos; Incluso después de habilitar la opción "Mostrar archivos ocultos" en el administrador de archivos. Para acceder a los archivos en el almacenamiento interno (GT), debe rootear su teléfono Android. Además, cuando el usuario desinstala su aplicación, estos archivos se eliminan / eliminan.

Por lo tanto, el almacenamiento interno (GT) **NO** es lo que pensamos como la memoria interna de 32/64 GB de Nexus 6P

En general, **la ubicación del almacenamiento interno (GT)** sería:
`/data/data/your.application.package.appname/someDirectory/`

Almacenamiento externo (GT):

> Todos los dispositivos compatibles con Android admiten un "almacenamiento externo" compartido que puede usar para guardar archivos. Los archivos guardados en el almacenamiento externo son legibles en todo el mundo y pueden ser modificados por el usuario cuando permiten que el almacenamiento masivo USB transfiera archivos a una computadora.

Ubicación de almacenamiento externo (GT): podría estar en *cualquier lugar* en su almacenamiento interno (UT) o en su almacenamiento extraíble (GT), es decir, en una tarjeta micro SD. Depende del OEM de su teléfono y también de la versión del sistema operativo Android.

Para leer o escribir archivos en el almacenamiento externo (GT), su aplicación debe adquirir los permisos del sistema `READ_EXTERNAL_STORAGE` o `WRITE_EXTERNAL_STORAGE` .

Por ejemplo:

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

> Si necesita leer y escribir archivos, debe solicitar solo el permiso `WRITE_EXTERNAL_STORAGE` , ya que implícitamente también requiere acceso de lectura.

En **Almacenamiento externo (GT)** , también puede guardar archivos **privados de la aplicación.**

Pero,

> Cuando el usuario desinstala su aplicación, este directorio y todo su contenido se eliminan.

¿Cuándo necesita guardar los archivos que son **privados de la aplicación** en el

almacenamiento externo (GT) ?

> Si está manejando archivos que no están destinados a otras aplicaciones (como texturas gráficas o efectos de sonido utilizados solo por su aplicación), debe usar un directorio de almacenamiento privado en el almacenamiento externo
>
> A partir de Android 4.4, leer o escribir archivos en los directorios privados de su aplicación no requiere los `READ_EXTERNAL_STORAGE` o `WRITE_EXTERNAL_STORAGE` . Por lo tanto, puede declarar que el permiso debe solicitarse solo en las versiones inferiores de Android agregando el atributo `maxSdkVersion` :

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
                     android:maxSdkVersion="18" />
    ...
</manifest
```

Métodos para almacenar en almacenamiento interno (GT):

Ambos métodos están presentes en la clase de contexto

```
File getDir (String name, int mode)

File getFilesDir ()
```

Métodos para almacenar en almacenamiento externo primario, es decir, almacenamiento interno (UT):

```
File getExternalStorageDirectory ()

File getExternalFilesDir (String type)

File getExternalStoragePublicDirectory (String type)
```

Al principio, todos usaban Environment.getExternalStorageDirectory () , que apuntaba a la **raíz** del **almacenamiento externo primario** . Como resultado, el almacenamiento externo primario se llenó con contenido aleatorio.

Posteriormente, se agregaron estos dos métodos:

1. En la clase de `Context` , agregaron getExternalFilesDir () , apuntando a un **directorio específico de la aplicación** en el almacenamiento externo primario. Este directorio y su contenido **se eliminarán** cuando la aplicación se desinstale.

2. Environment.getExternalStoragePublicDirectory () para lugares centralizados para almacenar tipos de archivos conocidos, como fotos y películas. Este directorio y su contenido **NO se eliminarán** cuando la aplicación se desinstale.

Métodos para almacenar en almacenamiento extraíble (GT), es decir, tarjeta micro SD

Antes del **nivel de API 19** , no **había forma oficial** de almacenar en la tarjeta SD. Pero, muchos

podrían hacerlo utilizando bibliotecas o API no oficiales.

Oficialmente, se introdujo un método en la clase de `Context` en el nivel de API 19 (versión 4.4 de Android - Kitkat).

```
File[] getExternalFilesDirs (String type)
```

> Devuelve las rutas absolutas a los directorios específicos de la aplicación en todos los dispositivos de almacenamiento compartidos / externos donde la aplicación puede colocar los archivos persistentes que posee. Estos archivos son internos a la aplicación y no suelen ser visibles para el usuario como medio.

Eso significa que devolverá las rutas a **ambos** tipos de almacenamiento externo (GT): memoria interna y tarjeta Micro SD. En general, la **segunda ruta** sería la ruta de almacenamiento de la tarjeta micro SD (pero no siempre). Así que necesitas comprobarlo ejecutando el código con este método.

Ejemplo con fragmento de código:

Creé un nuevo proyecto de Android con actividad vacía, escribí el siguiente código dentro de

`protected void onCreate(Bundle savedInstanceState)` método de `MainActivity.java`

```
    File internal_m1 = getDir("custom", 0);
    File internal_m2 = getFilesDir();

    File external_m1 =  Environment.getExternalStorageDirectory();

    File external_m2 =  getExternalFilesDir(null);
    File external_m2_Args = getExternalFilesDir(Environment.DIRECTORY_PICTURES);

    File external_m3 =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);

    File[] external_AND_removable_storage_m1 = getExternalFilesDirs(null);
    File[] external_AND_removable_storage_m1_Args =
getExternalFilesDirs(Environment.DIRECTORY_PICTURES);
```

Después de ejecutar el código anterior,

Salida:

```
internal_m1: /data/data/your.application.package.appname/app_custom

internal_m2: /data/data/your.application.package.appname/files

external_m1: /storage/emulated/0

external_m2: /storage/emulated/0/Android/data/your.application.package.appname/files

external_m2_Args:
/storage/emulated/0/Android/data/your.application.package.appname/files/Pictures

external_m3: /storage/emulated/0/Pictures
```

```
external_AND_removable_storage_m1 (first path):
/storage/emulated/0/Android/data/your.application.package.appname/files

external_AND_removable_storage_m1 (second path):
/storage/sdcard1/Android/data/your.application.package.appname/files

 external_AND_removable_storage_m1_Args (first path):
/storage/emulated/0/Android/data/your.application.package.appname/files/Pictures

external_AND_removable_storage_m1_Args (second path):
/storage/sdcard1/Android/data/your.application.package.appname/files/Pictures
```

Nota: He conectado mi teléfono a la PC con Windows; habilitó ambas opciones de desarrollador, depuración USB y luego ejecutó este código. Si **no conectas tu teléfono** ; pero en lugar de ejecutar esto en el **emulador de Android** , su salida puede variar. Mi modelo de teléfono es Coolpad Note 3, se ejecuta en Android 5.1

Ubicaciones de almacenamiento en mi teléfono:

Ubicación de almacenamiento Micro SD : `/storage/sdcard1`

Ubicación del almacenamiento interno (UT) : `/storage/sdcard0` .

Tenga en cuenta que `/sdcard` & `/storage/emulated/0` también apunta al almacenamiento interno (UT). Pero estos son enlaces simbólicos a `/storage/sdcard0` .

Para entender claramente las diferentes rutas de almacenamiento en Android, por favor, vaya a través de esta respuesta

Descargo de responsabilidad: todas las rutas de almacenamiento mencionadas anteriormente son rutas en **mi** teléfono. Es posible que sus archivos **no** se almacenen en las mismas rutas de almacenamiento. Debido a que las ubicaciones / rutas de almacenamiento pueden variar en otros teléfonos móviles dependiendo de su proveedor, fabricante y diferentes versiones del sistema operativo Android.

Guardar base de datos en la tarjeta SD (Copia de seguridad de base de datos en SD)

```
    public static Boolean ExportDB(String DATABASE_NAME , String packageName , String
folderName){
    //DATABASE_NAME including ".db" at the end like "mayApp.db"
    String DBName = DATABASE_NAME.substring(0, DATABASE_NAME.length() - 3);
    File data = Environment.getDataDirectory();
    FileChannel source=null;
    FileChannel destination=null;
    String currentDBPath = "/data/"+ packageName +"/databases/"+DATABASE_NAME; // getting app
db path

    File sd = Environment.getExternalStorageDirectory(); // getting phone SD card path
    String backupPath = sd.getAbsolutePath() + folderName; // if you want to set backup in
specific folder name
        /* be careful , foldername must initial like this : "/myFolder" . dont forget "/" at
begin of folder name
```

```
            you could define foldername like this : "/myOutterFolder/MyInnerFolder" and so on
...
        */
    File dir = new File(backupPath);
    if(!dir.exists()) // if there was no folder at this path , it create it .
    {
        dir.mkdirs();
    }

    DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss");
    Date date = new Date();
        /* use date including file name for arrange them and preventing to make file with the
same*/
    File currentDB = new File(data, currentDBPath);
    File backupDB = new File(backupPath, DBName +"("+ dateFormat.format(date)+").db");
    try {
        if (currentDB.exists() && !backupDB.exists()) {
            source = new FileInputStream(currentDB).getChannel();
            destination = new FileOutputStream(backupDB).getChannel();
            destination.transferFrom(source, 0, source.size());
            source.close();
            destination.close();
            return true;
        }
        return false;
    } catch(IOException e) {
        e.printStackTrace();
        return false;
    }
}
```

llame a este método de esta manera:

ExportDB ("myDB.db", "com.example.exam", "/ myFolder");

Fetch Directorio de dispositivos:

Primero agregue el permiso de almacenamiento para leer / buscar el directorio del dispositivo.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Crear clase de modelo

```
//create one directory model class
//to store directory title and type in list

public class DirectoryModel {
    String dirName;
    int dirType; // set 1 or 0, where 0 for directory and 1 for file.

    public int getDirType() {
        return dirType;
    }

    public void setDirType(int dirType) {
```

```
        this.dirType = dirType;
    }

    public String getDirName() {
        return dirName;
    }

    public void setDirName(String dirName) {
        this.dirName = dirName;
    }
}
```

Crear lista utilizando el modelo de directorio para agregar datos de directorio.

```
//define list to show directory

List<DirectoryModel> rootDir = new ArrayList<>();
```

Fetch directorio utilizando el siguiente método.

```
//to fetch device directory

private void getDirectory(String currDir) { // pass device root directory
        File f = new File(currDir);
        File[] files = f.listFiles();
        if (files != null) {
            if (files.length > 0) {
                rootDir.clear();
                for (File inFile : files) {
                    if (inFile.isDirectory()) { //return true if it's directory
                        // is directory
                        DirectoryModel dir = new DirectoryModel();
                        dir.setDirName(inFile.toString().replace("/storage/emulated/0", ""));
                        dir.setDirType(0); // set 0 for directory
                        rootDir.add(dir);
                    } else if (inFile.isFile()) { // return true if it's file
                        //is file
                        DirectoryModel dir = new DirectoryModel();
                        dir.setDirName(inFile.toString().replace("/storage/emulated/0", ""));
                        dir.setDirType(1); // set 1 for file
                        rootDir.add(dir);
                    }
                }
            }
            printDirectoryList();
        }
    }
```

Imprimir lista de directorios en el registro.

```
//print directory list in logs

private void printDirectoryList() {
    for (int i = 0; i < rootDir.size(); i++) {
        Log.e(TAG, "printDirectoryLogs: " + rootDir.get(i).toString());
    }
}
```

Uso

```
//to Fetch Directory Call function with root directory.

String rootPath = Environment.getExternalStorageDirectory().toString(); // return ==>
/storage/emulated/0/
getDirectory(rootPath );
```

Para recuperar archivos / carpetas internos de un directorio específico, use el mismo método, solo cambie el argumento, pase la ruta actual seleccionada en el argumento y maneje la respuesta para el mismo.

Para obtener la extensión de archivo:

```
private String getExtension(String filename) {

    String filenameArray[] = filename.split("\\.");
    String extension = filenameArray[filenameArray.length - 1];
    Log.d(TAG, "getExtension: " + extension);

    return extension;
}
```

Lea Almacenamiento de archivos en almacenamiento interno y externo en línea: https://riptutorial.com/es/android/topic/150/almacenamiento-de-archivos-en-almacenamiento-interno-y-externo

Capítulo 13: Añadiendo un FuseView a un proyecto de Android

Introducción

Exporte un Fuse.View desde fusetools y utilícelo dentro de un proyecto de Android existente.

Nuestro objetivo es exportar toda la aplicación de ejemplo de hikr y usarla dentro de una `Activity` .

El trabajo final se puede encontrar en lucamtudor / hikr-fuse-view

Examples

aplicación hikr, solo otro android.view.View

Prerrequisitos

- debe tener el fusible instalado (https://www.fusetools.com/downloads)
- deberías haber hecho el tutorial de introducción
- en terminal: `fuse install android`
- en terminal: `uno install Fuse.Views`

Paso 1

```
git clone https://github.com/fusetools/hikr
```

Paso 2 : Agregue la referencia del paquete a `Fuse.Views`

Encuentre el archivo `hikr.unoproj` dentro de la carpeta raíz del proyecto y agregue `"Fuse.Views"` a la matriz de `"Packages"` .

```
{
  "RootNamespace":"",
  "Packages": [
    "Fuse",
    "FuseJS",
    "Fuse.Views"
  ],
  "Includes": [
    "*",
    "Modules/*.js:Bundle"
  ]
}
```

Paso 3 : Haz que el componente `HikrApp` mantenga la aplicación completa

3.1 En la carpeta raíz del proyecto, `HikrApp.ux` un nuevo archivo llamado `HikrApp.ux` y pegue el contenido de `MainView.ux` .

HikrApp.ux

```
<App Background="#022328">
    <iOS.StatusBarConfig Style="Light" />
    <Android.StatusBarConfig Color="#022328" />

    <Router ux:Name="router" />

    <ClientPanel>
        <Navigator DefaultPath="splash">
            <SplashPage ux:Template="splash" router="router" />
            <HomePage ux:Template="home" router="router" />
            <EditHikePage ux:Template="editHike" router="router" />
        </Navigator>
    </ClientPanel>
</App>
```

3.2 En `HikrApp.ux`

- Reemplace las etiquetas `<App>` con `<Page>`
- agregue `ux:Class="HikrApp"` a la página inicial `<Page>`
- eliminar `<ClientPanel>` , ya no tenemos que preocuparnos por la barra de estado o los botones de navegación inferiores

HikrApp.ux

```
<Page ux:Class="HikrApp" Background="#022328">
    <iOS.StatusBarConfig Style="Light" />
    <Android.StatusBarConfig Color="#022328" />

    <Router ux:Name="router" />

    <Navigator DefaultPath="splash">
        <SplashPage ux:Template="splash" router="router" />
        <HomePage ux:Template="home" router="router" />
        <EditHikePage ux:Template="editHike" router="router" />
    </Navigator>
</Page>
```

3.3 Usar el componente `HikrApp` recién creado dentro de `MainView.ux`

Reemplace el contenido del archivo `MainView.ux` con:

```
<App>
    <HikrApp/>
</App>
```

Nuestra aplicación ha vuelto a su comportamiento normal, pero ahora la hemos extraído a un componente separado llamado `HikrApp`

Paso 4 Dentro de `MainView.ux` reemplace las etiquetas `<App>` con `<ExportedViews>` y agregue `ux:Template="HikrAppView"` a `<HikrApp />`

```
<ExportedViews>
    <HikrApp ux:Template="HikrAppView" />
</ExportedViews>
```

Recuerde la plantilla `HikrAppView` , porque la necesitaremos para obtener una referencia a nuestra vista desde Java.

Nota De la documentación del fusible:

> `ExportedViews` se comportará como una `App` cuando realice una `fuse preview` normal y `uno build`

No es verdad. Obtendrá este error al obtener una vista previa de Fuse Studio:

> Error: no se pudo encontrar una etiqueta de aplicación en ninguno de los archivos UX incluidos. ¿Has olvidado incluir el archivo UX que contiene la etiqueta de la aplicación?

Paso 5 Wrap `SplashPage.ux` 's `<DockPanel>` en un `<GraphicsView>`

```
<Page ux:Class="SplashPage">
    <Router ux:Dependency="router" />

    <JavaScript File="SplashPage.js" />

    <GraphicsView>
        <DockPanel ClipToBounds="true">
            <Video Layer="Background" File="../Assets/nature.mp4" IsLooping="true"
AutoPlay="true" StretchMode="UniformToFill" Opacity="0.5">
            <Blur Radius="4.75" />
        </Video>

            <hikr.Text Dock="Bottom" Margin="10" Opacity=".5" TextAlignment="Center"
FontSize="12">original video by Graham Uhelski</hikr.Text>

            <Grid RowCount="2">
                <StackPanel Alignment="VerticalCenter">
                    <hikr.Text Alignment="HorizontalCenter" FontSize="70">hikr</hikr.Text>
                    <hikr.Text Alignment="HorizontalCenter" Opacity=".5">get out
there</hikr.Text>
                </StackPanel>

                <hikr.Button Text="Get Started" FontSize="18" Margin="50,0"
Alignment="VerticalCenter" Clicked="{goToHomePage}" />
            </Grid>
        </DockPanel>
    </GraphicsView>
```

```
</Page>
```

Paso 6 Exportar el proyecto de fusible como una biblioteca aar

- en terminal, en carpeta de proyecto raíz: `uno clean`
- en la terminal, en la carpeta del proyecto raíz: `uno build -t=android -DLIBRARY`

Paso 7 Prepare su proyecto de Android

- copie el archivo aar de `.../rootHikeProject/build/Android/Debug/app/build/outputs/aar/app-debug.aar` a `.../androidRootProject/app/libs`
- agregue `flatDir { dirs 'libs' }` al archivo root `build.gradle`

```
// Top-level build file where you can add configuration options common to all sub-
projects/modules.

buildscript { ... }

...

allprojects {
    repositories {
        jcenter()
        flatDir {
            dirs 'libs'
        }
    }
}

...
```

- agregar `compile(name: 'app-debug', ext: 'aar')` a las dependencias en `app/build.gradle`

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "com.shiftstudio.fuseviewtest"
        minSdkVersion 16
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
```

```
}

dependencies {
    compile(name: 'app-debug', ext: 'aar')
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.3.1'
    testCompile 'junit:junit:4.12'
}
```

- agrega las siguientes propiedades a la actividad dentro de `AndroidManifest.xml`

```
android:launchMode="singleTask"
android:taskAffinity=""
android:configChanges="orientation|keyboardHidden|screenSize|smallestScreenSize"
```

Tu `AndroidManifest.xml` se verá así:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.shiftstudio.fuseviewtest">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:launchMode="singleTask"
            android:taskAffinity=""
            android:configChanges="orientation|keyboardHidden|screenSize|smallestScreenSize">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Paso 8 : Muestre el `Fuse.View HikrAppView` en su `Activity`

- tenga en cuenta que su `Activity` necesita heredar `FuseViewsActivity`

```
public class MainActivity extends FuseViewsActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);

        final ViewHandle fuseHandle = ExportedViews.instantiate("HikrAppView");

        final FrameLayout root = (FrameLayout) findViewById(R.id.fuse_root);
        final View fuseApp = fuseHandle.getView();
        root.addView(fuseApp);
    }
}
```

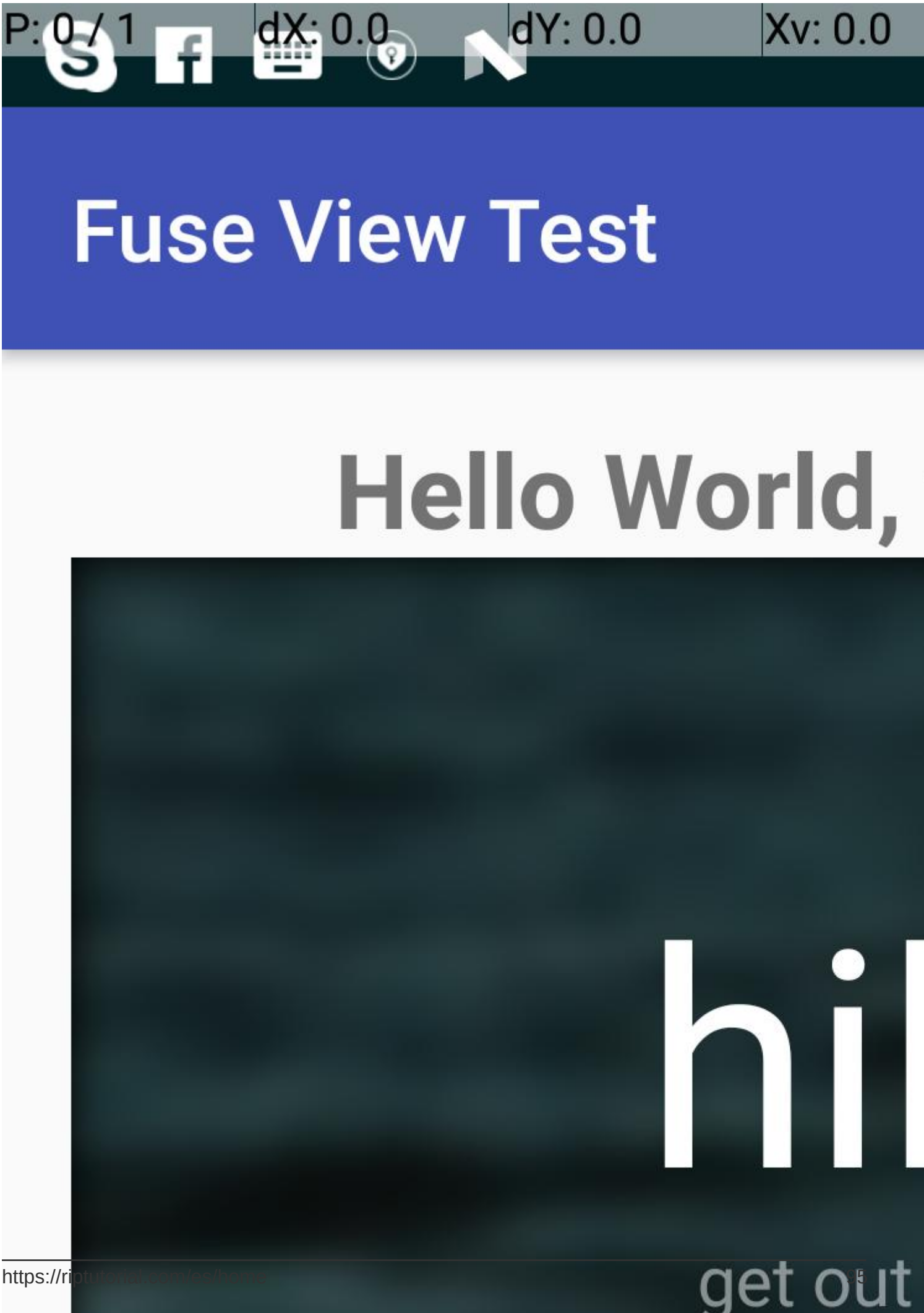
activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.shiftstudio.fuseviewtest.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_gravity="center_horizontal"
        android:textSize="24sp"
        android:textStyle="bold"
        android:layout_height="wrap_content"
        android:text="Hello World, from Kotlin" />

    <FrameLayout
        android:id="@+id/fuse_root"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:layout_width="wrap_content"
            android:text="THIS IS FROM NATIVE.\nBEHIND FUSE VIEW"
            android:layout_gravity="center"
            android:textStyle="bold"
            android:textSize="30sp"
            android:background="@color/colorAccent"
            android:textAlignment="center"
            android:layout_height="wrap_content" />

    </FrameLayout>

</LinearLayout>
```

Nota
Cuando presionas el botón Atrás, en Android, la aplicación falla. Puedes seguir el tema en el foro de fusibles .

```
A/libc: Fatal signal 11 (SIGSEGV), code 1, fault addr 0xdeadcab1 in tid 18026
(io.fuseviewtest)

        [ 05-25 11:52:33.658 16567:16567 W/ ]

        debuggerd: handling request: pid=18026 uid=10236 gid=10236 tid=18026
```

Y el resultado final es algo como esto. También puedes encontrar un clip corto en github .


P: 0 / 1
dX: 0.0
dY: 0.0
Xv: 0.0
Fuse View Test
Hello World,
hi
get out

https://riptutorial.com/es/android/topic/10052/anadiendo-un-fuseview-a-un-proyecto-de-android

Capítulo 14: Android NDK

Examples

Construyendo ejecutables nativos para Android

proyecto / jni / main.c

```
#include <stdio.h>
#include <unistd.h>

int main(void) {
  printf("Hello world!\n");
  return 0;
}
```

proyecto / jni / Android.mk

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)
LOCAL_MODULE    := hello_world
LOCAL_SRC_FILES := main.c
include $(BUILD_EXECUTABLE)
```

proyecto / jni / Application.mk

```
APP_ABI := all
APP_PLATFORM := android-21
```

Si desea admitir dispositivos con versiones de Android inferiores a 5.0 (API 21), debe compilar su binario con `APP_PLATFORM` establecido en una API más antigua, por ejemplo, `android-8` . Esto es una consecuencia de la aplicación de *binarios independientes de posición* (PIE) de Android 5.0, mientras que los dispositivos más antiguos no necesariamente admiten PIE. Por lo tanto, debe utilizar el PIE o el no PIE, dependiendo de la versión del dispositivo. Si desea utilizar el binario desde su aplicación de Android, debe verificar el nivel de API y extraer el binario correcto.

`APP_ABI` se puede cambiar a plataformas específicas como `armeabi` para construir el binario solo para esas arquitecturas.

En el peor de los casos, tendrá un binario PIE y otro no binario para cada arquitectura (aproximadamente 14 binarios diferentes que usan ndk-r10e).

Para construir el ejecutable:

```
cd project
ndk-build
```

Encontrará los binarios en `project/libs/<architecture>/hello_world` . Puede usarlos a través de `ADB` (`push` y `chmod` it con permiso ejecutable) o desde su aplicación (extraer y `chmod` it con permiso ejecutable).

Para determinar la arquitectura de la CPU, recupere la propiedad de construcción `ro.product.cpu.abi` para la arquitectura primaria o `ro.product.cpu.abilist` (en dispositivos más nuevos) para obtener una lista completa de las arquitecturas compatibles. Puede hacerlo usando la clase `android.os.Build` desde su aplicación o usando `getprop <name>` través de ADB.

Cómo limpiar la construcción

Si necesitas limpiar la construcción:

```
ndk-build clean
```

Cómo usar un makefile que no sea Android.mk

ndk-build NDK_PROJECT_PATH = PROJECT_PATH APP_BUILD_SCRIPT = MyAndroid.mk

Cómo iniciar sesión en ndk

Primero asegúrese de enlazar con la biblioteca de registro en su archivo `Android.mk` :

```
LOCAL_LDLIBS := -llog
```

Luego use una de las siguientes llamadas `__android_log_print()` :

```
#include <android/log.h>
#define TAG "MY LOG"

__android_log_print(ANDROID_LOG_VERBOSE,    TAG, "The value of 1 + 1 is %d", 1 + 1)
__android_log_print(ANDROID_LOG_WARN,       TAG, "The value of 1 + 1 is %d", 1 + 1)
__android_log_print(ANDROID_LOG_DEBUG,      TAG, "The value of 1 + 1 is %d", 1 + 1)
__android_log_print(ANDROID_LOG_INFO,       TAG, "The value of 1 + 1 is %d", 1 + 1)
__android_log_print(ANDROID_LOG_ERROR,      TAG, "The value of 1 + 1 is %d", 1 + 1)
```

O utilícelos de una manera más conveniente definiendo las macros correspondientes:

```
#define  LOGV(...)  __android_log_print(ANDROID_LOG_VERBOSE,    TAG, __VA_ARGS__)
#define  LOGW(...)  __android_log_print(ANDROID_LOG_WARN,       TAG, __VA_ARGS__)
#define  LOGD(...)  __android_log_print(ANDROID_LOG_DEBUG,      TAG, __VA_ARGS__)
#define  LOGI(...)  __android_log_print(ANDROID_LOG_INFO,       TAG, __VA_ARGS__)
#define  LOGE(...)  __android_log_print(ANDROID_LOG_ERROR,      TAG, __VA_ARGS__)
```

Ejemplo :

```
int x = 42;
LOGD("The value of x is %d", x);
```

Lea Android NDK en línea: https://riptutorial.com/es/android/topic/492/android-ndk

Capítulo 15: Android Studio

Examples

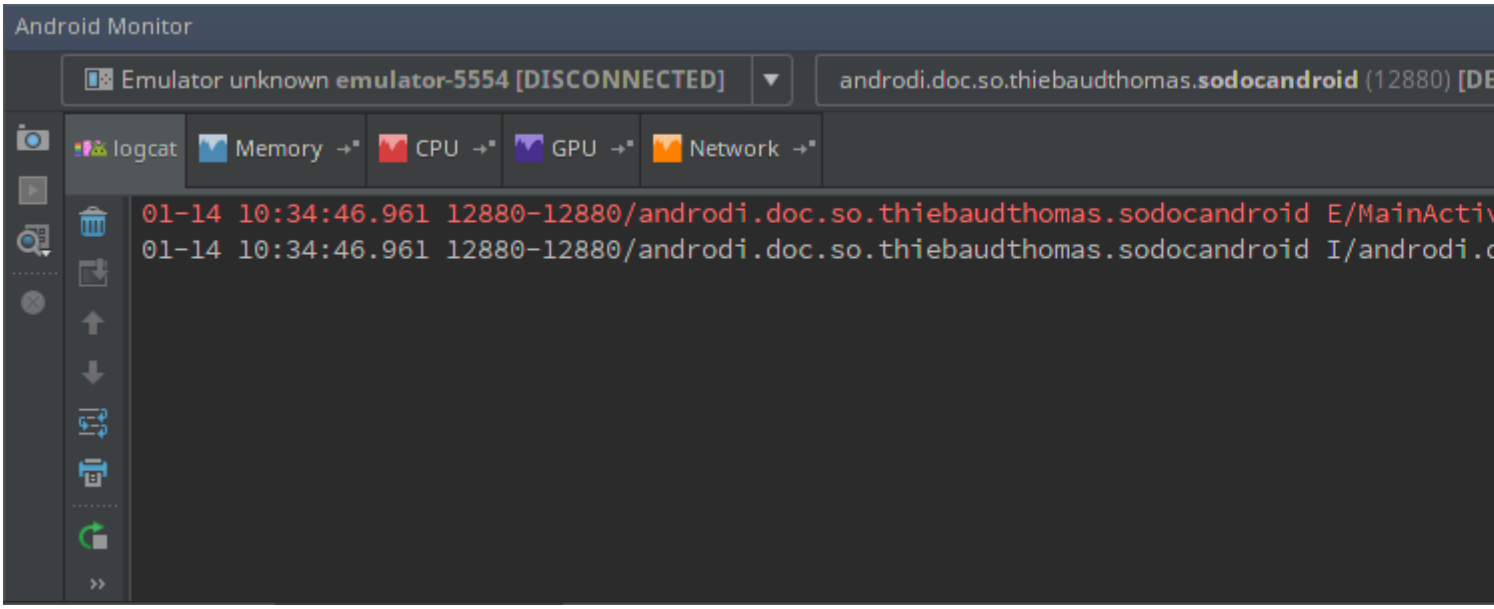
Filtrar los registros de la interfaz de usuario

Los registros de Android se pueden filtrar directamente desde la interfaz de usuario. Usando este codigo

```
public class MainActivity extends AppCompatActivity {
    private final static String TAG1 = MainActivity.class.getSimpleName();
    private final static String TAG2 = MainActivity.class.getCanonicalName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.e(TAG1,"Log from onCreate method with TAG1");
        Log.i(TAG2,"Log from onCreate method with TAG2");
    }
}
```

Si uso la expresión regular `TAG1|TAG2` y el nivel `verbose` que obtengo

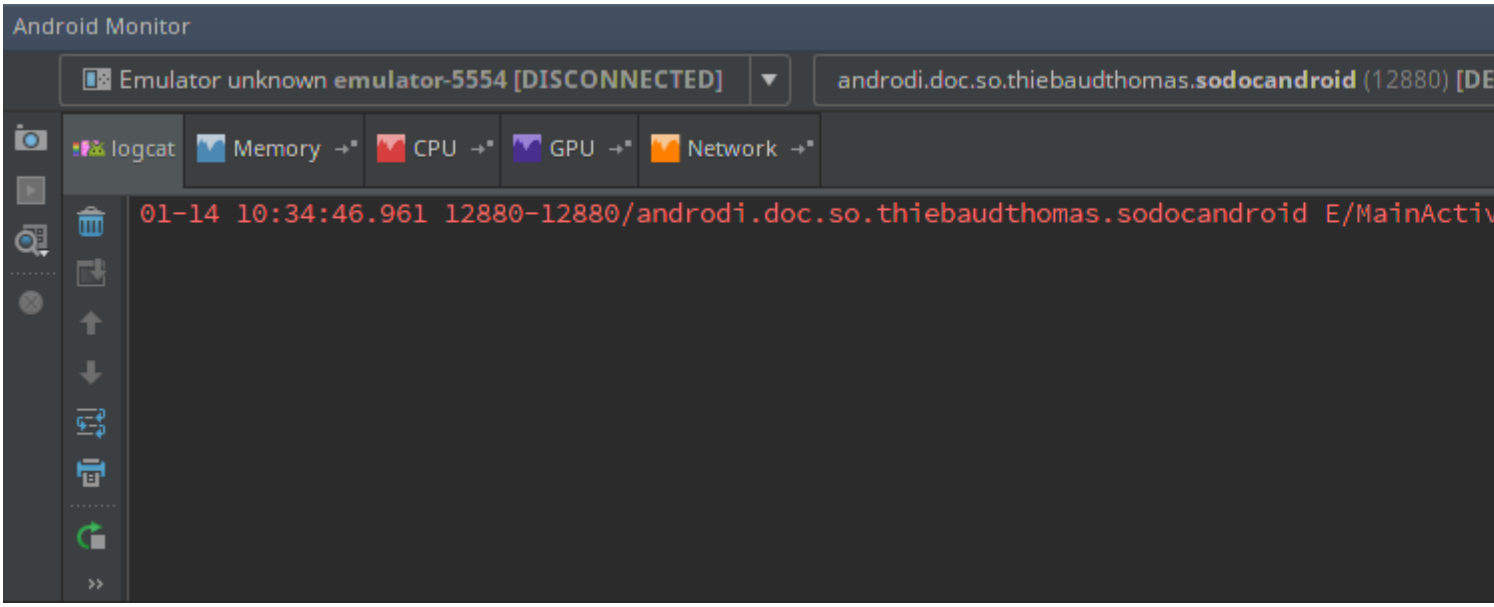
```
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log
from onCreate method with TAG1
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid
I/androdi.doc.so.thiebaudthomas.sodocandroid.MainActivity: Log from onCreate method with TAG2
```



El nivel se puede configurar para obtener registros con un nivel dado y superior. Por ejemplo, el nivel `verbose` capturará los registros `verbose, debug, info, warn, error and assert` .

Usando el mismo ejemplo, si configuro el nivel en `error` , solo obtengo

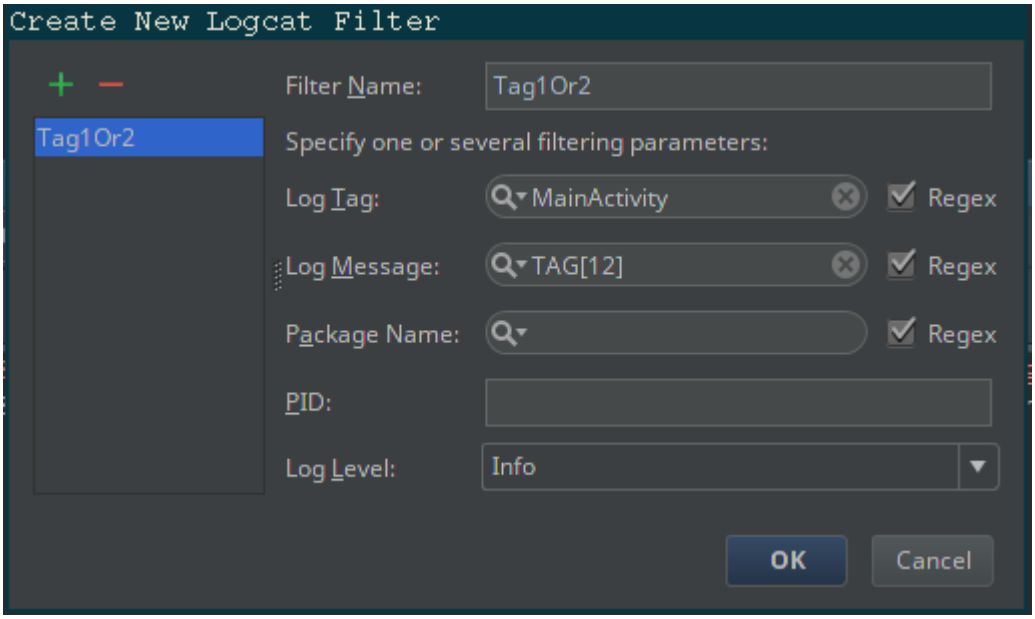
```
01-14 10:34:46.961 12880-12880/androdi.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log from onCreate method with TAG1
```



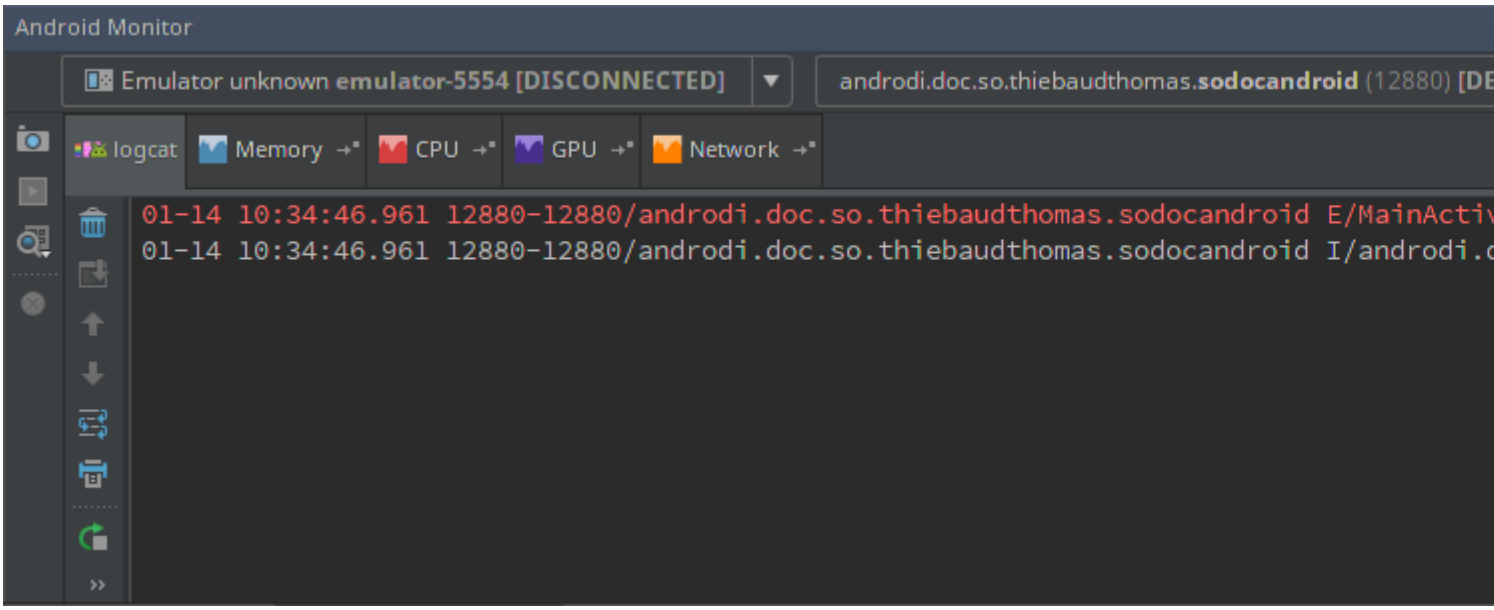
Crear configuración de filtros

Los filtros personalizados se pueden configurar y guardar desde la interfaz de usuario. En la pestaña `AndroidMonitor` , haga clic en el menú desplegable de la derecha (debe contener `Show only selected application` o `No filters`) y seleccione `Edit filter configuration` .

Ingrese el filtro que desee

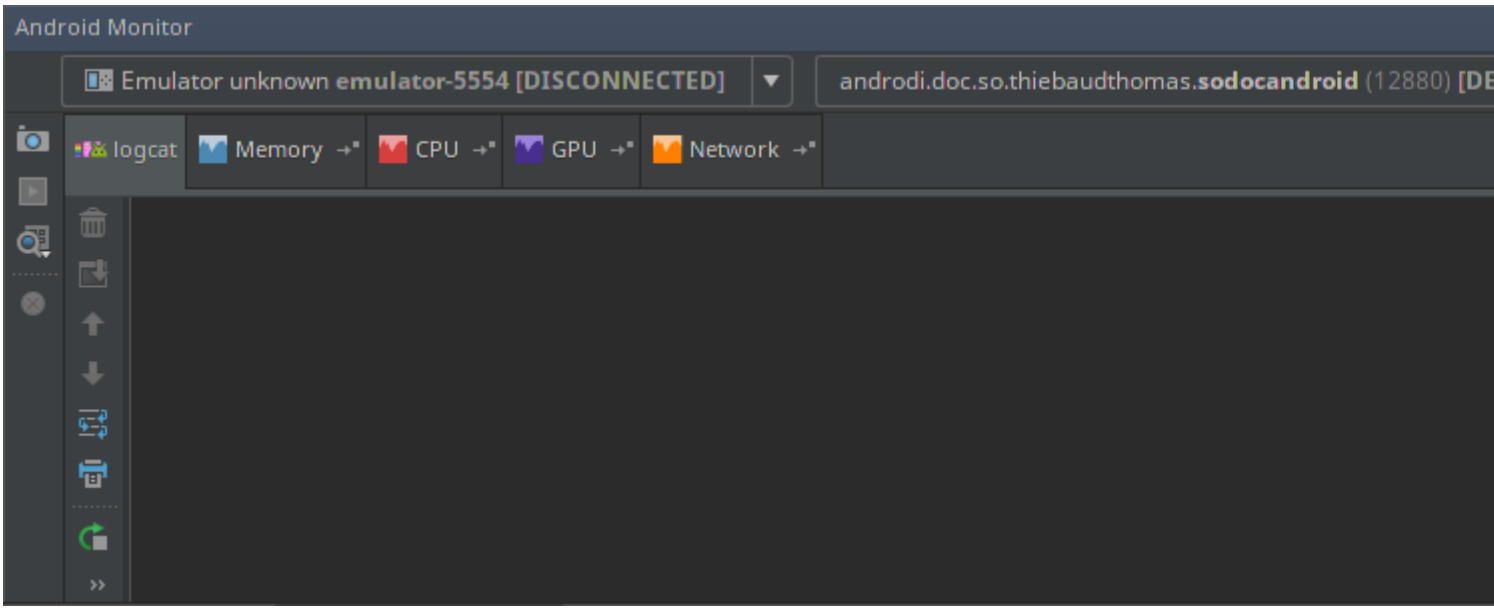


Y utilízalo (puedes seleccionarlo desde el mismo desplegable)

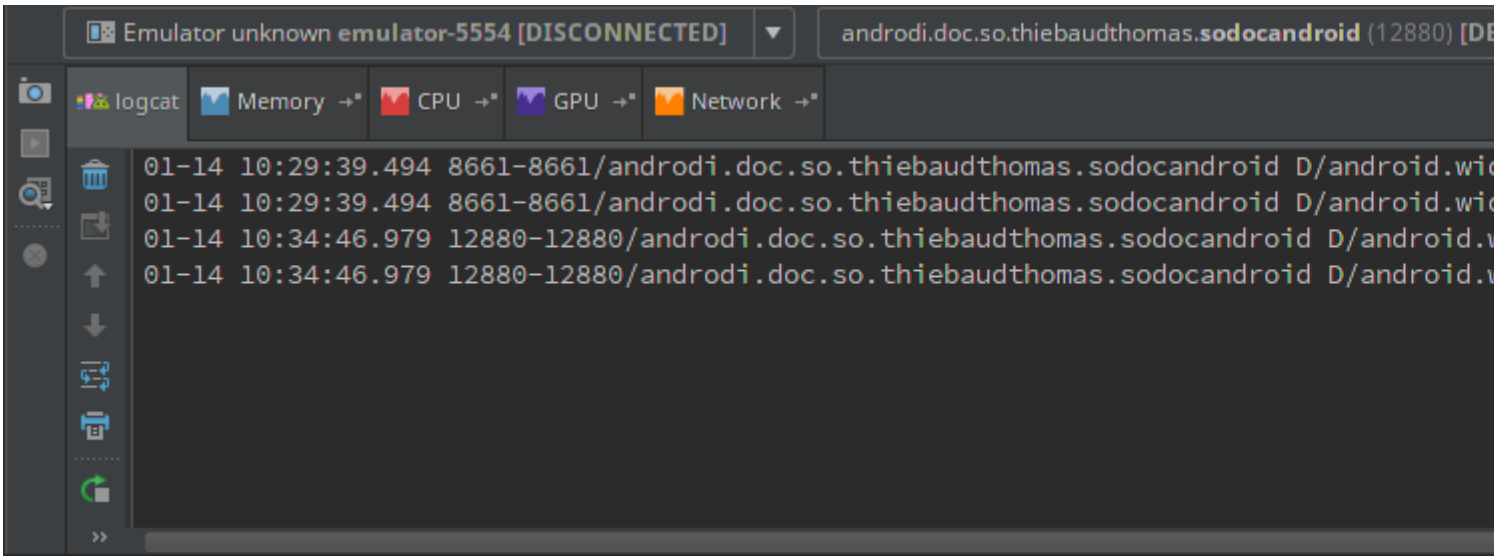


Importante Si agrega una entrada en la barra de filtros, Android Studio considerará tanto su filtro como su entrada.

Con entrada y filtro no hay salida



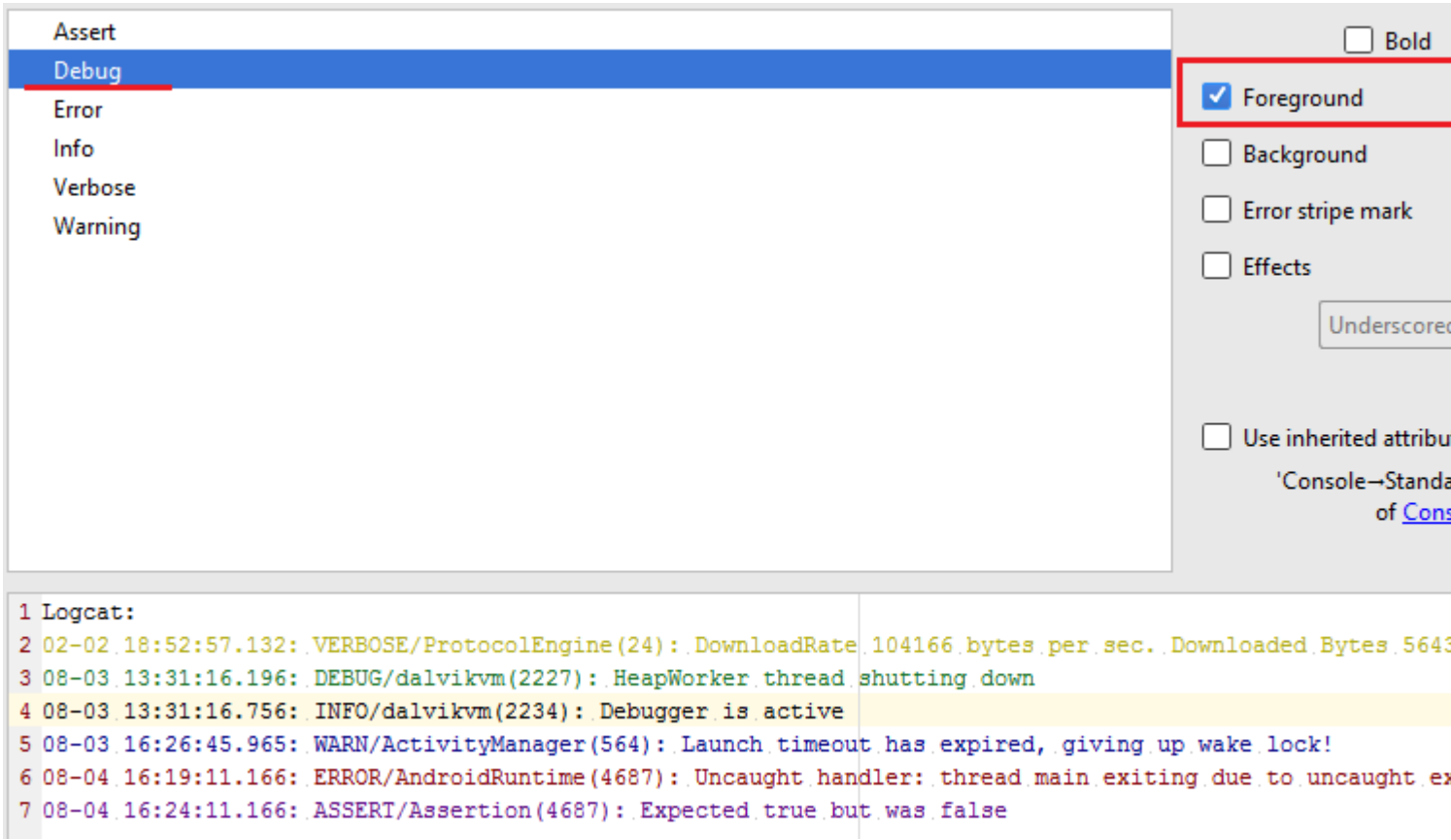
Sin filtro, hay algunas salidas.



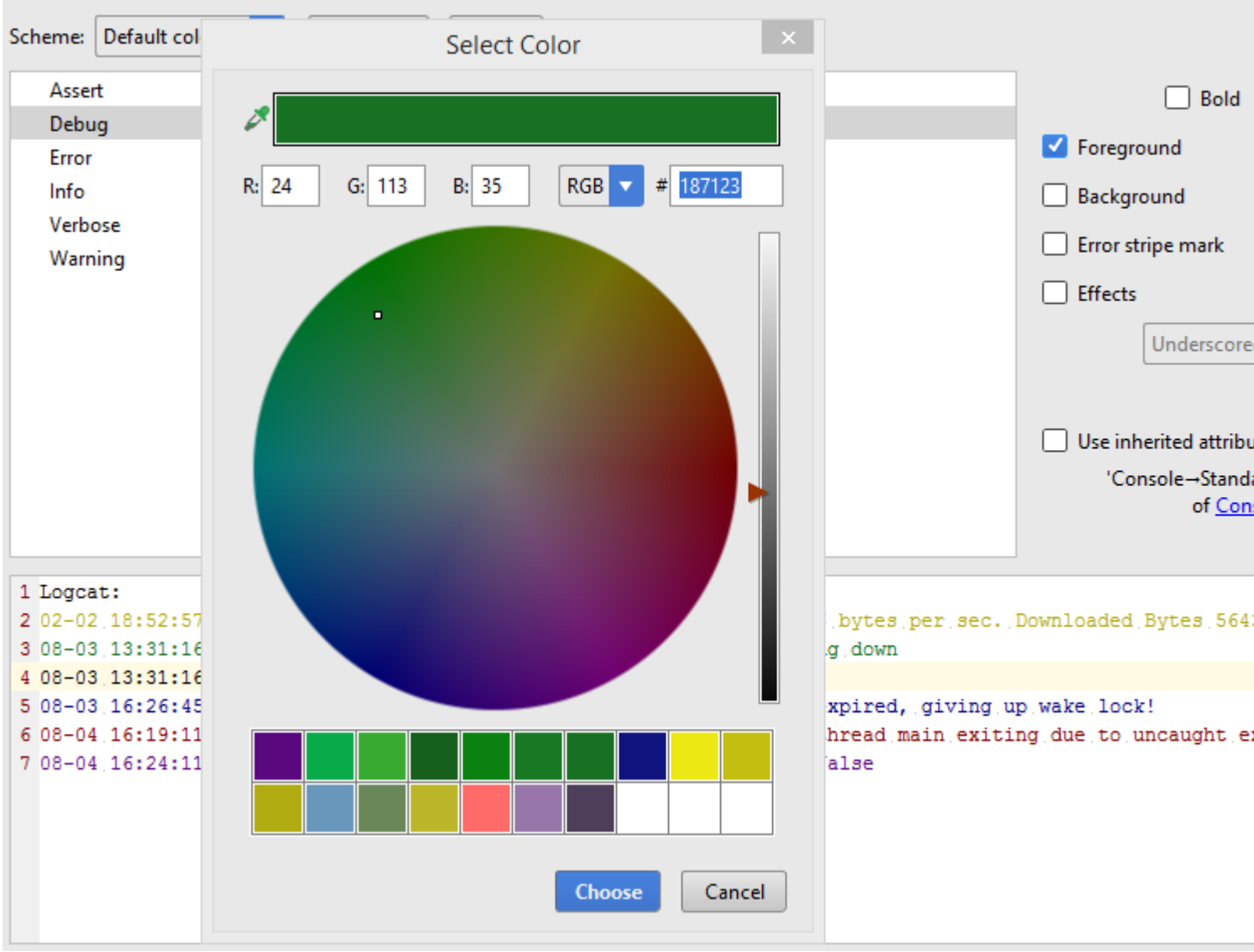
Colores personalizados del mensaje logcat basado en la importancia del mensaje

Vaya a Archivo -> Configuración -> Editor -> Colores y fuentes -> Logcat de Android

Cambia los colores que necesites:

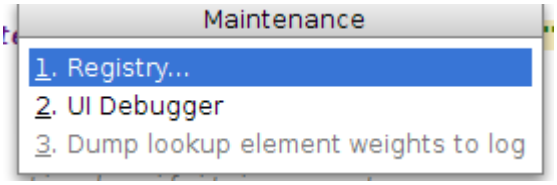


Elija el color apropiado:

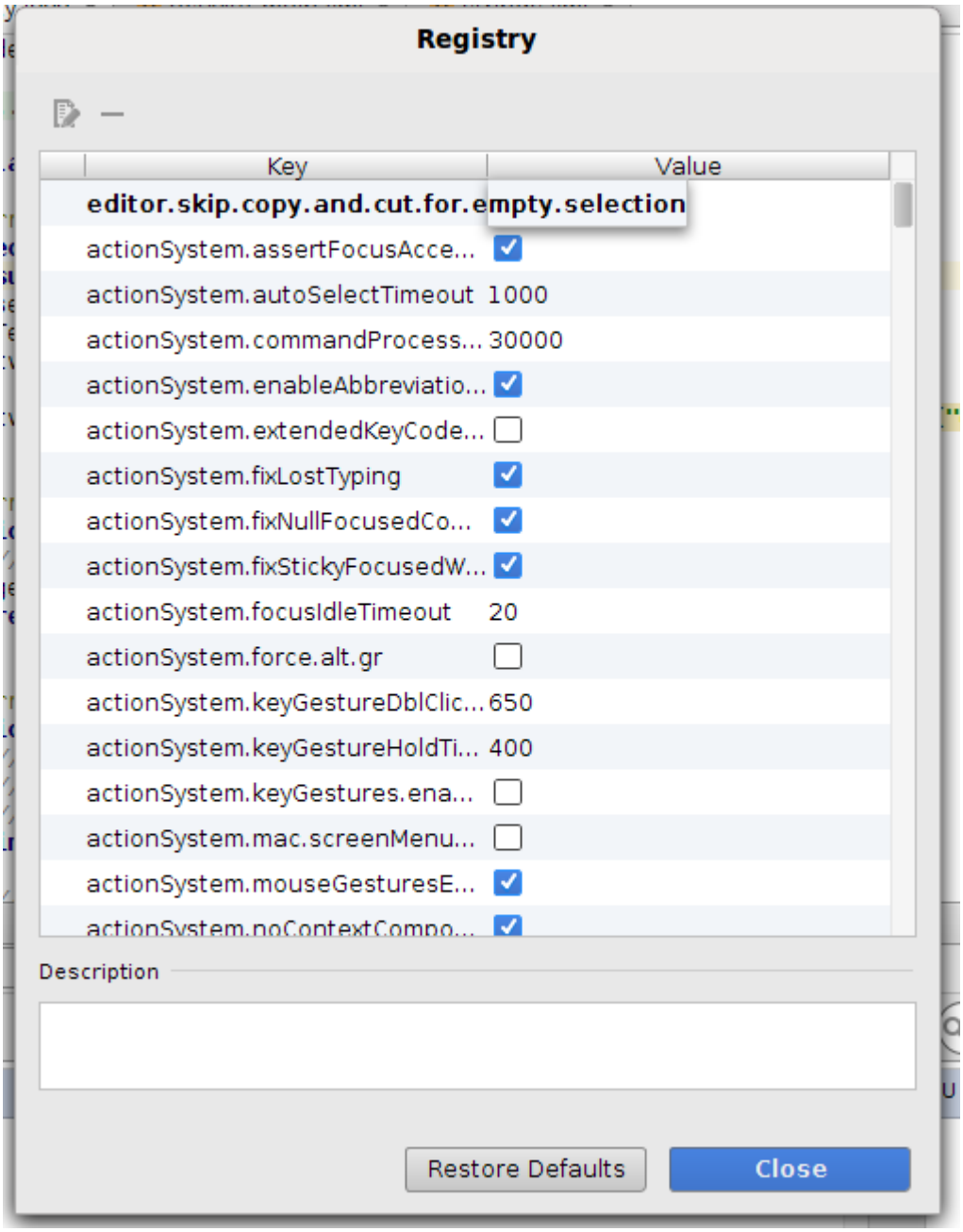


Activar / Desactivar copia de línea en blanco

`ctrl + alt + shift + /` (`cmd + alt + shift + /` en `MacOS`) debería mostrarle el siguiente cuadro de diálogo:



Al hacer clic en el `Registry` obtendrá



La clave que desea habilitar / deshabilitar es

```
editor.skip.copy.and.cut.for.empty.selection
```

Probado en `Linux Ubuntu` y `MacOS` .

Atajos útiles de Android Studio

Los siguientes son algunos de los atajos más comunes / útiles.

Estos se basan en el mapa de acceso directo predeterminado de IntelliJ. Puede cambiar a otros mapas de acceso directo IDE comunes a través de `File -> Settings -> Keymap -> <Choose Eclipse/Visual Studio/etc from Keymaps dropdown>` mapas de teclado `File -> Settings -> Keymap -> <Choose Eclipse/Visual Studio/etc from Keymaps dropdown>`

Acción	Atajo
Código de formato	CTRL + ALT + L
Añadir métodos no implementados	CTRL + I
Mostrar logcat	ALT + 6
Construir	CTRL + F9
Construir y ejecutar	CTRL + F10
Encontrar	CTRL + F
Encontrar en proyecto	CTRL + MAYÚS + F
Encontrar y reemplazar	CTRL + R
Encuentra y reemplaza en proyecto	CTRL + MAYÚS + R
Anular métodos	CTRL + O
Mostrar proyecto	ALT + 1
Ocultar proyecto - logcat	MAYÚS + ESC
Desplegar todo	CTRL + MAYÚS + NumPad +
Ver puntos de depuración	CTRL + MAYÚS + F8
Expandir todo	CTRL + MAYÚS + NumPad -
Configuración abierta	ALT + s
Seleccionar destino (abrir archivo actual en la vista Proyecto)	ALT + F1 → ENTER
Buscar en todas partes	SHIFT → SHIFT (doble turno)
Código | Envolvente con	CTRL → ALT + T
Crear código de forma de código seleccionado	ALT + CTRL

Refactor

Acción	Atajo
Refactor Este (menú / selector para todas las acciones de refactor aplicables del elemento actual)	Mac CTRL + T - Win / Linux CTRL + ALT + T
Rebautizar	MAYÚS + F6

Acción	Atajo
Método de extracción	Mac `CMD` + `ALT` + `M` - Win / Linux `CTRL` + `ALT` + `M`
Extraer Parámetro	Mac `CMD` + `ALT` + `P` - Win / Linux `CTRL` + `ALT` + `P`
Extraer variable	Mac `CMD` + `ALT` + `V` - Win / Linux `CTRL` + `ALT` + `V`

Android Studio Mejorar la punta de rendimiento

Habilitar el trabajo sin conexión:

1. Haga clic en Archivo -> Configuración. Busque "gradle" y haga clic en el cuadro de `Offline work` .
2. Vaya al compilador (en el mismo cuadro de diálogo de configuración justo debajo de `Gradle`) y agregue `--offline` al cuadro de texto `Command-line Options` .

Mejorar el rendimiento de Gradle

Agregue las siguientes dos líneas de código en su archivo gradle.properties.

```
org.gradle.daemon=true
org.gradle.parallel=true
```

Aumentar el valor de `-Xmx` y `-Xms` en el archivo `studio.vmoptions`

```
-Xms1024m
-Xmx4096m
-XX:MaxPermSize=1024m
-XX:ReservedCodeCacheSize=256m
-XX:+UseCompressedOops
```

Ventana

> % USPROFILE%. {FOLDER_NAME} \ studio.exe.vmoptions y / o% USERPROFILE%. {FOLDER_NAME} \ studio64.exe.vmoptions

Mac

> ~ / Library / Preferences / {FOLDER_NAME} /studio.vmoptions

Linux

> ~ /. {FOLDER_NAME} /studio.vmoptions y / o ~ /. {FOLDER_NAME} /studio64.vmoptions

Configurar Android Studio

Requisitos del sistema

- Microsoft® Windows® 8/7 / Vista / 2003 (32 o 64 bits).
- Mac® OS X® 10.8.5 o superior, hasta 10.9 (Mavericks)
- Escritorio de GNOME o KDE

Instalación

Ventana

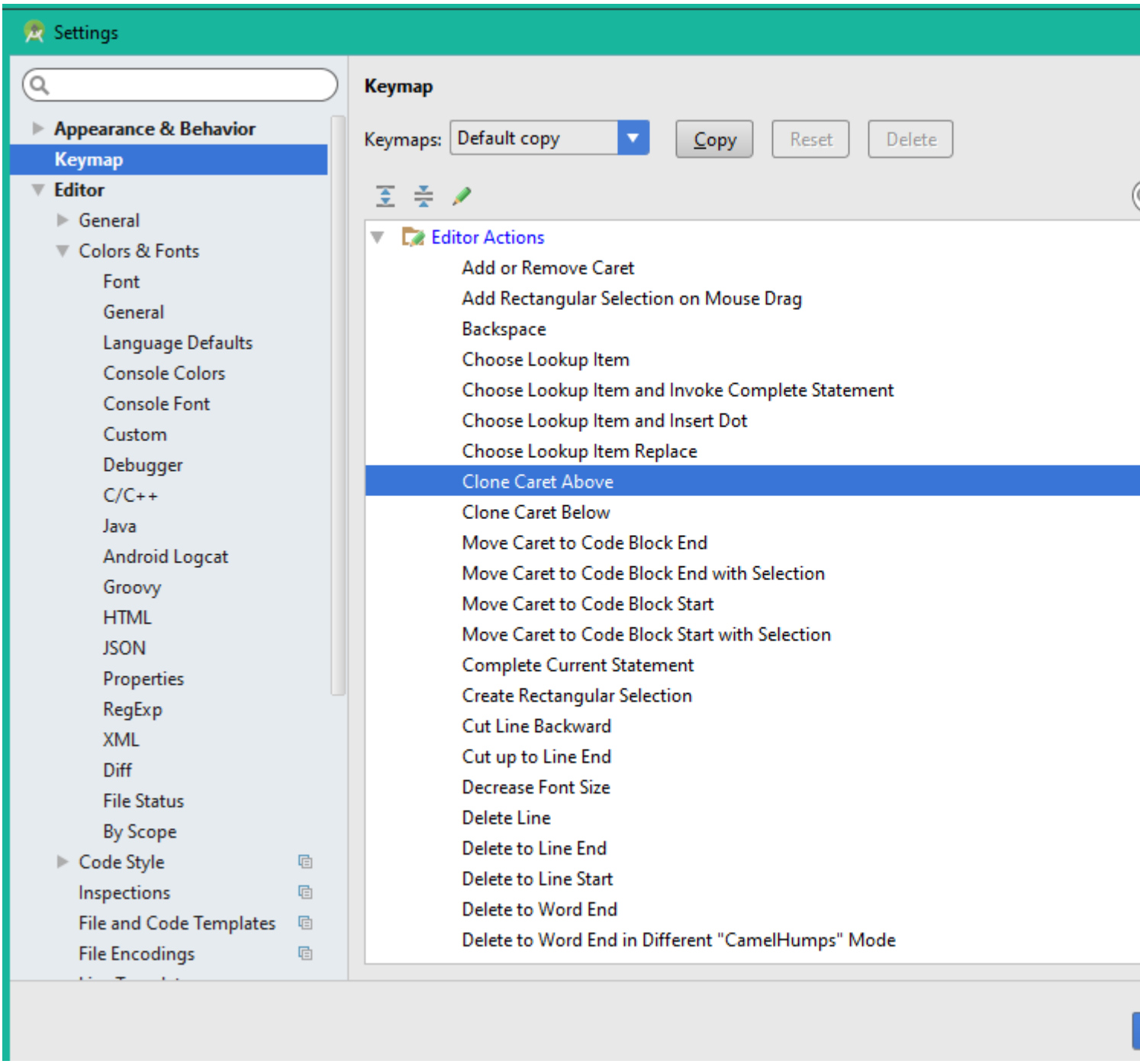
1. Descargue e instale JDK (Java Development Kit) versión 8
2. Descargar Android Studio
3. Inicie `Android Studio.exe` luego mencione la ruta JDK y descargue el último SDK

Linux

1. Descargue e instale JDK (Java Development Kit) versión 8
2. Descargar Android Studio
3. Extraer el archivo zip
4. Abra el terminal, cd a la carpeta extraída, cd a bin (ejemplo `cd android-studio/bin`)
5. Ejecutar ./studio.sh

Ver y agregar accesos directos en Android Studio

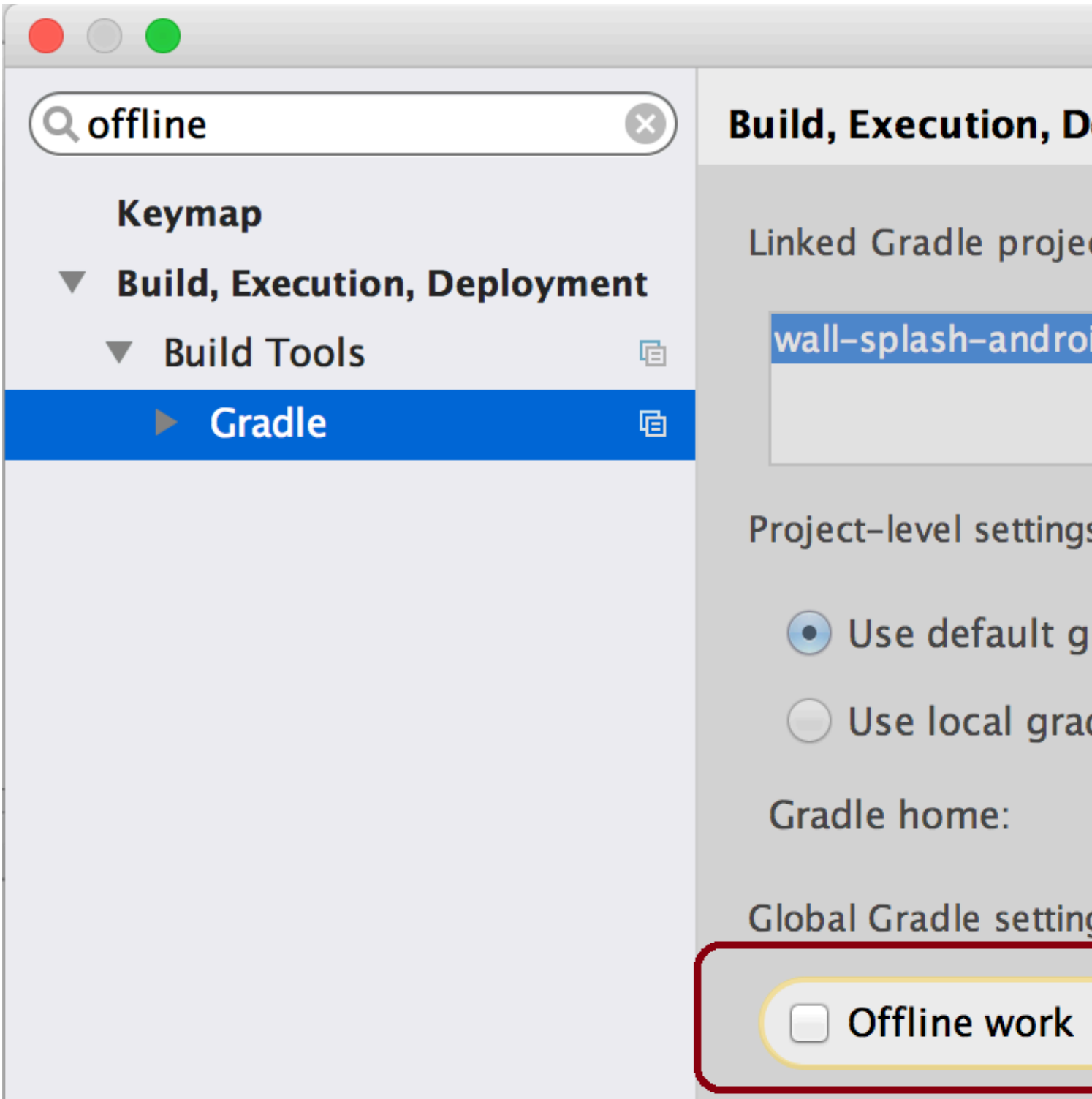
Al acceder a Configuración >> Mapa de teclas, aparecerá una ventana que muestra todas las `Editor Actions` del `Editor Actions` con su nombre y sus accesos directos. Algunas de las `Editor Actions` del `Editor Actions` no tienen accesos directos. Así que haz clic derecho en eso y agrega un nuevo atajo a eso.
Mira la imagen de abajo



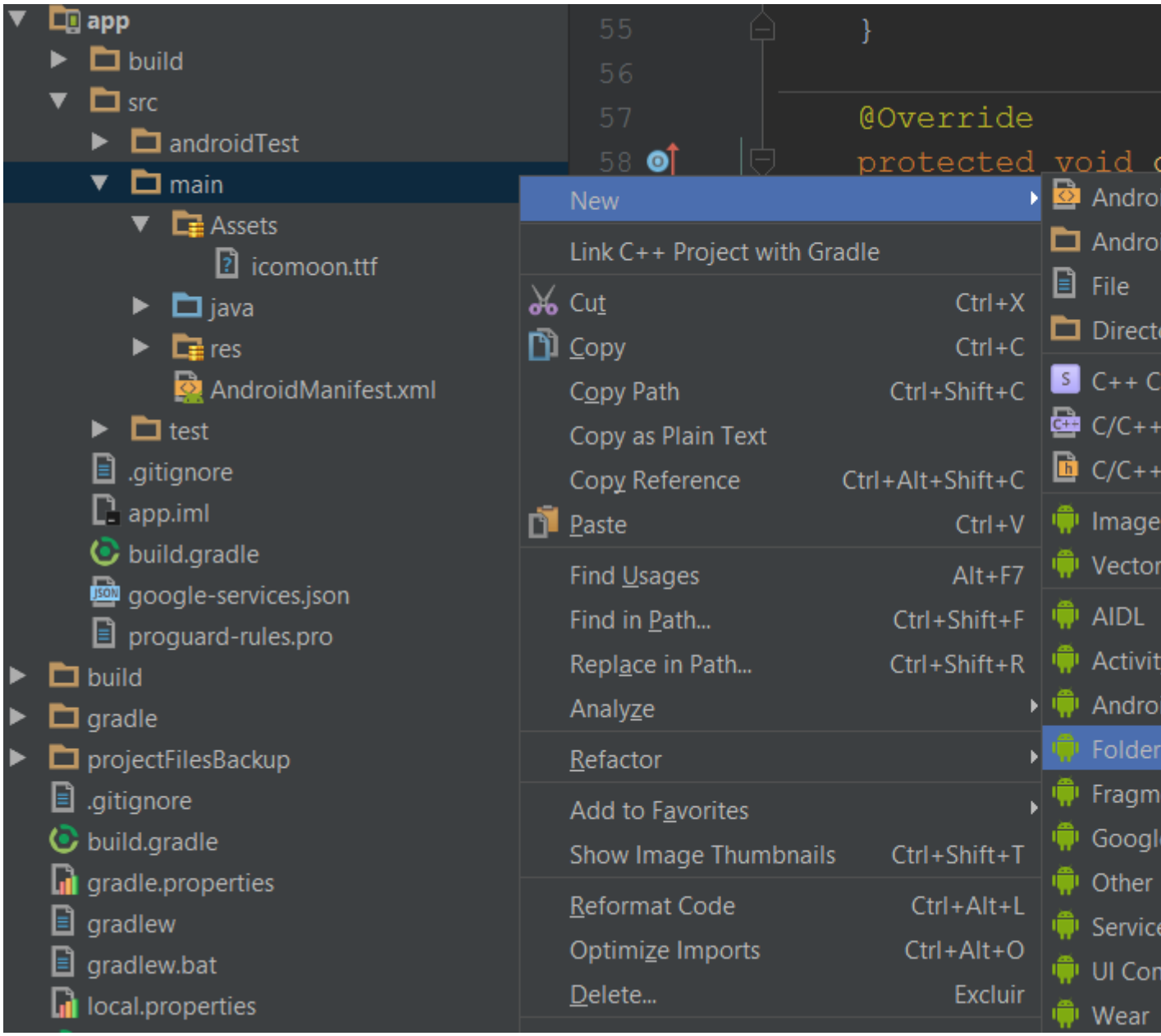
Proyecto de construcción Gradle toma para siempre

Android Studio -> **Preferencias** -> **Gradle** -> Marque **Trabajo sin conexión** y luego reinicie su estudio de Android.

Captura de pantalla de referencia:


offline
Keymap
Build, Execution, Deployment
Build Tools
Gradle
Build, Execution, D
Linked Gradle proje
wall-splash-androi
Project-level settings
Use default g
Use local grad
Gradle home:
Global Gradle setting
Offline work
Service directory

- La carpeta de activos estará debajo de la carpeta PRINCIPAL con el mismo símbolo que la carpeta RES.
- En este ejemplo pongo un archivo de fuente.



Lea Android Studio en línea: https://riptutorial.com/es/android/topic/107/android-studio

Capítulo 16: Android Vk Sdk

Examples

Inicialización y login

1. Crea una nueva aplicación aquí: crear aplicación
2. Elija la aplicación independiente y confirme la creación de la aplicación a través de SMS.
3. Llene el nombre del **paquete para Android** como el nombre de su paquete actual. *Puede obtener el nombre de su paquete dentro del archivo de manifiesto de Android, al principio.*
4. Obtenga su **huella digital de certificado** ejecutando este comando en su shell / cmd:

```
keytool -exportcert -alias androiddebugkey -keystore path-to-debug-or-production-keystore -
list -v
```

También puede obtener esta huella digital mediante el propio SDK:

```
String[] fingerprints = VKUtil.getCertificateFingerprint(this, this.getPackageName());
Log.d("MainActivity", fingerprints[0]);
```

5. Agregue la huella digital recibida en el campo de **huella digital del certificado de firma para Android:** en la configuración de la aplicación Vk (donde ingresó el nombre de su paquete)

6. Luego agrega esto a tu archivo de gradle:

```
compile 'com.vk:androidsdk:1.6.5'
```

8. Inicialice el SDK en el inicio utilizando el siguiente método. La mejor manera es llamarlo en el método de aplicaciones onCreate.

```
private static final int VK_ID = your_vk_id;
public static final String VK_API_VERSION = "5.52"; //current version
@Override
    public void onCreate() {
        super.onCreate();
        VKSdk.customInitialize(this, VK_ID, VK_API_VERSION);
}
```

Esta es la mejor manera de iniciar VKSdk. No uses el método de metida donde se debe colocar VK_ID dentro de strings.xml porque la API no funcionará correctamente después de eso.

9. El paso final es iniciar sesión usando vksdk.

```
    public static final String[] VK_SCOPES = new String[]{
            VKScope.FRIENDS,
            VKScope.MESSAGES,
            VKScope.NOTIFICATIONS,
```

```
            VKScope.OFFLINE,
            VKScope.STATUS,
            VKScope.STATS,
            VKScope.PHOTOS
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        someButtonForLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               VKSdk.login(this, VK_SCOPES);
            }
        });

    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        VKSdk.onActivityResult(requestCode, resultCode, data, new VKCallback<VKAccessToken>()
{
            @Override
            public void onResult(VKAccessToken res) {
                res.accessToken; //getting our token here.
            }

            @Override
            public void onError(VKError error) {
                Toast.makeText(SocialNetworkChooseActivity.this,
                        "User didn't pass Authorization", Toast.LENGTH_SHORT).show();
            }
        });
    }
```

Lea Android Vk Sdk en línea: https://riptutorial.com/es/android/topic/6046/android-vk-sdk

Capítulo 17: Android-x86 en VirtualBox

Introducción

La idea de esta sección es cubrir cómo instalar y usar VirtualBox con Android-x86 para fines de depuración. Esta es una tarea difícil porque hay diferencias entre las versiones. Por el momento voy a cubrir 6.0, que es con el que tuve que trabajar y luego tendremos que encontrar similitudes.

No cubre VirtualBox o Linux en detalle, pero muestra los comandos que he usado para hacer que funcione.

Examples

Configuración de la máquina virtual

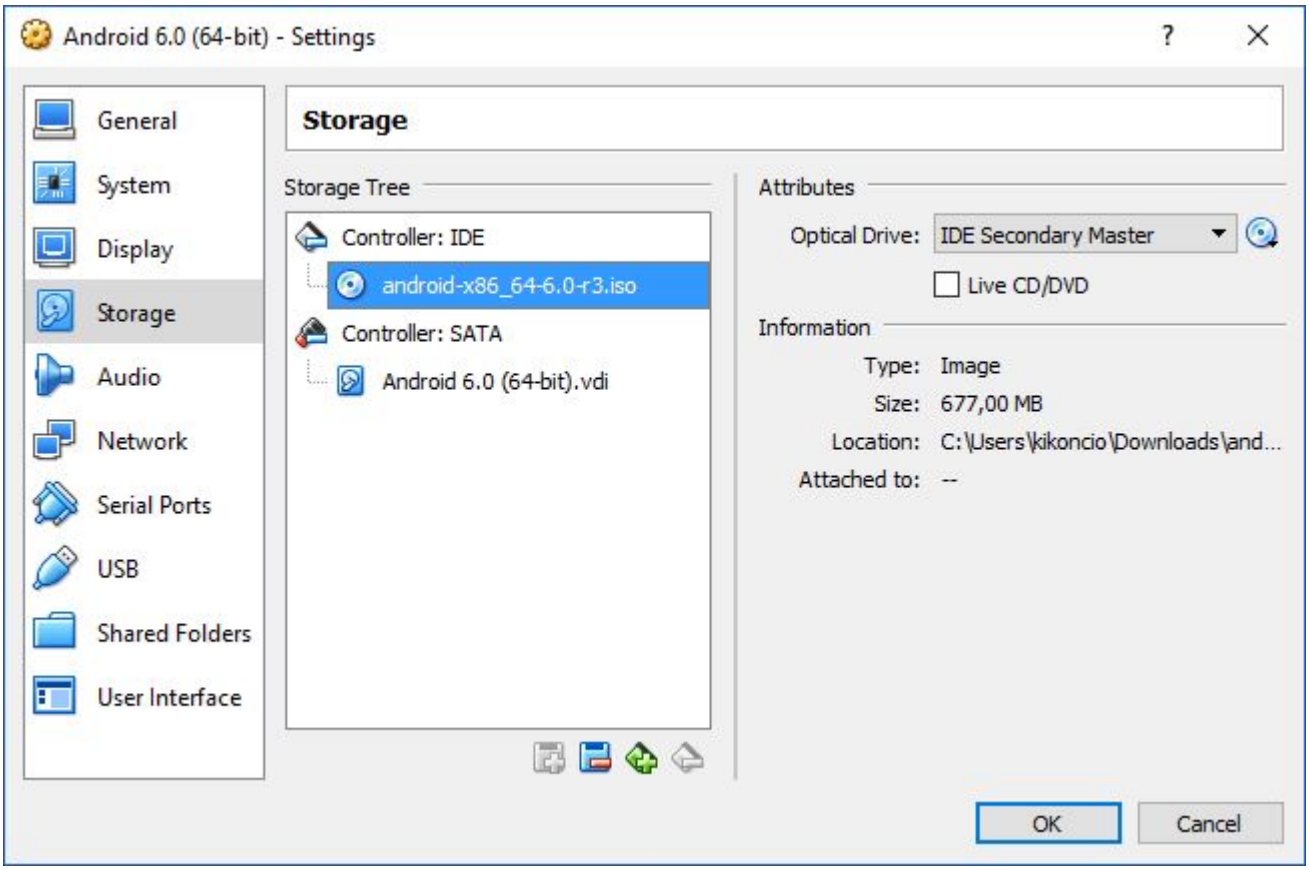
Estas son mis configuraciones de VirtualBox:

- Tipo de SO: Linux 2.6 (tengo un usuario de 64 bits porque mi computadora puede admitirlo)
- Tamaño del disco duro virtual: 4Gb
- Memoria RAM: 2048
- Memoria de video: 8M
- Dispositivo de sonido: Sound Blaster 16.
- Dispositivo de red: PCnet-Fast III, conectado a NAT. También puede usar un adaptador puenteado, pero necesita un servidor DHCP en su entorno.

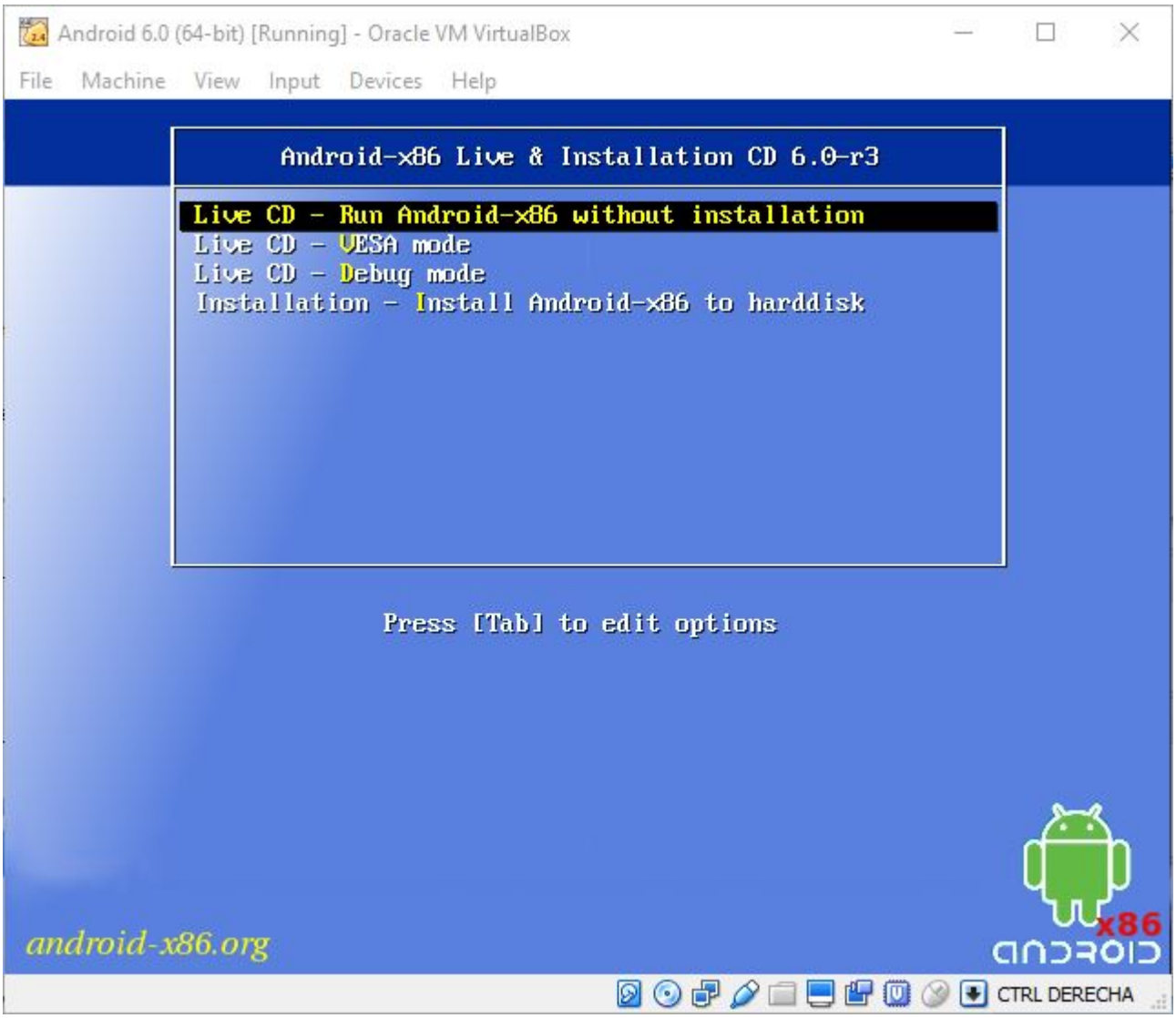
La imagen utilizada con esta configuración ha sido android-x86_64-6.0-r3.iso (es de 64 bits) descargada de http://www.android-x86.org/download . Supongo que también funciona con la versión de 32 bits.

Configuración de disco duro virtual para soporte de SDCARD

Con el disco duro virtual que acaba de crear, inicie la máquina virtual con la imagen de android-x86 en la unidad óptica.



Una vez que arranque, puede ver el menú de grub del Live CD



Elija la opción de modo de depuración, entonces debería ver el indicador del shell. Este es un shell de busybox. Puede obtener más shell cambiando entre la consola virtual Alt-F1 / F2 / F3.

Cree dos particiones por fdisk (algunas otras versiones usarían cfdisk). Formatearlos a ext3. Luego reinicie:

```
# fdisk /dev/sda
```

A continuación, escriba:

"n" (nueva partición)

"p" (partición primaria)

"1" (1ª partición)

"1" (primer cilindro)

"261" (elija un cilindro, dejaremos el 50% del disco para una segunda partición)

"2" (2ª partición)

"262" (262nd cilindro)

"522" (elegir el último cilindro)

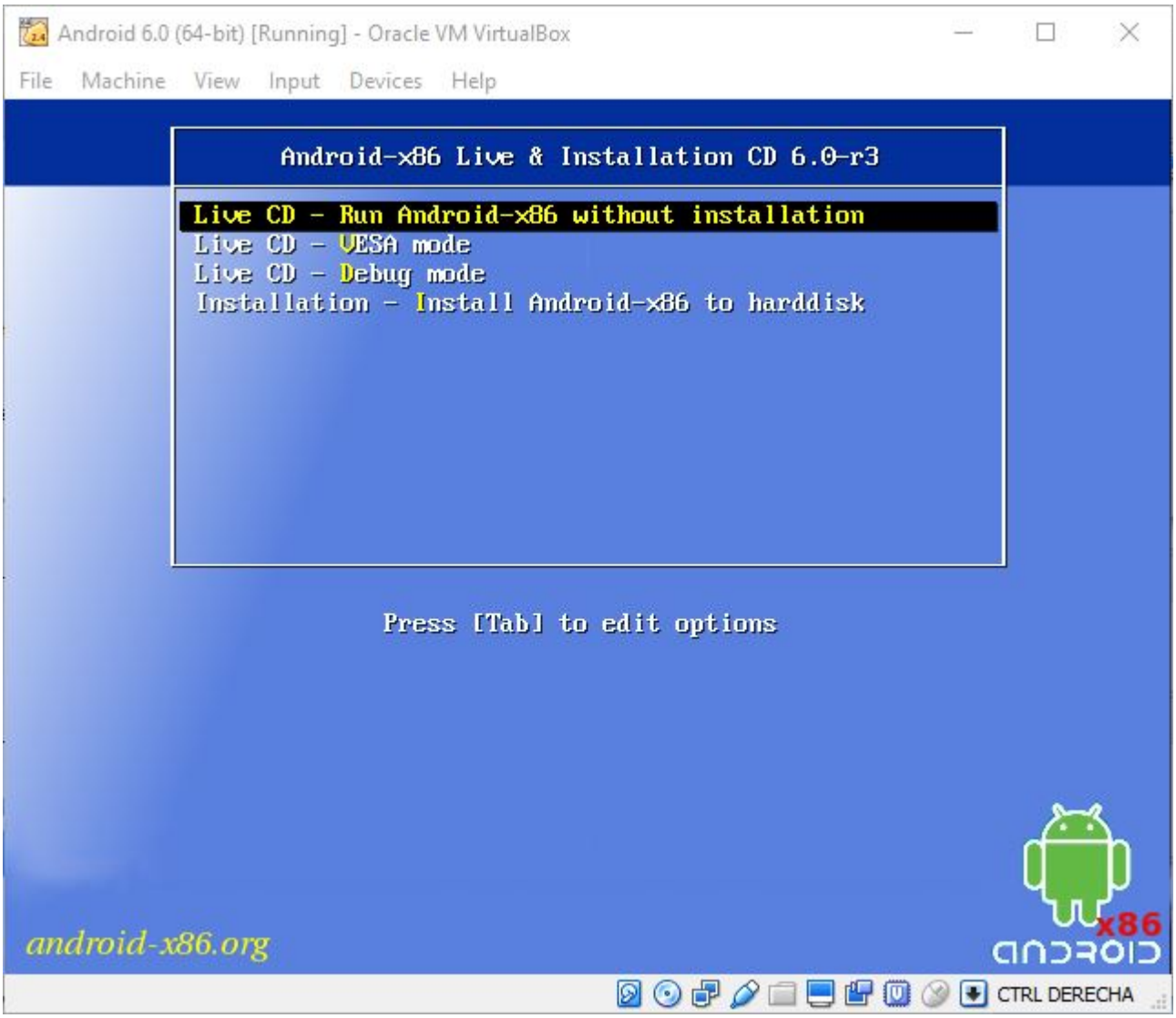
"w" (escribe la partición)

```
#mdev -s
#mke2fs -j -L DATA /dev/sda1
#mke2fs -j -L SDCARD /dev/sda2
#reboot -f
```

Cuando reinicie la máquina virtual y aparezca el menú de grub, podrá editar la línea de arranque del kernel para que pueda agregar las opciones `DATA=sda1 SDCARD=sda2` para apuntar a la sdcard o la partición de datos.

Instalación en partición

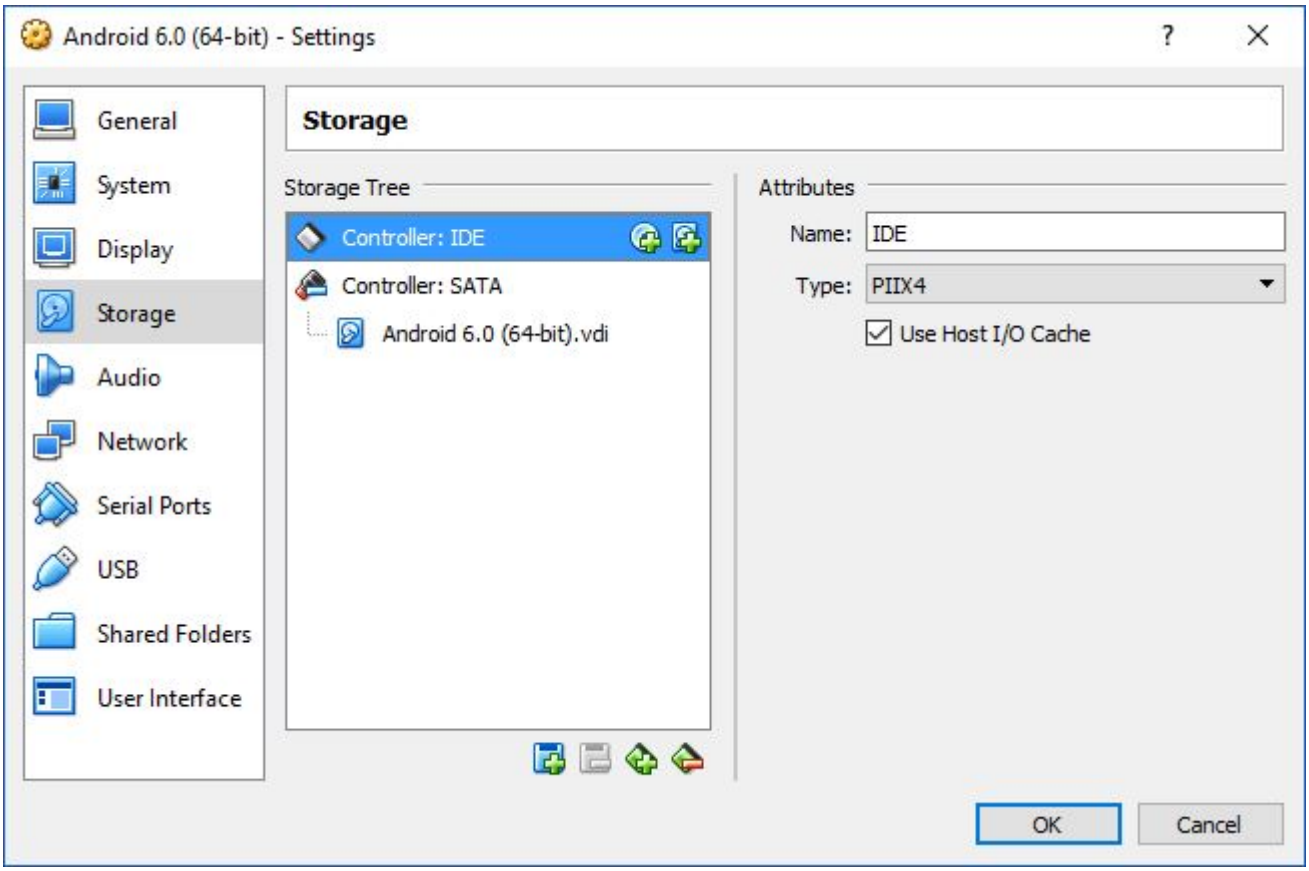
Con el disco duro virtual que se acaba de crear, inicie la máquina virtual con la imagen de android-x86 como unidad óptica.



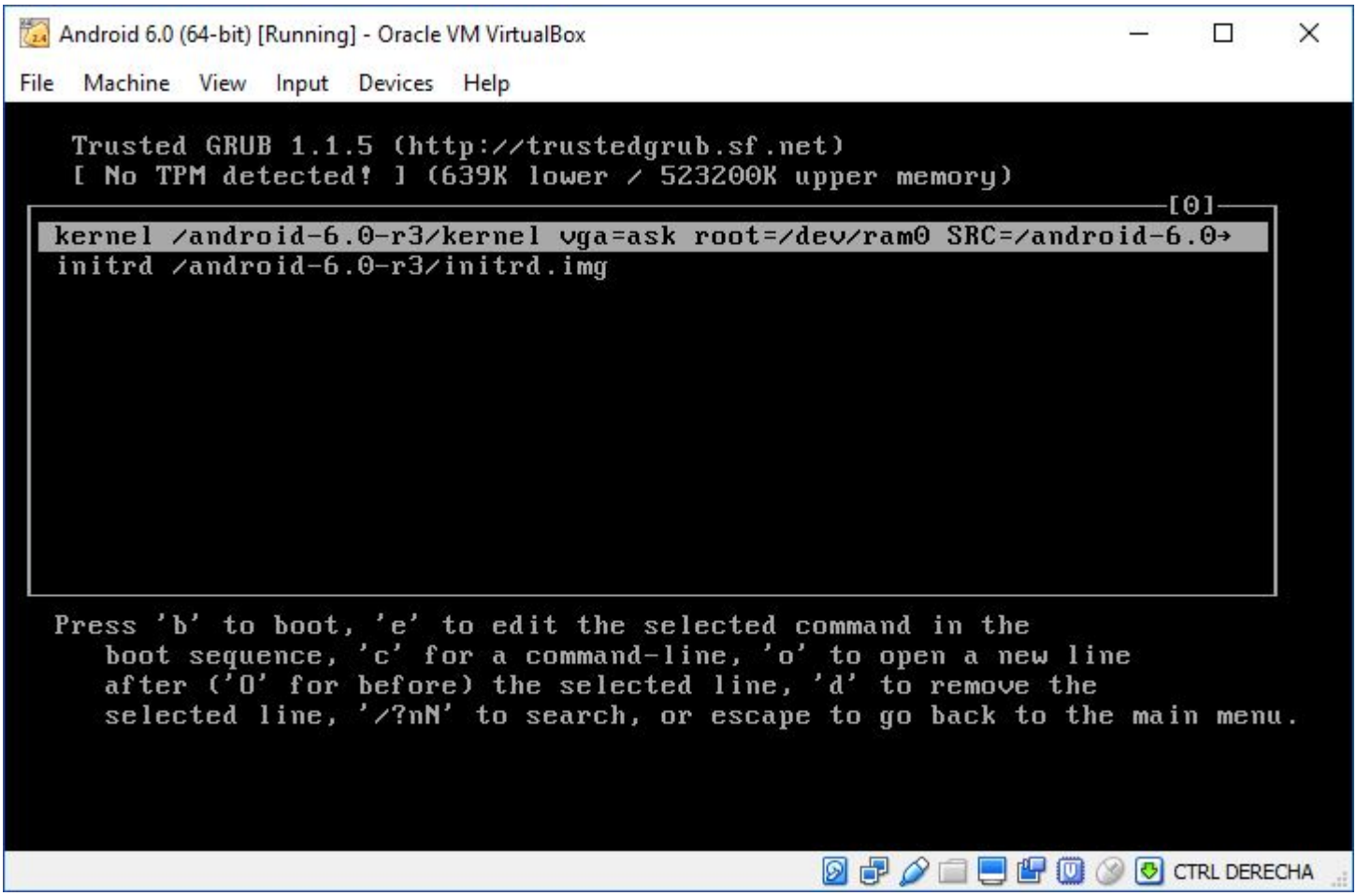
En las opciones de arranque del Live CD, elija "Instalación - Instalar Android en el disco duro"

Elige la partición sda1 e instala Android y nosotros instalaremos grub.

Reinicie la máquina virtual, pero asegúrese de que la imagen no esté en la unidad óptica para que pueda reiniciarse desde la unidad de disco virtual.



En el menú de grub necesitamos editar el kernel como en la opción "Android-x86 6.0-r3", así que presione e.

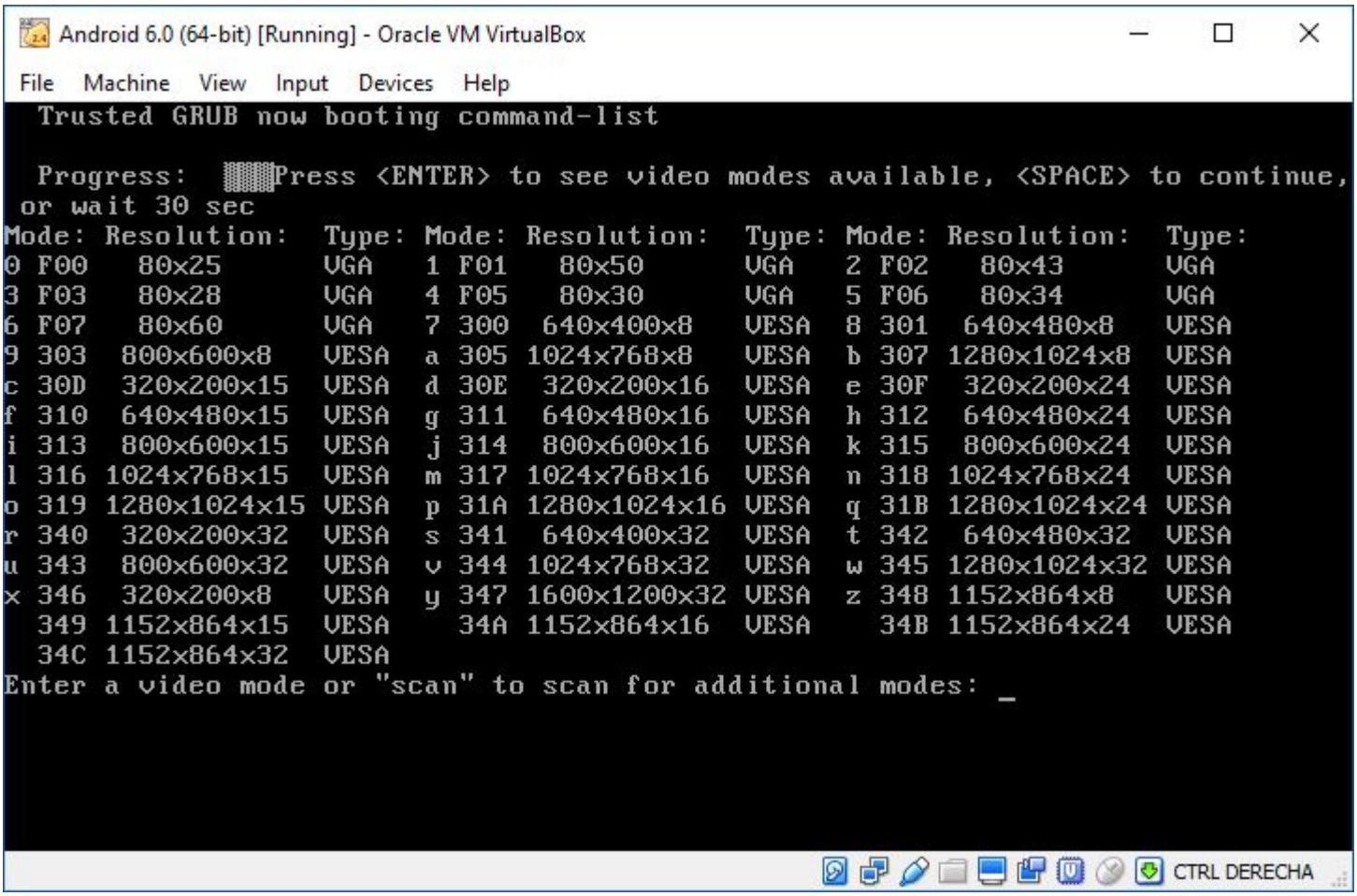


Luego sustituimos "quiet" con "vga = ask" y agregamos la opción "SDCARD = sda2"

En mi caso, la línea del kernel se ve así después de la modificación:

```
kenel /android-6.0-r3/kernel vga=ask root=ram0 SRC=/android-6/android-6.0-r3 SDCARD=sda2
```

Presione b para iniciar, luego podrá elegir el tamaño de la pantalla presionando ENTER (la opción `vga=ask`)



Una vez que el asistente de instalación ha comenzado a elegir el idioma. Podía elegir inglés (Estados Unidos) y español (Estados Unidos) y tuve problemas para elegir cualquier otro.

Lea Android-x86 en VirtualBox en línea: https://riptutorial.com/es/android/topic/9903/android-x86-en-virtualbox

Capítulo 18: Animadores

Examples

Agitar la animación de un ImageView

En la carpeta res, cree una nueva carpeta llamada "anim" para almacenar sus recursos de animación y colóquela en esa carpeta.

shakeanimation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="100"
    android:fromDegrees="-15"
    android:pivotX="50%"
    android:pivotY="50%"
    android:repeatCount="infinite"
    android:repeatMode="reverse"
    android:toDegrees="15" />
```

Crear una actividad en blanco llamada Aterrizaje

activity_landing.xml

```
  <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ImageView
            android:id="@+id/imgBell"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:src="@mipmap/ic_notifications_white_48dp"/>

    </RelativeLayout>
```

Y el método para animar la vista de imagen en Landing.java.

```
Context mContext;


 @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            mContext=this;
                setContentView(R.layout.activity_landing);

                AnimateBell();
        }

public void AnimateBell() {
```

```
        Animation shake = AnimationUtils.loadAnimation(mContext, R.anim.shakeanimation);
        ImageView imgBell= (ImageView) findViewById(R.id.imgBell);
        imgBell.setImageResource(R.mipmap.ic_notifications_active_white_48dp);
        imgBell.setAnimation(shake);
    }
```

Fade in / out animación

Para obtener una vista que desaparezca o desaparezca lentamente, use un `ObjectAnimator` . Como se ve en el código a continuación, establezca una duración utilizando `.setDuration(millis)` donde el parámetro `millis` es la duración (en milisegundos) de la animación. En el código de abajo, las vistas se desvanecerán a lo largo de 500 milisegundos, o 1/2 segundo. Para iniciar la `ObjectAnimator` del `ObjectAnimator` , llame a `.start()` . Una vez que se completa la animación, se `onAnimationEnd(Animator animation)` . Aquí es un buen lugar para establecer la visibilidad de su vista en `View.GONE` o `View.VISIBLE` .

```
import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.animation.ValueAnimator;

void fadeOutAnimation(View viewToFadeOut) {
    ObjectAnimator fadeOut = ObjectAnimator.ofFloat(viewToFadeOut, "alpha", 1f, 0f);

    fadeOut.setDuration(500);
    fadeOut.addListener(new AnimatorListenerAdapter() {
        @Override
        public void onAnimationEnd(Animator animation) {
            // We wanna set the view to GONE, after it's fade out. so it actually disappear
from the layout & don't take up space.
            viewToFadeOut.setVisibility(View.GONE);
        }
    });

    fadeOut.start();
}

void fadeInAnimation(View viewToFadeIn) {
    ObjectAnimator fadeIn = ObjectAnimator.ofFloat(viewToFadeIn, "alpha", 0f, 1f);
    fadeIn.setDuration(500);

    fadeIn.addListener(new AnimatorListenerAdapter() {
        @Override
        public void onAnimationStar(Animator animation) {
            // We wanna set the view to VISIBLE, but with alpha 0. So it appear invisible in
the layout.
            viewToFadeIn.setVisibility(View.VISIBLE);
            viewToFadeIn.setAlpha(0);
        }
    });

    fadeIn.start();
}
```

Animación transitionDrawable

Este ejemplo muestra una transacción para una vista de imagen con solo dos imágenes (puede

usar más imágenes y una después de la otra para las posiciones de la primera y la segunda capa después de cada transacción como un bucle)

- agrega una matriz de imágenes a `res/values/arrays.xml`

```
<resources>

    <array
        name="splash_images">
        <item>@drawable/spash_imge_first</item>
        <item>@drawable/spash_img_second</item>
    </array>

</resources>
```

```
    private Drawable[] backgroundsDrawableArrayForTransition;
    private TransitionDrawable transitionDrawable;

private void backgroundAnimTransAction() {

        // set res image array
        Resources resources = getResources();
        TypedArray icons = resources.obtainTypedArray(R.array.splash_images);

        @SuppressWarnings("ResourceType")
        Drawable drawable = icons.getDrawable(0);    // ending image

        @SuppressWarnings("ResourceType")
        Drawable drawableTwo = icons.getDrawable(1);   // starting image


        backgroundsDrawableArrayForTransition = new Drawable[2];
        backgroundsDrawableArrayForTransition[0] = drawable;
        backgroundsDrawableArrayForTransition[1] = drawableTwo;
        transitionDrawable = new TransitionDrawable(backgroundsDrawableArrayForTransition);

        // your image view here - backgroundImageView
        backgroundImageView.setImageDrawable(transitionDrawable);
        transitionDrawable.startTransition(4000);

        transitionDrawable.setCrossFadeEnabled(false); // call public methods


    }
```

ValueAnimator

`ValueAnimator` introduce una forma sencilla de animar un valor (de un tipo en particular, por ejemplo, `int` , `float` , etc.).

La forma habitual de usarlo es:

1. Cree un `ValueAnimator` que animará un valor de `min` a `max`
2. Agregue un `UpdateListener` en el que usará el valor animado calculado (que puede obtener con `getAnimatedValue()`)

Hay dos formas de crear el `ValueAnimator` :

(el código de ejemplo anima un `float` de `20f` a `40f` en `250ms`)

1. Desde `xml` (póngalo en `/res/animator/`):

```
<animator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="250"
    android:valueFrom="20"
    android:valueTo="40"
    android:valueType="floatType"/>
```

```
ValueAnimator animator = (ValueAnimator) AnimatorInflater.loadAnimator(context,
        R.animator.example_animator);
animator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator anim) {
        // ... use the anim.getAnimatedValue()
    }
});
// set all the other animation-related stuff you want (interpolator etc.)
animator.start();
```

2. Desde el código:

```
ValueAnimator animator = ValueAnimator.ofFloat(20f, 40f);
animator.setDuration(250);
animator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator anim) {
        // use the anim.getAnimatedValue()
    }
});
// set all the other animation-related stuff you want (interpolator etc.)
animator.start();
```

ObjectAnimator

`ObjectAnimator` es una subclase de `ValueAnimator` con la capacidad adicional de establecer el valor calculado en la propiedad de una `View` `target` .

Al igual que en el `ValueAnimator` , hay dos formas de crear el `ObjectAnimator` :

(el código ejemplo, se anima un `alpha` de un `View` desde `0.4f` a `0.2f` en `250ms`)

1. Desde `xml` (ponlo en el `/res/animator`)

```
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
                android:duration="250"
                android:propertyName="alpha"
                android:valueFrom="0.4"
                android:valueTo="0.2"
                android:valueType="floatType"/>
```

```
ObjectAnimator animator = (ObjectAnimator) AnimatorInflater.loadAnimator(context,
        R.animator.example_animator);
animator.setTarget(exampleView);
// set all the animation-related stuff you want (interpolator etc.)
animator.start();
```

2. Desde el código:

```
ObjectAnimator animator = ObjectAnimator.ofFloat(exampleView, View.ALPHA, 0.4f, 0.2f);
animator.setDuration(250);
// set all the animation-related stuff you want (interpolator etc.)
animator.start();
```

ViewPropertyAnimator

`ViewPropertyAnimator` es una forma simplificada y optimizada de animar las propiedades de una `View` .

Cada `View` individual tiene un objeto `ViewPropertyAnimator` disponible a través del método `animate()` . Puede usar eso para animar múltiples propiedades a la vez con una simple llamada. Cada método único de un `ViewPropertyAnimator` especifica el valor **objetivo** de un parámetro específico con el que debe animarse el `ViewPropertyAnimator` .

```
View exampleView = ...;
exampleView.animate()
        .alpha(0.6f)
        .translationY(200)
        .translationXBy(10)
        .scaleX(1.5f)
        .setDuration(250)
        .setInterpolator(new FastOutLinearInInterpolator());
```

Nota: Llamar a `start()` en un objeto `ViewPropertyAnimator` **NO** es obligatorio. Si no lo hace, simplemente está dejando que la plataforma maneje el inicio de la animación en el momento adecuado (siguiente pase de manejo de la animación). Si realmente haces eso (`start()` llamada `start()`), te aseguras de que la animación se inicie de inmediato.

Expandir y contraer la animación de la vista.

```
public class ViewAnimationUtils {

        public static void expand(final View v) {
            v.measure(LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT);
            final int targtetHeight = v.getMeasuredHeight();

            v.getLayoutParams().height = 0;
            v.setVisibility(View.VISIBLE);
            Animation a = new Animation()
            {
                @Override
                protected void applyTransformation(float interpolatedTime, Transformation t) {
                    v.getLayoutParams().height = interpolatedTime == 1
                            ? LayoutParams.WRAP_CONTENT
```

```
                            : (int)(targtetHeight * interpolatedTime);
                    v.requestLayout();
                }

                @Override
                public boolean willChangeBounds() {
                    return true;
                }
            };

            a.setDuration((int)(targtetHeight /
v.getContext().getResources().getDisplayMetrics().density));
            v.startAnimation(a);
        }

        public static void collapse(final View v) {
            final int initialHeight = v.getMeasuredHeight();

            Animation a = new Animation()
            {
                @Override
                protected void applyTransformation(float interpolatedTime, Transformation t) {
                    if(interpolatedTime == 1){
                        v.setVisibility(View.GONE);
                    }else{
                        v.getLayoutParams().height = initialHeight - (int)(initialHeight *
interpolatedTime);
                        v.requestLayout();
                    }
                }

                @Override
                public boolean willChangeBounds() {
                    return true;
                }
            };

            a.setDuration((int)(initialHeight /
v.getContext().getResources().getDisplayMetrics().density));
            v.startAnimation(a);
        }
    }
```

Lea Animadores en línea: https://riptutorial.com/es/android/topic/1829/animadores

Capítulo 19: Anotaciones Typedef: @IntDef, @StringDef

Observaciones

El paquete de anotaciones incluye una serie de anotaciones de metadatos útiles con las que puede decorar su propio código para ayudar a detectar errores.

Solo agrega la dependencia en el archivo `build.gradle` .

```
dependencies {
    compile 'com.android.support:support-annotations:25.3.1'
}
```

Examples

Anotaciones IntDef

Esta anotación garantiza que solo se utilicen las constantes enteras válidas que espera.
El siguiente ejemplo ilustra los pasos para crear una anotación:

```
import android.support.annotation.IntDef;

public abstract class Car {

    //Define the list of accepted constants
    @IntDef({MICROCAR, CONVERTIBLE, SUPERCAR, MINIVAN, SUV})

    //Tell the compiler not to store annotation data in the .class file
    @Retention(RetentionPolicy.SOURCE)
    //Declare the CarType annotation
    public @interface CarType {}

    //Declare the constants
    public static final int MICROCAR = 0;
    public static final int CONVERTIBLE = 1;
    public static final int SUPERCAR = 2;
    public static final int MINIVAN = 3;
    public static final int SUV = 4;

    @CarType
    private int mType;

    @CarType
    public int getCarType(){
        return mType;
    };

    public void setCarType(@CarType int type){
        mType = type;
    }
```

```
}
```

También permiten la finalización del código para ofrecer automáticamente las constantes permitidas.
Cuando crea este código, se genera una advertencia si el parámetro de tipo no hace referencia a una de las constantes definidas.

Combinando constantes con banderas

Usando el `IntDef#flag()` establecido en `true` , se pueden combinar múltiples constantes.

Usando el mismo ejemplo en este tema:

```
public abstract class Car {

    //Define the list of accepted constants
    @IntDef(flag=true, value={MICROCAR, CONVERTIBLE, SUPERCAR, MINIVAN, SUV})

    //Tell the compiler not to store annotation data in the .class file
    @Retention(RetentionPolicy.SOURCE)

    .....

}
```

Los usuarios pueden combinar las constantes permitidas con una marca (como `|` , `&` , `^`).

Lea Anotaciones Typedef: @IntDef, @StringDef en línea:
https://riptutorial.com/es/android/topic/4505/anotaciones-typedef---intdef---stringdef

Capítulo 20: API de Android Places

Examples

Ejemplo de uso del selector de lugar

Place Picker es un widget de interfaz de usuario realmente simple proporcionado por la API de Places. Proporciona un mapa incorporado, ubicación actual, lugares cercanos, capacidades de búsqueda y autocompletar.

Este es un ejemplo de uso del widget de la interfaz de usuario del Selector de lugares.

```
private static int PLACE_PICKER_REQUEST = 1;

private TextView txtPlaceName;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_place_picker_sample);

    txtPlaceName = (TextView) this.findViewById(R.id.txtPlaceName);
    Button btnSelectPlace = (Button) this.findViewById(R.id.btnSelectPlace);
    btnSelectPlace.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            openPlacePickerView();
        }
    });

}

private void openPlacePickerView(){
    PlacePicker.IntentBuilder builder = new PlacePicker.IntentBuilder();
    try {
        startActivityForResult(builder.build(this), PLACE_PICKER_REQUEST);
    } catch (GooglePlayServicesRepairableException e) {
        e.printStackTrace();
    } catch (GooglePlayServicesNotAvailableException e) {
        e.printStackTrace();
    }
}

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == PLACE_PICKER_REQUEST) {
        if (resultCode == RESULT_OK) {
            Place place = PlacePicker.getPlace(this, data);
            Log.i(LOG_TAG, String.format("Place Name : %s", place.getName()));
            Log.i(LOG_TAG, String.format("Place Address : %s", place.getAddress()));
            Log.i(LOG_TAG, String.format("Place Id : %s", place.getId()));

            txtPlaceName.setText(String.format("Place : %s - %s" , place.getName() ,
place.getAddress()));
        }
    }
```

```
}
```

Obtener lugares actuales utilizando la API de lugares

Puede obtener la ubicación actual y los lugares locales de usuario utilizando la API de Google Places .

Primero, debe llamar al método `PlaceDetectionApi.getCurrentPlace()` para recuperar negocios locales u otros lugares. Este método devuelve un objeto `PlaceLikelihoodBuffer` que contiene una lista de objetos `PlaceLikelihood` . Luego, puede obtener un objeto `Place` llamando al método `PlaceLikelihood.getPlace()` .

Importante: debe solicitar y obtener el permiso `ACCESS_FINE_LOCATION` para permitir que su aplicación acceda a información de ubicación precisa.

```
private static final int PERMISSION_REQUEST_TO_ACCESS_LOCATION = 1;

private TextView txtLocation;
private GoogleApiClient googleApiClient;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_location);

    txtLocation = (TextView) this.findViewById(R.id.txtLocation);
    googleApiClient = new GoogleApiClient.Builder(this)
            .addApi(Places.GEO_DATA_API)
            .addApi(Places.PLACE_DETECTION_API)
            .enableAutoManage(this, this)
            .build();

    getCurrentLocation();
}

private void getCurrentLocation() {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        Log.e(LOG_TAG, "Permission is not granted");

        ActivityCompat.requestPermissions(this,new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},PERMISSION_REQUEST_TO_ACCESS_LOCATION);
        return;
    }

    Log.i(LOG_TAG, "Permission is granted");

    PendingResult<PlaceLikelihoodBuffer> result =
Places.PlaceDetectionApi.getCurrentPlace(googleApiClient, null);
    result.setResultCallback(new ResultCallback<PlaceLikelihoodBuffer>() {
        @Override
        public void onResult(PlaceLikelihoodBuffer likelyPlaces) {
            Log.i(LOG_TAG, String.format("Result received : %d " , likelyPlaces.getCount() ));
            StringBuilder stringBuilder = new StringBuilder();

            for (PlaceLikelihood placeLikelihood : likelyPlaces) {
                stringBuilder.append(String.format("Place : '%s' %n",
```

```
                        placeLikelihood.getPlace().getName()));
            }
            likelyPlaces.release();
            txtLocation.setText(stringBuilder.toString());
        }
    });
}

@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[]
grantResults) {
    switch (requestCode) {
        case PERMISSION_REQUEST_TO_ACCESS_LOCATION: {
            // If the request is cancelled, the result arrays are empty.
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                getCurrentLocation();
            } else {
                // Permission denied, boo!
                // Disable the functionality that depends on this permission.
            }
            return;
        }

        // Add further 'case' lines to check for other permissions this app might request.
    }
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    Log.e(LOG_TAG, "GoogleApiClient connection failed: " +
connectionResult.getErrorMessage());
}
```

Integración automática de lugares

La función de autocompletar en la API de Google Places para Android proporciona predicciones de lugar al usuario. Mientras el usuario escribe en el cuadro de búsqueda, autocompletar muestra los lugares de acuerdo con las consultas del usuario.

AutoCompleteActivity.java

```
private TextView txtSelectedPlaceName;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_autocomplete);

    txtSelectedPlaceName = (TextView) this.findViewById(R.id.txtSelectedPlaceName);

    PlaceAutocompleteFragment autocompleteFragment = (PlaceAutocompleteFragment)
            getFragmentManager().findFragmentById(R.id.fragment_autocomplete);

    autocompleteFragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {
        @Override
        public void onPlaceSelected(Place place) {
            Log.i(LOG_TAG, "Place: " + place.getName());
```

```
            txtSelectedPlaceName.setText(String.format("Selected places : %s  - %s" ,
place.getName(), place.getAddress()));
        }

        @Override
        public void onError(Status status) {
            Log.i(LOG_TAG, "An error occurred: " + status);
            Toast.makeText(AutoCompleteActivity.this, "Place cannot be selected!!",
Toast.LENGTH_SHORT).show();
        }
    });

}
```

}

activity_autocomplete.xml

```
<fragment
    android:id="@+id/fragment_autocomplete"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:name="com.google.android.gms.location.places.ui.PlaceAutocompleteFragment"
    />



<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/txtSelectedPlaceName"
    android:layout_margin="20dp"
    android:padding="15dp"
    android:hint="@string/txt_select_place_hint"
    android:textSize="@dimen/place_autocomplete_prediction_primary_text"/>
```

Agregando más de una actividad de google auto complete.

```
 public static final int PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE=1;
   public static final int PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE=2;

fromPlaceEdit.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    try {
                      //Do your stuff from place
                        startActivityForResult(intent,
PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE);

                    } catch (GooglePlayServicesRepairableException e) {
                        // TODO: Handle the error.
                    } catch (GooglePlayServicesNotAvailableException e) {
                        // TODO: Handle the error.
                    }
                }
            });
toPlaceEdit.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {

                try {
                   //Do your stuff to place
                    startActivityForResult(intent, PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE);

                } catch (GooglePlayServicesRepairableException e) {
                    // TODO: Handle the error.
                } catch (GooglePlayServicesNotAvailableException e) {
                    // TODO: Handle the error.
                }
            }
        });
@Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
       if (requestCode == PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE) {
           if (resultCode == RESULT_OK) {
              //Do your ok >from place< stuff here
           } else if (resultCode == PlaceAutocomplete.RESULT_ERROR) {
              //Handle your error >from place<
           } else if (resultCode == RESULT_CANCELED) {
               // The user canceled the operation.
           }
       } else if (requestCode == PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE) {
           if (resultCode == RESULT_OK) {
              //Do your ok >to place< stuff here
           } else if (resultCode == PlaceAutocomplete.RESULT_ERROR) {
              //Handle your error >to place<
           } else if (resultCode == RESULT_CANCELED) {
               // The user canceled the operation.
           }
        }
    }
```

Configuración de filtros de tipo de lugar para PlaceAutocomplete

En algunos casos, es posible que desee limitar los resultados que muestra **PlaceAutocompletar** a un país específico o tal vez mostrar solo las Regiones. Esto se puede lograr estableciendo un **AutocompleteFilter** en la intención. Por ejemplo, si deseo buscar solo lugares de tipo REGIÓN y que pertenezcan solo a la India, haría lo siguiente:

MainActivity.java

```
public class MainActivity extends AppComatActivity {

    private static final int PLACE_AUTOCOMPLETE_REQUEST_CODE = 1;
    private TextView selectedPlace;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        selectedPlace = (TextView) findViewById(R.id.selected_place);
        try {
            AutocompleteFilter typeFilter = new AutocompleteFilter.Builder()
                    .setTypeFilter(AutocompleteFilter.TYPE_FILTER_REGIONS)
                    .setCountry("IN")
```

```
                    .build();

            Intent intent =
                    new PlaceAutocomplete.IntentBuilder(PlaceAutocomplete.MODE_FULLSCREEN)
                            .setFilter(typeFilter)
                            .build(this);
            startActivityForResult(intent, PLACE_AUTOCOMPLETE_REQUEST_CODE);

        } catch (GooglePlayServicesRepairableException
                | GooglePlayServicesNotAvailableException e) {
            e.printStackTrace();
        }
}

protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == PLACE_AUTOCOMPLETE_REQUEST_CODE && resultCode == Activity.RESULT_OK) {
        final Place place = PlacePicker.getPlace(this, data);
        selectedPlace.setText(place.getName().toString().toUpperCase());
    } else {
        Toast.makeText(MainActivity.this, "Could not get location.",
Toast.LENGTH_SHORT).show();
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
          android:orientation="vertical"
          android:layout_width="match_parent"
          android:layout_height="match_parent">

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/selected_place"/>

</LinearLayout>
```

El **PlaceAutocomplete** se iniciará automáticamente y, a continuación, puede seleccionar un lugar de los resultados que solo serán del tipo **REGIÓN** y que solo pertenecerán al país especificado. La intención también se puede lanzar con el clic de un botón.

Lea API de Android Places en línea: https://riptutorial.com/es/android/topic/4111/api-de-android-places

Capítulo 21: API de conocimiento de Google

Observaciones

Recuerde, la API de instantáneas se utiliza para solicitar el estado actual, mientras que la API de cercado comprueba continuamente un estado específico y envía devoluciones de llamada cuando una aplicación no se está ejecutando.

En general, hay algunos pasos básicos para utilizar la API de instantáneas o la API de Fence:

- Obtenga una clave API de la Consola de Desarrolladores de Google
- Agregue los permisos necesarios y la clave API al manifiesto:

```
<!-- Not required for getting current headphone state -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<!-- Only required for actvity recognition -->
<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION"/>

<!-- Replace with your actual API key from console -->
<meta-data android:name="com.google.android.awareness.API_KEY"
           android:value="YOUR_API_KEY"/>

<!-- Required for Snapshot API only -->
<meta-data android:name="com.google.android.geo.API_KEY"
           android:value="YOUR_API_KEY"/>
```

- Inicialice el `GoogleApiClient` algún lugar, preferiblemente en el método onCreate () de su actividad.

```
GoogleApiClient client = new GoogleApiClient.Builder(context)
    .addApi(Awareness.API)
    .build();
client.connect();
```

- Llame a la API de su elección
- Parse el resultado

Una forma fácil de verificar el permiso de usuario necesario es un método como este:

```
private boolean isFineLocationGranted() {
    if (ActivityCompat.checkSelfPermission(context, Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED) {
        Log.e(getClass().getSimpleName(), "Fine location permission not granted!");
    }
}
```

Examples

Obtenga la actividad actual del usuario utilizando la API de instantáneas

Para solicitudes no constantes de una sola vez para la actividad física de un usuario, use la API de instantáneas:

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getDetectedActivity(client)
    .setResultCallback(new ResultCallback<DetectedActivityResult>() {
        @Override
        public void onResult(@NonNull DetectedActivityResult detectedActivityResult) {
            if (!detectedActivityResult.getStatus().isSuccess()) {
                Log.e(getClass().getSimpleName(), "Could not get the current activity.");
                return;
            }
            ActivityRecognitionResult result = detectedActivityResult
                .getActivityRecognitionResult();
            DetectedActivity probableActivity = result.getMostProbableActivity();
            Log.i(getClass().getSimpleName(), "Activity received : " +
                probableActivity.toString());
        }
    });
```

Obtener el estado de los auriculares con la API de instantáneas

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getHeadphoneState(client)
    .setResultCallback(new ResultCallback<HeadphoneStateResult>() {
        @Override
        public void onResult(@NonNull HeadphoneStateResult headphoneStateResult) {
            Log.i(TAG, "Headphone state connection state: " +
                headphoneStateResult.getHeadphoneState()
                .getState() == HeadphoneState.PLUGGED_IN));
        }
    });
```

Obtener ubicación actual utilizando API de instantáneas

```
// Remember to intialize your client as described in the Remarks section
Awareness.SnapshotApi.getLocation(client)
    .setResultCallback(new ResultCallback<LocationResult>() {
        @Override
        public void onResult(@NonNull LocationResult locationResult) {
            Location location = locationResult.getLocation();
            Log.i(getClass().getSimpleName(), "Coordinates: "location.getLatitude() + "," +
                location.getLongitude() + ", radius : " + location.getAccuracy());
        }
    });
```

Obtener lugares cercanos utilizando API de instantáneas

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getPlaces(client)
    .setResultCallback(new ResultCallback<PlacesResult>() {
        @Override
```

```
        public void onResult(@NonNull PlacesResult placesResult) {
            List<PlaceLikelihood> likelihoodList = placesResult.getPlaceLikelihoods();
            if (likelihoodList == null || likelihoodList.isEmpty()) {
                Log.e(getClass().getSimpleName(), "No likely places");
            }
        }
    });
```

En cuanto a obtener los datos en esos lugares, aquí hay algunas opciones:

```
Place place = placeLikelihood.getPlace();
String likelihood = placeLikelihood.getLikelihood();
Place place = likelihood.getPlace();
String placeName = place.getName();
String placeAddress = place.getAddress();
String placeCoords = place.getLatLng();
String locale = extractFromLocale(place.getLocale()));
```

Obtener el clima actual utilizando API de instantáneas

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getWeather(client)
    .setResultCallback(new ResultCallback<WeatherResult>() {
        @Override
        public void onResult(@NonNull WeatherResult weatherResult) {
            Weather weather = weatherResult.getWeather();
            if (weather == null) {
                Log.e(getClass().getSimpleName(), "No weather received");
            } else {
                Log.i(getClass().getSimpleName(), "Temperature is " +
                        weather.getTemperature(Weather.CELSIUS) + ", feels like " +
                        weather.getFeelsLikeTemperature(Weather.CELSIUS) +
                        ", humidity is " + weather.getHumidity());
            }
        }
    });
```

Obtén cambios en la actividad del usuario con Fence API

Si desea detectar cuándo su usuario comienza o finaliza una actividad como caminar, correr o cualquier otra actividad de la clase `DetectedActivityFence` , puede crear una cerca para la actividad que desea detectar y recibir una notificación cuando su usuario comience / Termina esta actividad. Al utilizar un `BroadcastReceiver` , obtendrá un `Intent` con datos que contienen la actividad:

```
// Your own action filter, like the ones used in the Manifest.
private static final String FENCE_RECEIVER_ACTION = BuildConfig.APPLICATION_ID +
    "FENCE_RECEIVER_ACTION";
private static final String FENCE_KEY = "walkingFenceKey";
private FenceReceiver mFenceReceiver;
private PendingIntent mPendingIntent;

// Make sure to initialize your client as described in the Remarks section.
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
    // etc.

    // The 0 is a standard Activity request code that can be changed to your needs.
    mPendingIntent = PendingIntent.getBroadcast(this, 0,
        new Intent(FENCE_RECEIVER_ACTION), 0);
    registerReceiver(mFenceReceiver, new IntentFilter(FENCE_RECEIVER_ACTION));

    // Create the fence.
    AwarenessFence fence = DetectedActivityFence.during(DetectedActivityFence.WALKING);
    // Register the fence to receive callbacks.
    Awareness.FenceApi.updateFences(client, new FenceUpdateRequest.Builder()
        .addFence(FENCE_KEY, fence, mPendingIntent)
        .build())
        .setResultCallback(new ResultCallback<Status>() {
            @Override
            public void onResult(@NonNull Status status) {
                if (status.isSuccess()) {
                    Log.i(FENCE_KEY, "Successfully registered.");
                } else {
                    Log.e(FENCE_KEY, "Could not be registered: " + status);
                }
            }
        });
    }
}
```

Ahora puede recibir la intención con un `BroadcastReceiver` para obtener devoluciones de llamada cuando el usuario cambia la actividad:

```
public class FenceReceiver extends BroadcastReceiver {

    private static final String TAG = "FenceReceiver";

    @Override
    public void onReceive(Context context, Intent intent) {
        // Get the fence state
        FenceState fenceState = FenceState.extract(intent);

        switch (fenceState.getCurrentState()) {
            case FenceState.TRUE:
                Log.i(TAG, "User is walking");
                break;
            case FenceState.FALSE:
                Log.i(TAG, "User is not walking");
                break;
            case FenceState.UNKNOWN:
                Log.i(TAG, "User is doing something unknown");
                break;
        }
    }
}
```

Obtenga cambios para la ubicación dentro de un cierto rango usando la API de Fence

Si desea detectar cuándo su usuario ingresa a una ubicación específica, puede crear una cerca

para la ubicación específica con el radio que desee y recibir una notificación cuando su usuario ingrese o salga de la ubicación.

```
// Your own action filter, like the ones used in the Manifest
private static final String FENCE_RECEIVER_ACTION = BuildConfig.APPLICATION_ID +
    "FENCE_RECEIVER_ACTION";
private static final String FENCE_KEY = "locationFenceKey";
private FenceReceiver mFenceReceiver;
private PendingIntent mPendingIntent;

// Make sure to initialize your client as described in the Remarks section
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // etc

    // The 0 is a standard Activity request code that can be changed for your needs
    mPendingIntent = PendingIntent.getBroadcast(this, 0,
        new Intent(FENCE_RECEIVER_ACTION), 0);
    registerReceiver(mFenceReceiver, new IntentFilter(FENCE_RECEIVER_ACTION));

    // Create the fence
    AwarenessFence fence = LocationFence.entering(48.136334, 11.581660, 25);
    // Register the fence to receive callbacks.
    Awareness.FenceApi.updateFences(client, new FenceUpdateRequest.Builder()
        .addFence(FENCE_KEY, fence, mPendingIntent)
        .build())
        .setResultCallback(new ResultCallback<Status>() {
            @Override
            public void onResult(@NonNull Status status) {
                if (status.isSuccess()) {
                    Log.i(FENCE_KEY, "Successfully registered.");
                } else {
                    Log.e(FENCE_KEY, "Could not be registered: " + status);
                }
            }
        });
    }
}
```

Ahora cree un BroadcastReciver para recibir actualizaciones en el estado del usuario:

```
public class FenceReceiver extends BroadcastReceiver {

    private static final String TAG = "FenceReceiver";

    @Override
    public void onReceive(Context context, Intent intent) {
        // Get the fence state
        FenceState fenceState = FenceState.extract(intent);

        switch (fenceState.getCurrentState()) {
            case FenceState.TRUE:
                Log.i(TAG, "User is in location");
                break;
            case FenceState.FALSE:
                Log.i(TAG, "User is not in location");
                break;
            case FenceState.UNKNOWN:
                Log.i(TAG, "User is doing something unknown");
```

```
                break;
            }
        }
    }
```

Lea API de conocimiento de Google en línea: https://riptutorial.com/es/android/topic/3361/api-de-conocimiento-de-google

Capítulo 22: API de Google Drive

Introducción

Google Drive es un servicio de alojamiento de archivos creado por **Google** . Proporciona un servicio de almacenamiento de archivos y le permite al usuario cargar archivos en la nube y también compartirlos con otras personas. Al utilizar la API de Google Drive, podemos sincronizar archivos entre una computadora o dispositivo móvil y Google Drive Cloud.

Observaciones

Legal

Si utiliza la API de Android de Google Drive en su aplicación, debe incluir el texto de atribución de Google Play Services como parte de una sección de "Avisos legales" en su aplicación.

Se recomienda que incluya avisos legales como un elemento de menú independiente o como parte de un elemento de menú "Acerca de".

Puede realizar una llamada a `GooglePlayServicesUtil.getOpenSourceSoftwareLicenseInfo()` para obtener el texto de atribución en tiempo de ejecución.

Examples

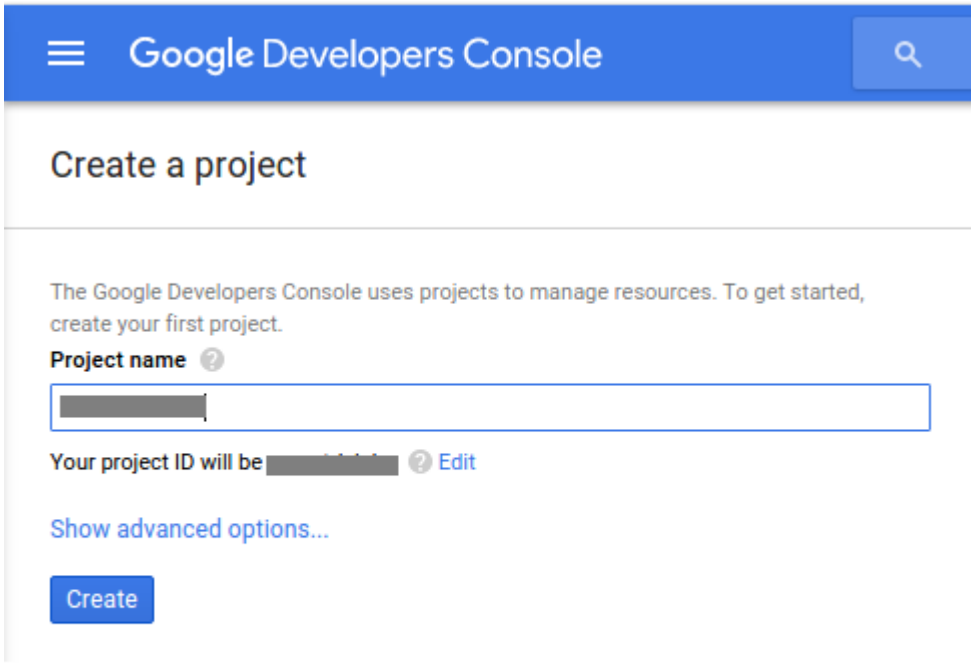
Integrar Google Drive en Android

Crear un nuevo proyecto en la consola de desarrolladores de Google

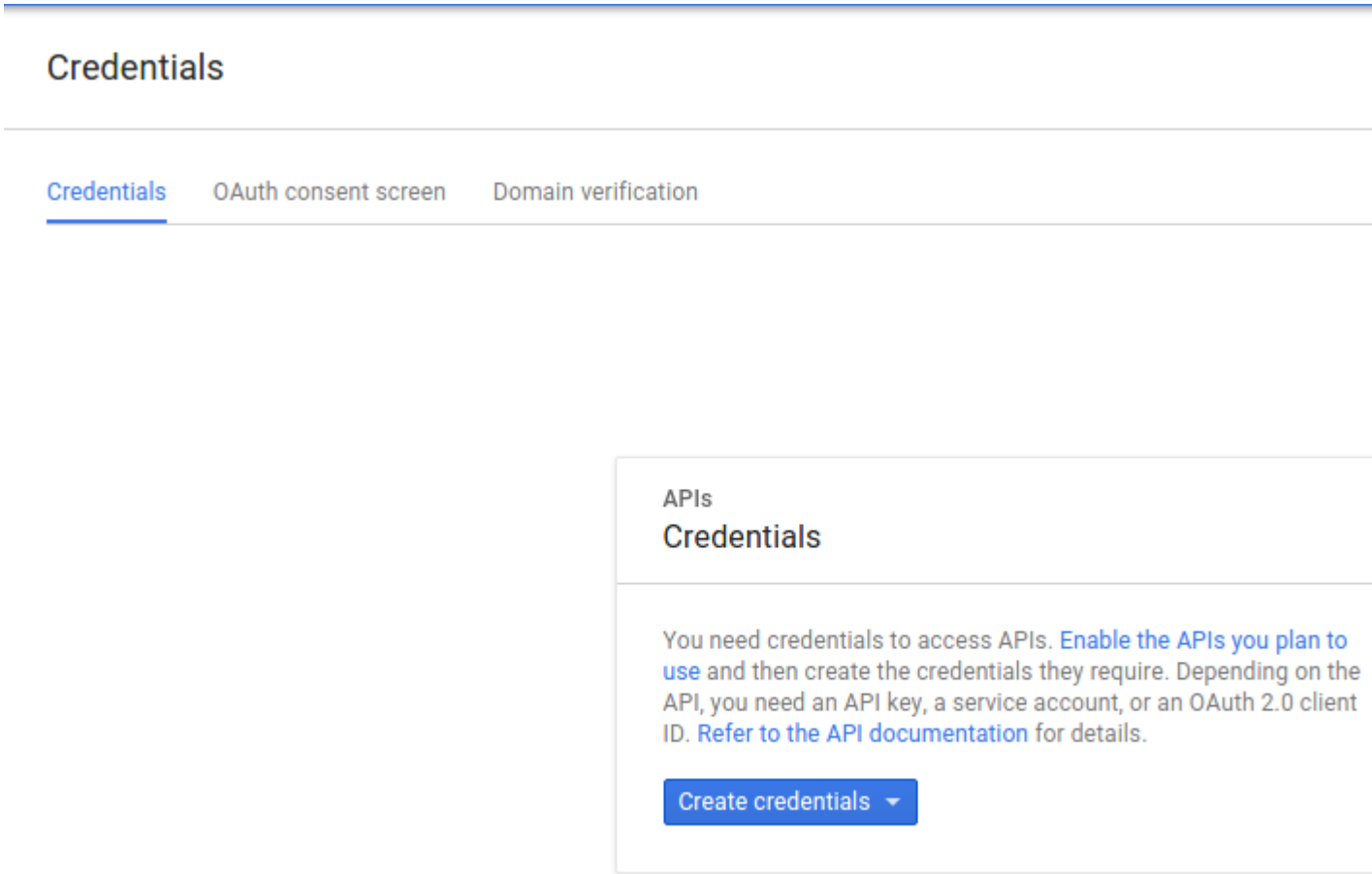
Para integrar la aplicación de Android con Google Drive, cree las credenciales del proyecto en la Consola de desarrolladores de Google. Por lo tanto, necesitamos crear un proyecto en la consola de desarrolladores de Google.

Para crear un proyecto en la Consola de desarrollador de Google, siga estos pasos:

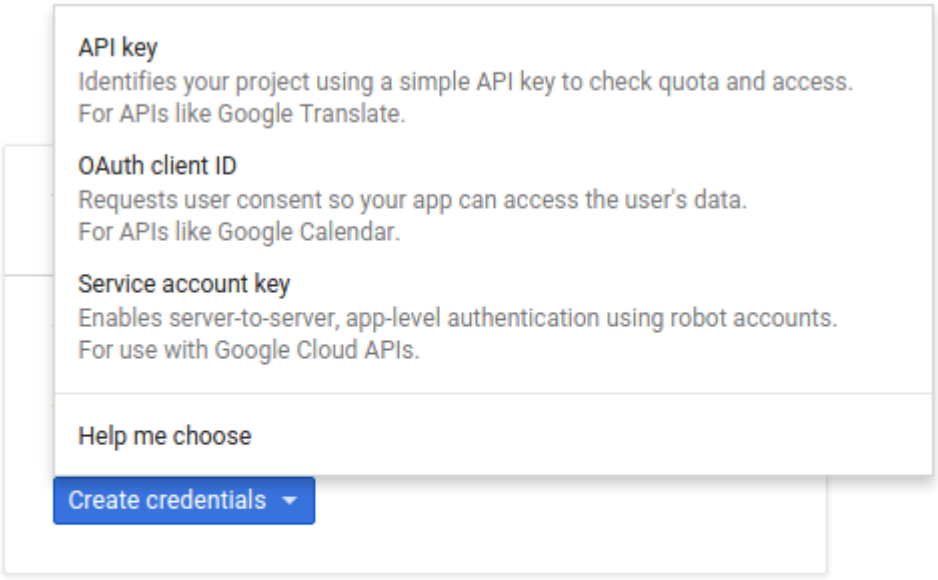
- Ir a la consola de desarrolladores de Google para Android. Rellene el **nombre del proyecto** en el campo de entrada y haga clic en el botón **Crear** para crear un nuevo proyecto en Google consola de desarrollador.



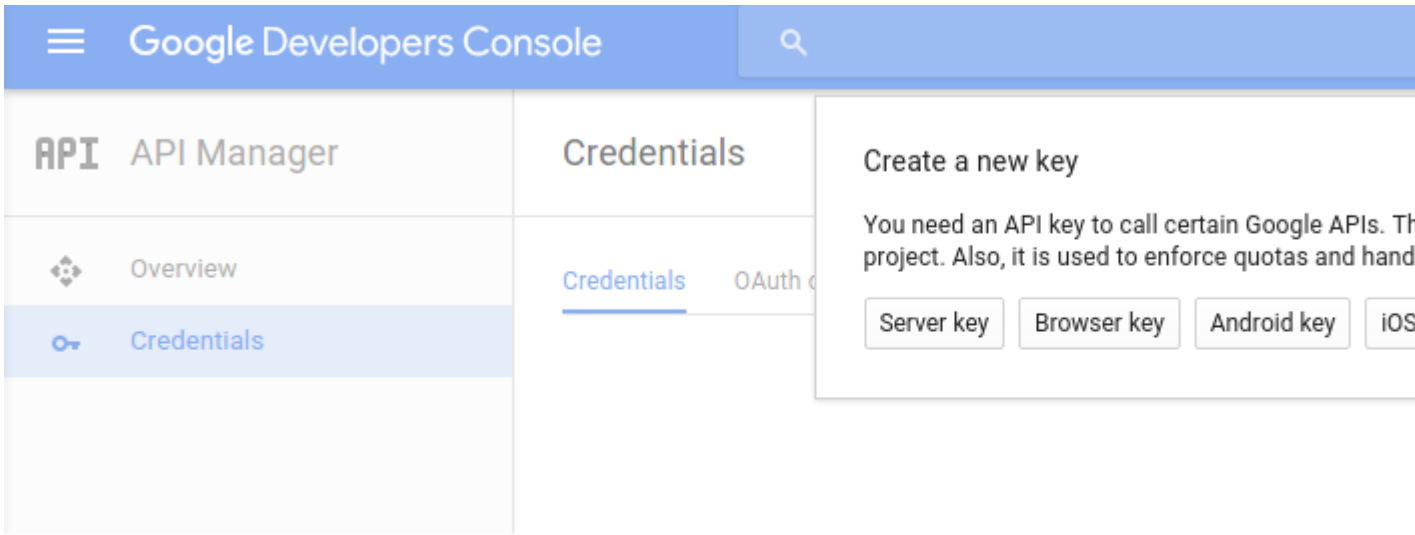
- Necesitamos crear credenciales para acceder a la API. Por lo tanto, haga clic en el botón **Crear credenciales** .



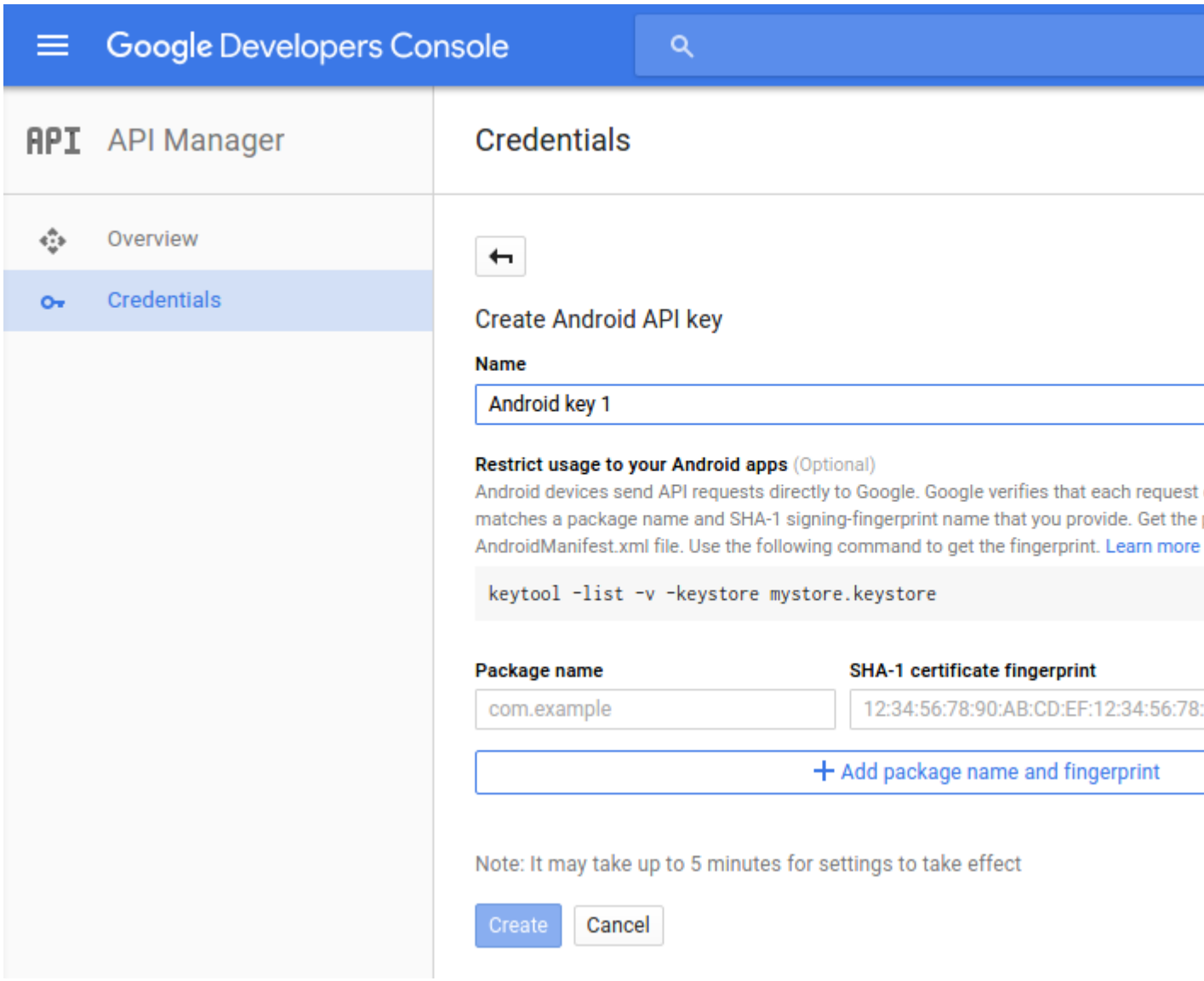
- Ahora, se abrirá una ventana emergente. Haga clic en la opción **Clave de API** en la lista para crear la clave de API.



- Necesitamos una clave API para llamar a las API de Google para Android. Por lo tanto, haga clic en la **tecla Android** para identificar su proyecto Android.



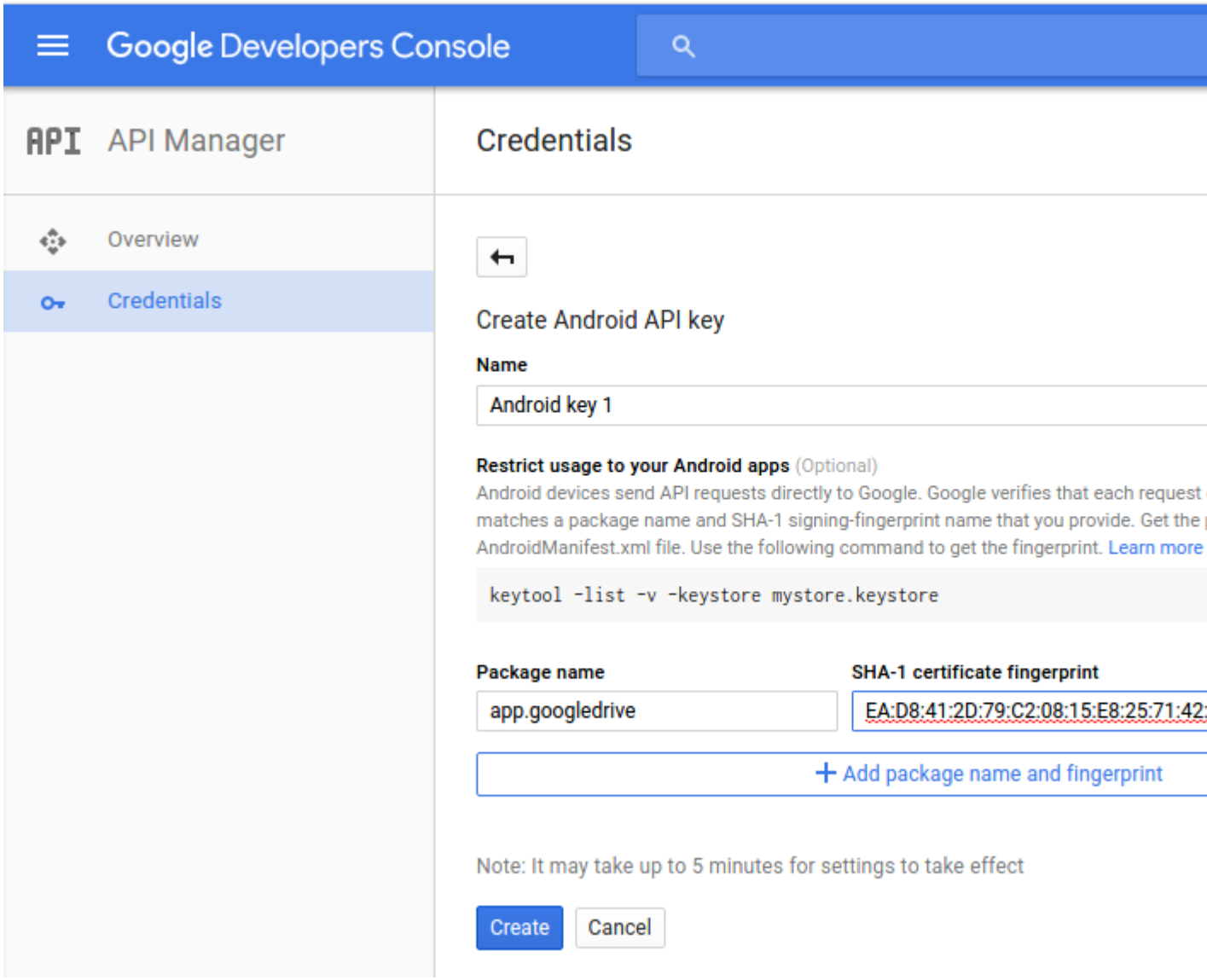
- A continuación, debemos agregar el Nombre del paquete del proyecto de Android y la **huella dactilar SHA-1** en los campos de entrada para crear la clave API.



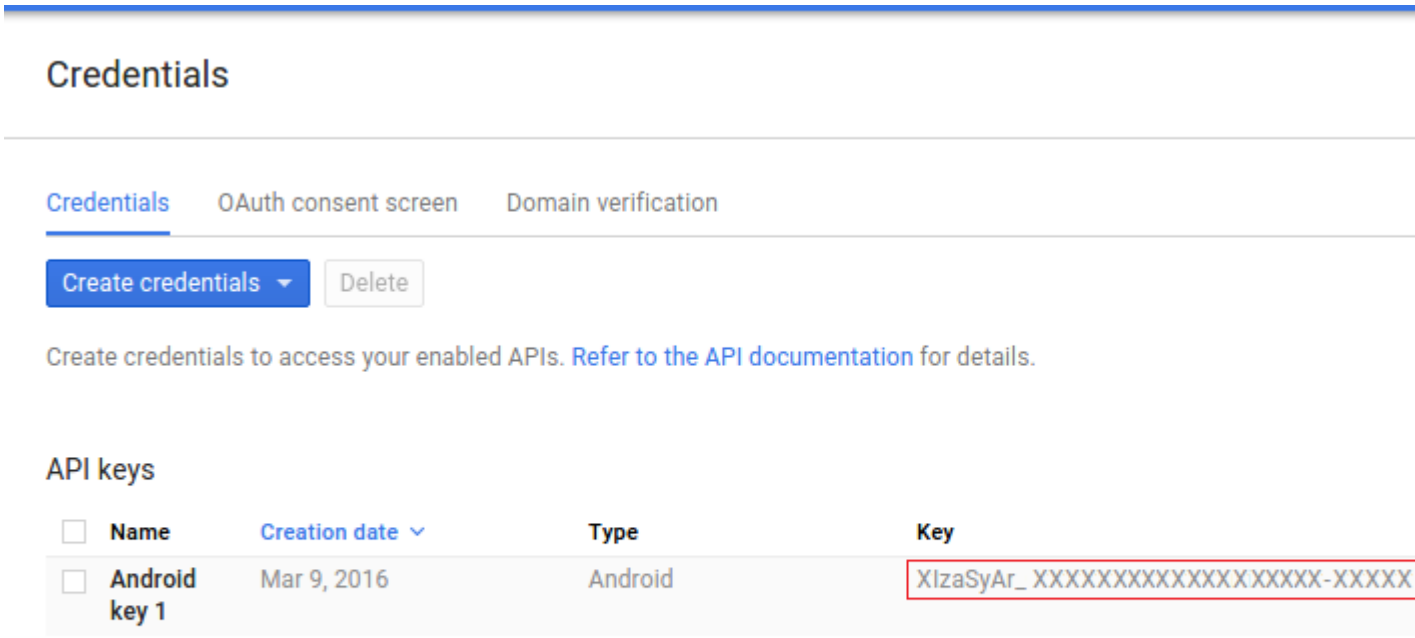
- Necesitamos generar la **huella dactilar SHA-1** . Por lo tanto, abra su terminal y ejecute la **utilidad Keytool** para obtener la huella digital SHA1. Mientras ejecuta la utilidad Keytool, debe proporcionar la **contraseña del almacén de claves** . La clave de desarrollo predeterminada de la herramienta keytool es **"android"** . `keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -list -v`

```
[redacted]@[redacted]:~$ keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -list -v
Enter keystore password:
Alias name: androiddebugkey
Creation date: 18 Jul, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 3adbdb98
Valid from: Sat Jul 18 09:32:08 IST 2015 until: Mon Jul 10 09:32:08 IST 2045
Certificate fingerprints:
         MD5:  77:C7:A9:6A:30:0F:43:B9:84:E0:61:0F:B2:B6:22:74
         SHA1: EA:D8:41:2D:79:C2:08:15:E8:25:71:42:3F:0E:51:A5:52:4C:EF:40
         SHA256: A2:12:5A:18:E2:F3:FE:8B:93:E8:03:0C:12:3A:52:8D:B5:B0:70:32:C
F3:A7:C3:47:F0:9E:B6:8E:AF:33:68
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: D3 8F C7 0C 95 B4 DA 73   6B 67 99 5A A3 C0 05 4A  .......skg.Z...J
0010: 93 BE 25 4F                                        ..%O
]
]
```

- Ahora, agregue el **nombre del paquete** y la **huella digital SHA-1** en los campos de entrada en la página de credenciales. Finalmente, haga clic en el botón crear para crear la clave API.



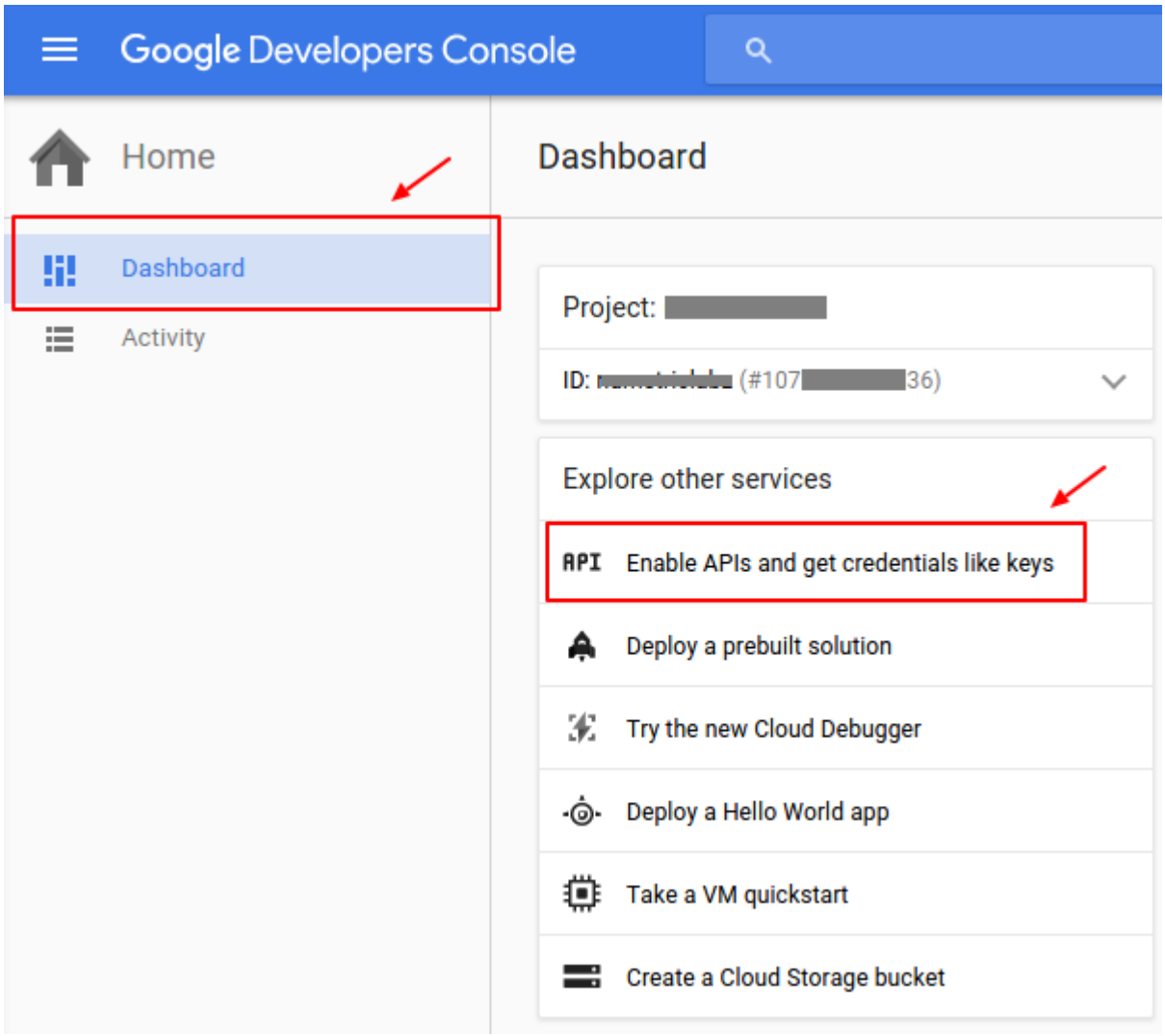
- Esto creará la clave API para Android. Utilizaremos esta clave API para integrar la aplicación de Android con Google Drive.



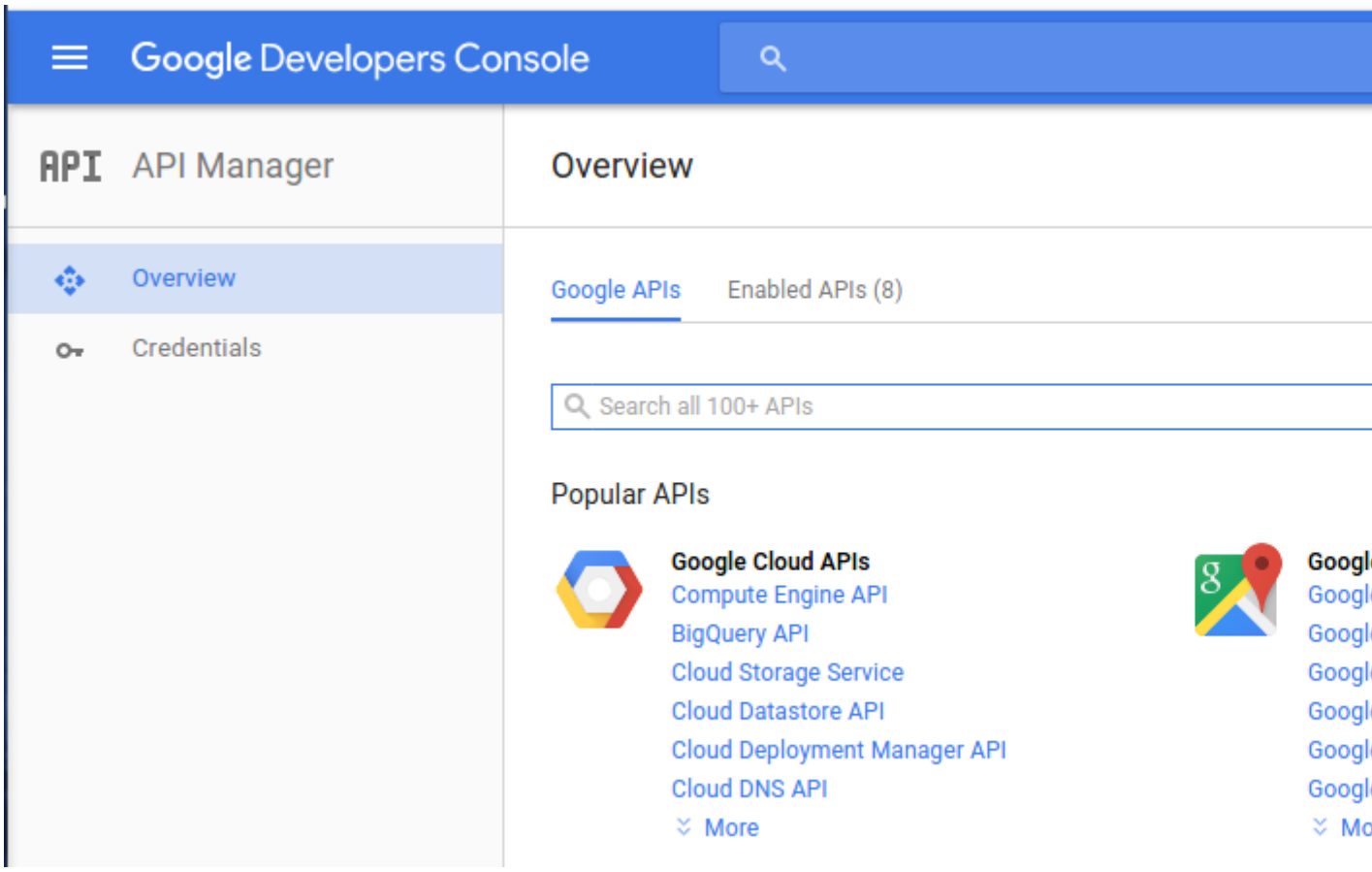
Habilitar API de Google Drive

Necesitamos habilitar Google Drive Api para acceder a los archivos almacenados en Google Drive desde la aplicación de Android. Para habilitar la API de Google Drive, siga los siguientes pasos:

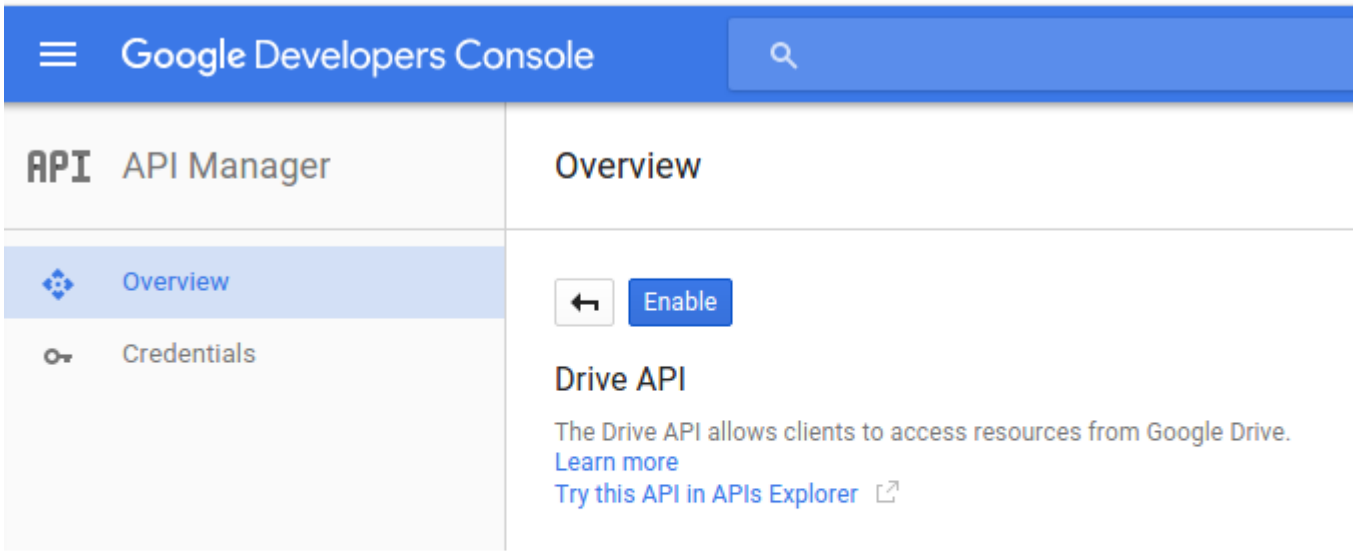
- Vaya al panel de la consola de Google Developer y haga clic en **Habilitar APIs para obtener credenciales como claves,** luego verá las populares API de Google.



- Haga clic en el enlace de **Drive API** para abrir la página de información general de Google Drive API.



- Haga clic en el botón Habilitar para habilitar la API de Google drive. Permite el acceso del cliente a Google Drive.



Añadir permiso de Internet

La aplicación necesita acceso a **Internet** archivos de Google Drive. Use el siguiente código para

configurar los permisos de Internet en el archivo AndroidManifest.xml:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Añadir servicios de Google Play

Utilizaremos la **API de servicios de Google Play,** que incluye la **API de Android de Google Drive** . Por lo tanto, necesitamos configurar los servicios de Google Play SDK en la aplicación de Android. Abra su `build.gradle` (módulo de aplicación) y agregue el SDK de servicios de Google Play como dependencias.

```
dependencies {
  ....
    compile 'com.google.android.gms:play-services:<latest_version>'
  ....
}
```

Añadir clave de API en el archivo de manifiesto

Para utilizar la API de Google en la aplicación de Android, debemos agregar la clave de la API y la versión del servicio Google Play en el archivo AndroidManifest.xml. Agregue las etiquetas de metadatos correctas dentro de la etiqueta del archivo AndroidManifest.xml.

Conectar y Autorizar la API de Android de Google Drive

Necesitamos autenticar y conectar la **API de Android de Google Drive** con la aplicación de Android. La autorización de **Google Drive Android API** es manejada por **GoogleApiClient** . Usaremos **GoogleApiClient** dentro del método **onResume ()** .

```
/**
 * Called when the activity will start interacting with the user.
 * At this point your activity is at the top of the activity stack,
 * with user input going to it.
 */
@Override
protected void onResume() {
   super.onResume();
   if (mGoogleApiClient == null) {

       /**
        * Create the API client and bind it to an instance variable.
        * We use this instance as the callback for connection and connection failures.
        * Since no account name is passed, the user is prompted to choose.
        */
         mGoogleApiClient = new GoogleApiClient.Builder(this)
                 .addApi(Drive.API)
                 .addScope(Drive.SCOPE_FILE)
                 .addConnectionCallbacks(this)
                 .addOnConnectionFailedListener(this)
                 .build();
        }

        mGoogleApiClient.connect();
    }
```

Desconecta Google Deive Android API

Cuando la actividad se detenga, desconectaremos la conexión de la API de Android de Google Drive con la aplicación de Android llamando al método **disconnect ()** dentro del método **onStop () de la actividad** .

```
@Override
protected void onStop() {
    super.onStop();
    if (mGoogleApiClient != null) {

         // disconnect Google Android Drive API connection.
         mGoogleApiClient.disconnect();
    }
    super.onPause();
}
```

Implementar devoluciones de llamada de conexión y escucha de conexión fallida

Implementaremos las devoluciones de llamada de conexión y la escucha de conexión fallida del cliente API de Google en el archivo MainActivity.java para conocer el estado de la conexión del cliente API de Google. Estos escuchas proporcionan el **método onConnected (), onConnectionFailed (), onConnectionSuspended ()** para manejar los problemas de conexión entre la aplicación y la unidad.

Si el usuario ha autorizado la aplicación, se invoca el método **onConnected ()** . Si el usuario no ha autorizado la aplicación, se invoca el método **onConnectionFailed ()** y se muestra un cuadro de diálogo que **indica** que su aplicación no está autorizada para acceder a Google Drive. En caso de que se suspenda la conexión, se llama al método **onConnectionSuspended ()** .

Debe implementar **ConnectionCallbacks** y **OnConnectionFailedListener** en su actividad. Usa el siguiente código en tu archivo Java.

```
@Override
    public void onConnectionFailed(ConnectionResult result) {

        // Called whenever the API client fails to connect.
        Log.i(TAG, "GoogleApiClient connection failed:" + result.toString());

        if (!result.hasResolution()) {

            // show the localized error dialog.
            GoogleApiAvailability.getInstance().getErrorDialog(this, result.getErrorCode(),
0).show();
            return;
        }

        /**
         *  The failure has a resolution. Resolve it.
         *  Called typically when the app is not yet authorized, and an  authorization
         *  dialog is displayed to the user.
         */

        try {
```

```
            result.startResolutionForResult(this, REQUEST_CODE_RESOLUTION);

        } catch (SendIntentException e) {

            Log.e(TAG, "Exception while starting resolution activity", e);
        }
    }

   /**
    * It invoked when Google API client connected
    * @param connectionHint
    */
    @Override
    public void onConnected(Bundle connectionHint) {

        Toast.makeText(getApplicationContext(), "Connected", Toast.LENGTH_LONG).show();
    }

   /**
    * It invoked when connection suspended
    * @param cause
    */
    @Override
    public void onConnectionSuspended(int cause) {

        Log.i(TAG, "GoogleApiClient connection suspended");
    }
```

Crear un archivo en Google Drive

Añadiremos un archivo en Google Drive. Usaremos el método `createFile()` de un objeto `Drive` para crear un archivo mediante programación en Google Drive. En este ejemplo, estamos agregando un nuevo archivo de texto en la carpeta raíz del usuario. Cuando se agrega un archivo, debemos especificar el conjunto inicial de metadatos, el contenido del archivo y la carpeta principal.

Necesitamos crear un método de devolución de llamada `CreateMyFile()` y, dentro de este método, usar el objeto `Drive` para recuperar un recurso `DriveContents` . Luego pasamos el cliente API al objeto `Drive` y llamamos al método de devolución de llamada `driveContentsCallback` para manejar el resultado de `DriveContents` .

Un recurso `DriveContents` contiene una copia temporal del flujo binario del archivo que solo está disponible para la aplicación.

```
public void CreateMyFile(){
    fileOperation = true;
    // Create new contents resource.
    Drive.DriveApi.newDriveContents(mGoogleApiClient)
                  .setResultCallback(driveContentsCallback);
}
```

Controlador de resultados de DriveContents

El manejo de la respuesta requiere verificar si la llamada fue exitosa o no. Si la llamada fue exitosa, podemos recuperar el recurso `DriveContents` .

Crearemos un manejador de resultados de `DriveContents` . Dentro de este método, llamamos al método `CreateFileOnGoogleDrive()` y pasamos el resultado de `DriveContentsResult` :

```
/**
 * This is the Result result handler of Drive contents.
 * This callback method calls the CreateFileOnGoogleDrive() method.
 */
final ResultCallback<DriveContentsResult> driveContentsCallback =
         new ResultCallback<DriveContentsResult>() {
            @Override
            public void onResult(DriveContentsResult result) {
                if (result.getStatus().isSuccess()) {
                    if (fileOperation == true){
                        CreateFileOnGoogleDrive(result);
                    }
                }
            }
        };
```

Crear archivo programáticamente

Para crear archivos, necesitamos usar un objeto `MetadataChangeSet` . Al usar este objeto, establecemos el título (nombre del archivo) y el tipo de archivo. Además, debemos usar el método `createFile()` de la clase `DriveFolder` y pasar la API del cliente de Google, el objeto `MetaDataChangeSet` y `driveContents` para crear un archivo. Llamamos a la devolución de llamada del manejador de resultados para manejar el resultado del archivo creado.

Usamos el siguiente código para crear un nuevo archivo de texto en la carpeta raíz del usuario:

```
/**
 * Create a file in the root folder using a MetadataChangeSet object.
 * @param result
 */
public void CreateFileOnGoogleDrive(DriveContentsResult result){

    final DriveContents driveContents = result.getDriveContents();

    // Perform I/O off the UI thread.
    new Thread() {
        @Override
        public void run() {
            // Write content to DriveContents.
            OutputStream outputStream = driveContents.getOutputStream();
            Writer writer = new OutputStreamWriter(outputStream);
            try {
                writer.write("Hello Christlin!");
                writer.close();
            } catch (IOException e) {
                Log.e(TAG, e.getMessage());
            }
```

```
            MetadataChangeSet changeSet = new MetadataChangeSet.Builder()
                    .setTitle("My First Drive File")
                    .setMimeType("text/plain")
                    .setStarred(true).build();

            // Create a file in the root folder.
            Drive.DriveApi.getRootFolder(mGoogleApiClient)
                    .createFile(mGoogleApiClient, changeSet, driveContents)
                    setResultCallback(fileCallback);
        }
    }.start();
}
```

Manejar el resultado del archivo creado

El siguiente código creará un método de devolución de llamada para manejar el resultado del archivo creado:

```
/**
 * Handle result of Created file
 */
final private ResultCallback<DriveFolder.DriveFileResult> fileCallback = new
        ResultCallback<DriveFolder.DriveFileResult>() {
            @Override
            public void onResult(DriveFolder.DriveFileResult result) {
                if (result.getStatus().isSuccess()) {
                    Toast.makeText(getApplicationContext(), "file created: "+
                                result.getDriveFile().getDriveId(), Toast.LENGTH_LONG).show();
                }
                return;
            }
        };
```

Lea API de Google Drive en línea: https://riptutorial.com/es/android/topic/10646/api-de-google-drive

Capítulo 23: API de Google Maps v2 para Android

Parámetros

Parámetro	Detalles
Mapa de Google	`GoogleMap` es un objeto que se recibe en un evento `onMapReady()`
MarkerOptions	`MarkerOptions` es la clase de constructor de un `Marker` , y se utiliza para agregar un marcador a un mapa.

Observaciones

Requerimientos

1. Google Play Services SDK instalado.
2. Una cuenta de Google Console.
3. Una clave de API de Google Maps obtenida en la consola de Google.

Examples

Actividad predeterminada de Google Map

Este código de actividad proporcionará una funcionalidad básica para incluir un mapa de Google usando un SupportMapFragment.

La API de Google Maps V2 incluye una nueva forma de cargar mapas.

Las actividades ahora tienen que implementar la interfaz **OnMapReadyCallBack** , que viene con una **anulación del** método **onMapReady ()** que se ejecuta cada vez que ejecutamos **SupportMapFragment** . **getMapAsync (OnMapReadyCallback)** ; y la llamada se completa con éxito.

Los mapas utilizan **marcadores** , **polígonos** y **líneas poligonales** para mostrar información interactiva al usuario.

MapsActivity.java:

```
public class MapsActivity extends AppCompatActivity implements OnMapReadyCallback {

    private GoogleMap mMap;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney, Australia, and move the camera.
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}
```

Observe que el código anterior infla un diseño, que tiene un SupportMapFragment anidado dentro del diseño del contenedor, definido con un ID de `R.id.map` . El archivo de diseño se muestra a continuación:

activity_maps.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:map="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/map"
        tools:context="com.example.app.MapsActivity"
        android:name="com.google.android.gms.maps.SupportMapFragment"/>

</LinearLayout>
```

Estilos de mapas de Google personalizados

Estilo de mapa

Google Maps viene con un conjunto de diferentes estilos para ser aplicados, usando este código:

```
// Sets the map type to be "hybrid"
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

Los diferentes estilos de mapas son:

Normal

```
map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
```

Mapa de carreteras típico. Se muestran caminos, algunas características hechas por el hombre e importantes características naturales como los ríos. Las etiquetas de carreteras y de características también son visibles.



Híbrido

```
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

Datos de fotografías satelitales con mapas de carreteras añadidos. Las etiquetas de carreteras y de características también son visibles.



Satélite

```
map.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
```

Datos de la fotografía del satélite. Las etiquetas de carreteras y características no son visibles.



Terreno

```
map.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

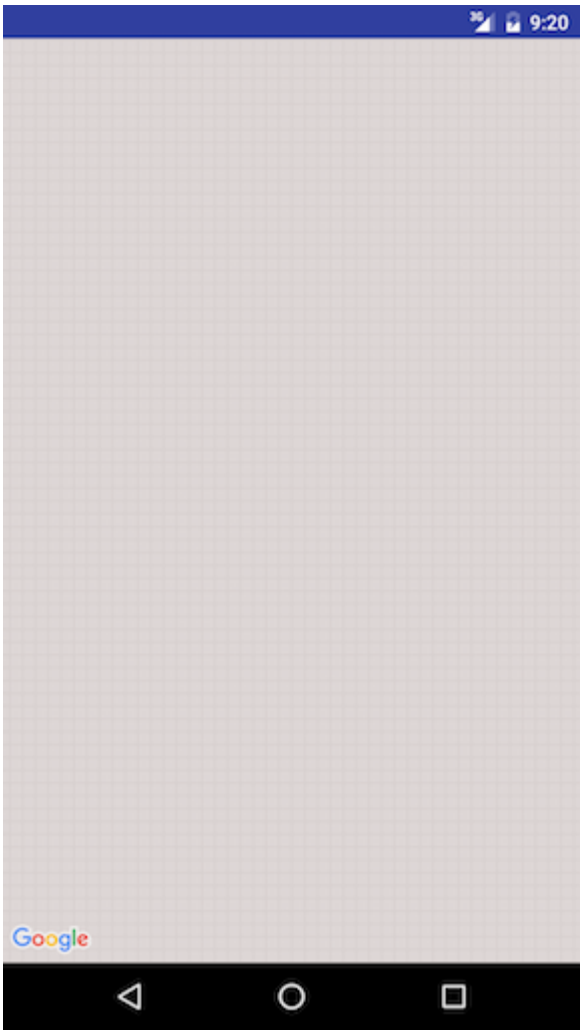
Datos topográficos. El mapa incluye colores, líneas de contorno y etiquetas, y sombreado en perspectiva. Algunas carreteras y etiquetas también son visibles.



Ninguna

```
map.setMapType(GoogleMap.MAP_TYPE_NONE);
```

No hay azulejos. El mapa se representará como una cuadrícula vacía sin mosaicos cargados.



OTRAS OPCIONES DE ESTILO

Mapas interiores

En niveles de zoom altos, el mapa mostrará planos de planta para espacios interiores. Estos se denominan mapas interiores y se muestran solo para los tipos de mapa "normal" y "satélite".

para habilitar o deshabilitar los mapas interiores, así es como se hace:

```
GoogleMap.setIndoorEnabled(true).
GoogleMap.setIndoorEnabled(false).
```

Podemos añadir estilos personalizados a los mapas.

En el método onMapReady agrega el siguiente fragmento de código

```
mMap = googleMap;
    try {
        // Customise the styling of the base map using a JSON object defined
        // in a raw resource file.
        boolean success = mMap.setMapStyle(
                MapStyleOptions.loadRawResourceStyle(
                        MapsActivity.this, R.raw.style_json));
```

```
        if (!success) {
            Log.e(TAG, "Style parsing failed.");
        }
    } catch (Resources.NotFoundException e) {
        Log.e(TAG, "Can't find style.", e);
    }
```

en la carpeta *res* cree un nombre de carpeta sin *formato* y agregue el archivo de estilos json. Ejemplo de archivo style.json

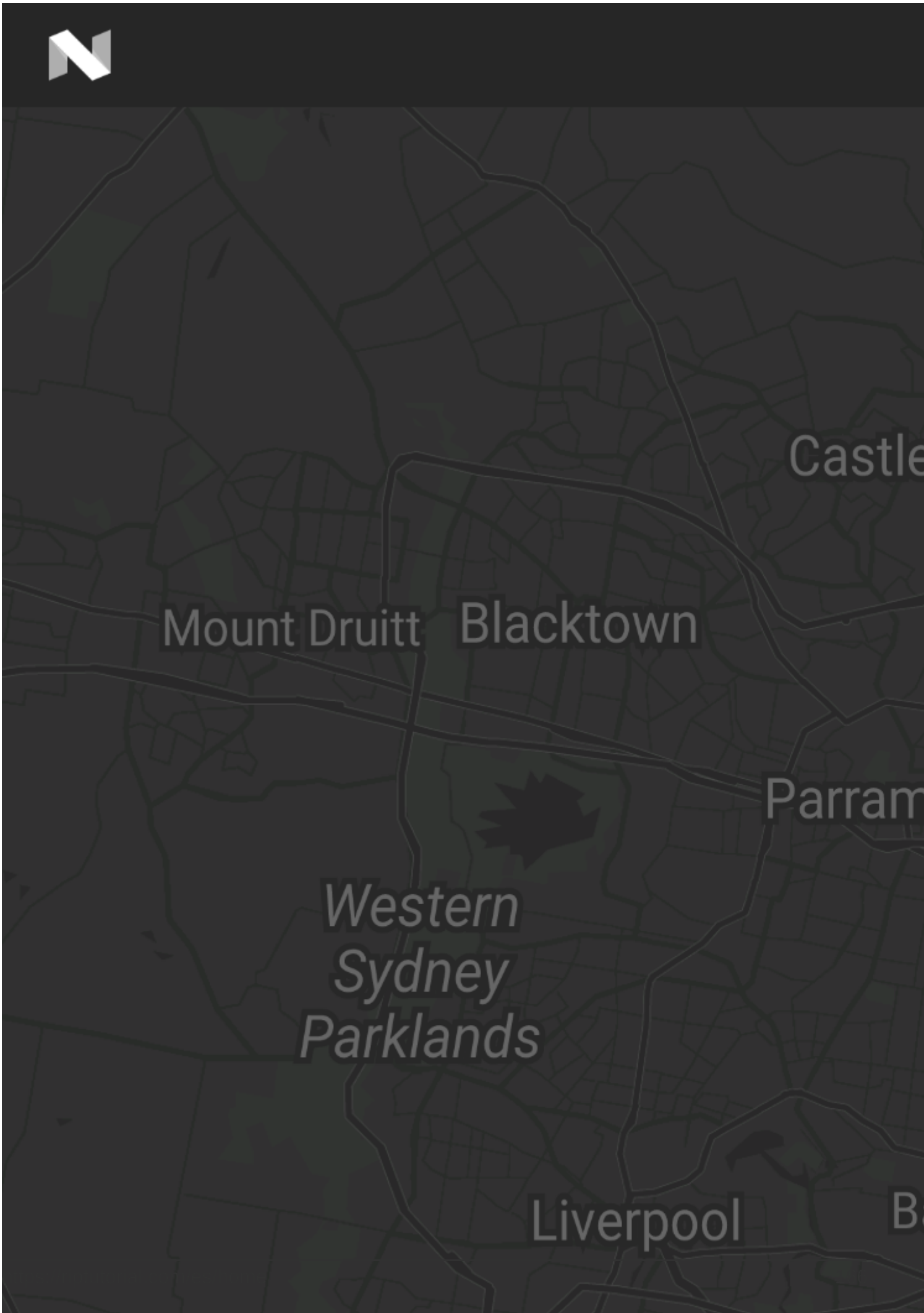
```
  [
{
  "featureType": "all",
  "elementType": "geometry",
  "stylers": [
    {
      "color": "#242f3e"
    }
  ]
},
{
  "featureType": "all",
  "elementType": "labels.text.stroke",
  "stylers": [
    {
      "lightness": -80
    }
  ]
},
{
  "featureType": "administrative",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#746855"
    }
  ]
},
{
  "featureType": "administrative.locality",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#d59563"
    }
  ]
},
{
  "featureType": "poi",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#d59563"
    }
  ]
},
{
  "featureType": "poi.park",
  "elementType": "geometry",
  "stylers": [
```

```
    {
      "color": "#263c3f"
    }
  ]
},
{
  "featureType": "poi.park",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#6b9a76"
    }
  ]
},
{
  "featureType": "road",
  "elementType": "geometry.fill",
  "stylers": [
    {
      "color": "#2b3544"
    }
  ]
},
{
  "featureType": "road",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#9ca5b3"
    }
  ]
},
{
  "featureType": "road.arterial",
  "elementType": "geometry.fill",
  "stylers": [
    {
      "color": "#38414e"
    }
  ]
},
{
  "featureType": "road.arterial",
  "elementType": "geometry.stroke",
  "stylers": [
    {
      "color": "#212a37"
    }
  ]
},
{
  "featureType": "road.highway",
  "elementType": "geometry.fill",
  "stylers": [
    {
      "color": "#746855"
    }
  ]
},
{
  "featureType": "road.highway",
```

```
    "elementType": "geometry.stroke",
    "stylers": [
      {
        "color": "#1f2835"
      }
    ]
  },
  {
    "featureType": "road.highway",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#f3d19c"
      }
    ]
  },
  {
    "featureType": "road.local",
    "elementType": "geometry.fill",
    "stylers": [
      {
        "color": "#38414e"
      }
    ]
  },
  {
    "featureType": "road.local",
    "elementType": "geometry.stroke",
    "stylers": [
      {
        "color": "#212a37"
      }
    ]
  },
  {
    "featureType": "transit",
    "elementType": "geometry",
    "stylers": [
      {
        "color": "#2f3948"
      }
    ]
  },
  {
    "featureType": "transit.station",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#d59563"
      }
    ]
  },
  {
    "featureType": "water",
    "elementType": "geometry",
    "stylers": [
      {
        "color": "#17263c"
      }
    ]
  },
```

```
  {
    "featureType": "water",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#515c6d"
      }
    ]
  },
  {
    "featureType": "water",
    "elementType": "labels.text.stroke",
    "stylers": [
      {
        "lightness": -20
      }
    ]
  }
]
```

Para generar los estilos del archivo json pulsa este enlace.


Castle
Mount Druitt
Blacktown
Parram
Western
Sydney
Parklands
Liverpool

Objects, podemos hacerlo de esta manera.

La clase titular de `MyLocation` :

```
public class MyLocation {
  LatLng latLng;
  String title;
  String snippet;
}
```

Aquí hay un método que tomaría una lista de objetos `MyLocation` y colocaría un marcador para cada uno:

```
private void LocationsLoaded(List<MyLocation> locations){

 for (MyLocation myLoc : locations){
    mMap.addMarker(new MarkerOptions()
     .position(myLoc.latLng)
     .title(myLoc.title)
     .snippet(myLoc.snippet)
     .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_MAGENTA));
 }
}
```

Nota: A los efectos de este ejemplo, `mMap` es una variable miembro de la clase de la Actividad, donde la asignamos a la referencia de mapa recibida en la `onMapReady()` .

MapView: incrustar un mapa de Google en un diseño existente

Es posible tratar un GoogleMap como una vista de Android si hacemos uso de la clase MapView proporcionada. Su uso es muy similar a MapFragment.

En su diseño use MapView de la siguiente manera:

```
<com.google.android.gms.maps.MapView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    <!--
    map:mapType="0" Specifies a change to the initial map type
    map:zOrderOnTop="true"  Control whether the map view's surface is placed on top of its
window
    map:useVieLifecycle="true" When using a MapFragment, this flag specifies whether the
lifecycle of the map should be tied to the fragment's view or the fragment itself
    map:uiCompass="true" Enables or disables the compass
    map:uiRotateGestures="true" Sets the preference for whether rotate gestures should be
enabled or disabled
    map:uiScrollGestures="true" Sets the preference for whether scroll gestures should be
enabled or disabled
    map:uiTiltGestures="true" Sets the preference for whether tilt gestures should be enabled
or disabled
    map:uiZoomGestures="true" Sets the preference for whether zoom gestures should be enabled
or disabled
```

```
    map:uiZoomControls="true" Enables or disables the zoom controls
    map:liteMode="true" Specifies whether the map should be created in lite mode
    map:uiMapToolbar="true" Specifies whether the mapToolbar should be enabled
    map:ambientEnabled="true" Specifies whether ambient-mode styling should be enabled
    map:cameraMinZoomPreference="0.0" Specifies a preferred lower bound for camera zoom
    map:cameraMaxZoomPreference="1.0" Specifies a preferred upper bound for camera zoom -->
    />
```

Su actividad necesita implementar la interfaz OnMapReadyCallback para funcionar:

```
/**
* This shows how to create a simple activity with a raw MapView and add a marker to it. This
* requires forwarding all the important lifecycle methods onto MapView.
*/
public class RawMapViewDemoActivity extends AppCompatActivity implements OnMapReadyCallback {

    private MapView mMapView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.raw_mapview_demo);

        mMapView = (MapView) findViewById(R.id.map);
        mMapView.onCreate(savedInstanceState);

        mMapView.getMapAsync(this);
    }

    @Override
    protected void onResume() {
        super.onResume();
        mMapView.onResume();
    }

    @Override
    public void onMapReady(GoogleMap map) {
        map.addMarker(new MarkerOptions().position(new LatLng(0, 0)).title("Marker"));
    }

    @Override
    protected void onPause() {
        mMapView.onPause();
        super.onPause();
    }

    @Override
    protected void onDestroy() {
        mMapView.onDestroy();
        super.onDestroy();
    }

    @Override
    public void onLowMemory() {
        super.onLowMemory();
        mMapView.onLowMemory();
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
```

```
        super.onSaveInstanceState(outState);
        mMapView.onSaveInstanceState(outState);
    }
}
```

Mostrar ubicación actual en un mapa de Google

Aquí hay una clase de actividad completa que coloca un marcador en la ubicación actual y también mueve la cámara a la posición actual.

Hay algunas cosas que suceden en secuencia aquí:

- Comprobar el permiso de ubicación
- Una vez que se conceda el permiso de ubicación, llame a `setMyLocationEnabled()` , genere el GoogleApiClient y conéctelo
- Una vez que el GoogleApiClient esté conectado, solicite actualizaciones de ubicación

```
public class MapLocationActivity extends AppCompatActivity
        implements OnMapReadyCallback,
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        LocationListener {

    GoogleMap mGoogleMap;
    SupportMapFragment mapFrag;
    LocationRequest mLocationRequest;
    GoogleApiClient mGoogleApiClient;
    Location mLastLocation;
    Marker mCurrLocationMarker;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        getSupportActionBar().setTitle("Map Location Activity");

        mapFrag = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
        mapFrag.getMapAsync(this);
    }

    @Override
    public void onPause() {
        super.onPause();

        //stop location updates when Activity is no longer active
        if (mGoogleApiClient != null) {
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
        }
    }

    @Override
    public void onMapReady(GoogleMap googleMap)
    {
        mGoogleMap=googleMap;
        mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

```
        //Initialize Google Play Services
        if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)
                    == PackageManager.PERMISSION_GRANTED) {
                //Location Permission already granted
                buildGoogleApiClient();
                mGoogleMap.setMyLocationEnabled(true);
            } else {
                //Request Location Permission
                checkLocationPermission();
            }
        }
        else {
            buildGoogleApiClient();
            mGoogleMap.setMyLocationEnabled(true);
        }
    }

    protected synchronized void buildGoogleApiClient() {
        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
        mGoogleApiClient.connect();
    }

    @Override
    public void onConnected(Bundle bundle) {
        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(1000);
        mLocationRequest.setFastestInterval(1000);
        mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
        }
    }

    @Override
    public void onConnectionSuspended(int i) {}

    @Override
    public void onConnectionFailed(ConnectionResult connectionResult) {}

    @Override
    public void onLocationChanged(Location location)
    {
        mLastLocation = location;
        if (mCurrLocationMarker != null) {
            mCurrLocationMarker.remove();
        }

        //Place current location marker
        LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.position(latLng);
```

```
        markerOptions.title("Current Position");

markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_MAGENTA));

        mCurrLocationMarker = mGoogleMap.addMarker(markerOptions);

        //move map camera
        mGoogleMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mGoogleMap.animateCamera(CameraUpdateFactory.zoomTo(11));

        //stop location updates
        if (mGoogleApiClient != null) {
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
        }
    }

    public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;
    private void checkLocationPermission() {
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED) {

            // Should we show an explanation?
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)) {

                // Show an explanation to the user *asynchronously* -- don't block
                // this thread waiting for the user's response! After the user
                // sees the explanation, try again to request the permission.
                new AlertDialog.Builder(this)
                        .setTitle("Location Permission Needed")
                        .setMessage("This app needs the Location permission, please accept to
use location functionality")
                        .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialogInterface, int i) {
                                //Prompt the user once explanation has been shown
                                ActivityCompat.requestPermissions(MapLocationActivity.this,
                                        new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                                        MY_PERMISSIONS_REQUEST_LOCATION );
                            }
                        })
                        .create()
                        .show();


            } else {
                // No explanation needed, we can request the permission.
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                        MY_PERMISSIONS_REQUEST_LOCATION );
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           String permissions[], int[] grantResults) {
        switch (requestCode) {
            case MY_PERMISSIONS_REQUEST_LOCATION: {
                // If request is cancelled, the result arrays are empty.
```

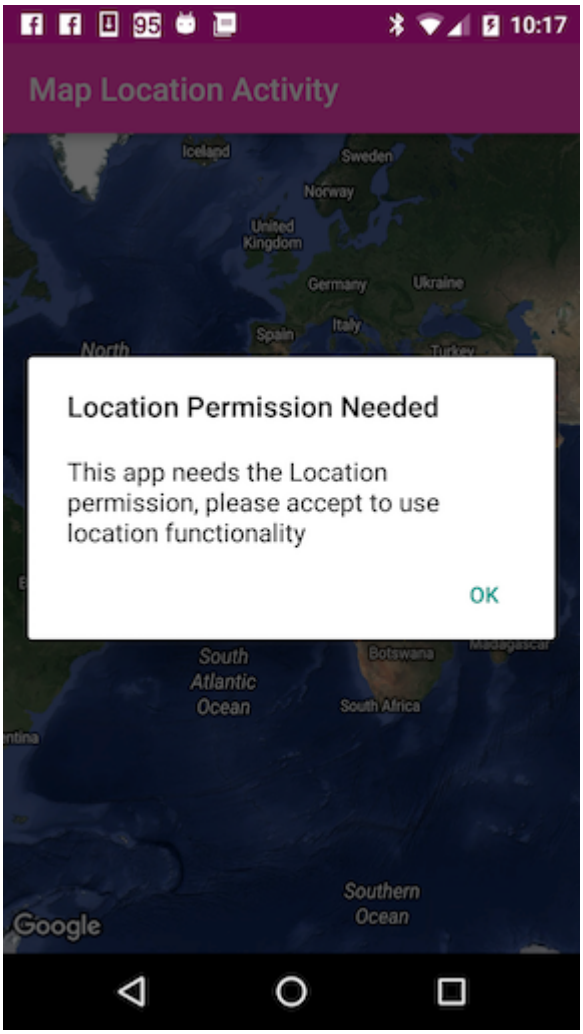
```
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                    // permission was granted, yay! Do the
                    // location-related task you need to do.
                    if (ContextCompat.checkSelfPermission(this,
                            Manifest.permission.ACCESS_FINE_LOCATION)
                            == PackageManager.PERMISSION_GRANTED) {

                        if (mGoogleApiClient == null) {
                            buildGoogleApiClient();
                        }
                        mGoogleMap.setMyLocationEnabled(true);
                    }

                } else {

                    // permission denied, boo! Disable the
                    // functionality that depends on this permission.
                    Toast.makeText(this, "permission denied", Toast.LENGTH_LONG).show();
                }
                return;
            }

            // other 'case' lines to check for other
            // permissions this app might request
        }
    }

}
```

activity_main.xml:

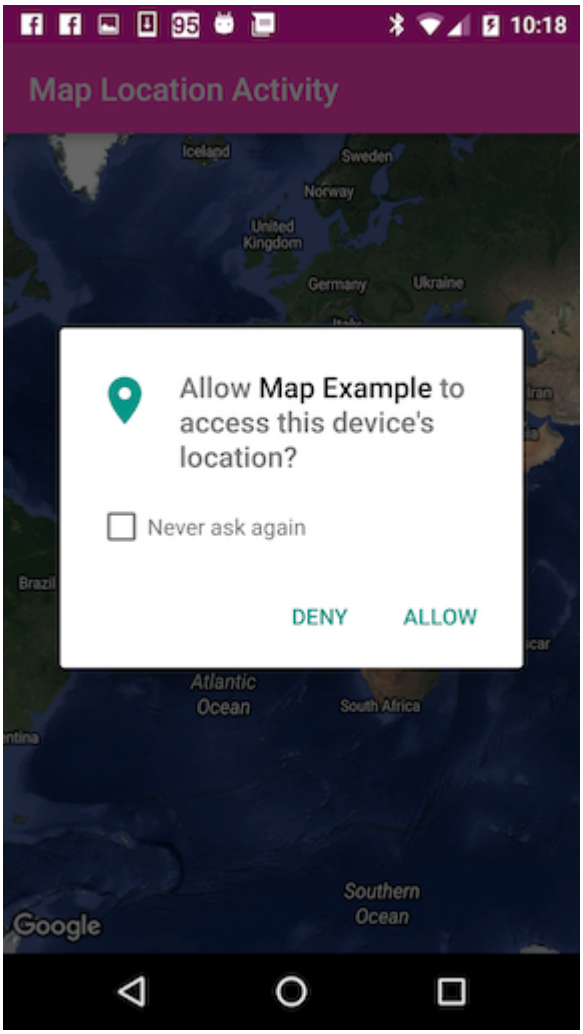
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:map="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/map"
        tools:context="com.example.app.MapLocationActivity"
        android:name="com.google.android.gms.maps.SupportMapFragment"/>

</LinearLayout>
```

Resultado:

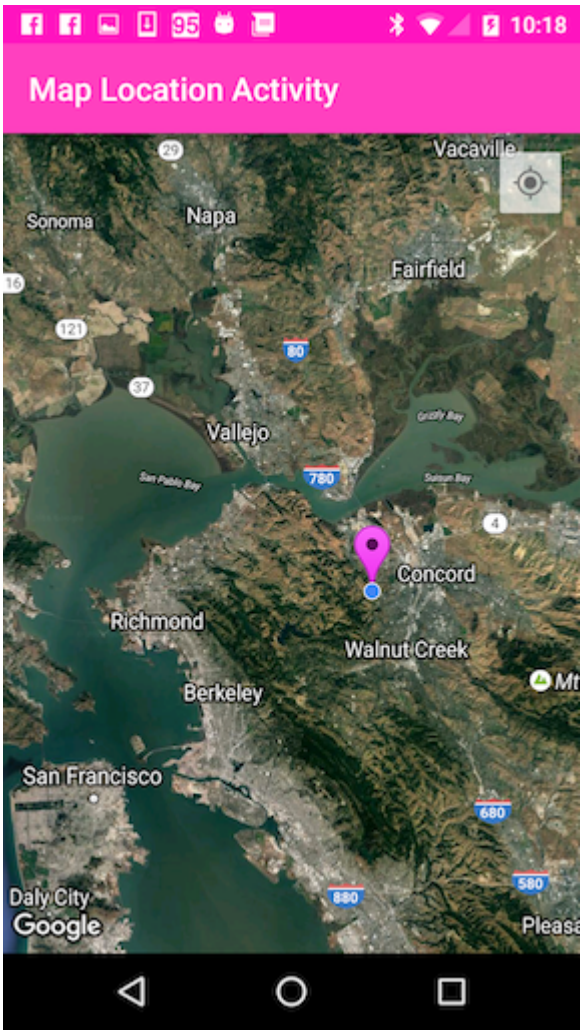
Muestre la explicación si es necesario en Marshmallow y Nougat usando un AlertDialog (este caso ocurre cuando el usuario ha denegado previamente una solicitud de permiso, ha otorgado el permiso y luego lo ha revocado en la configuración):



Solicite al usuario el permiso de ubicación en Marshmallow y Nougat llamando a `ActivityCompat.requestPermissions()` :



Mueva la cámara a la ubicación actual y coloque el Marcador cuando se otorgue el permiso de Ubicación:



Obtención de la huella digital SH1 de su archivo de almacén de claves de certificado

Para obtener una clave API de Google Maps para su certificado, debe proporcionar a la consola API la huella digital SH1 de su almacén de claves de depuración / lanzamiento.

Puede obtener el almacén de claves utilizando el programa **keytool de JDK** como se describe aquí en la documentación.

Otro enfoque es obtener la huella digital programáticamente ejecutando este fragmento con su aplicación firmada con el certificado de depuración / liberación e imprimiendo el hash en el registro.

```
PackageInfo info;
try {
    info = getPackageManager().getPackageInfo("com.package.name",
PackageManager.GET_SIGNATURES);
    for (Signature signature : info.signatures) {
        MessageDigest md;
        md = MessageDigest.getInstance("SHA");
        md.update(signature.toByteArray());
        String hash= new String(Base64.encode(md.digest(), 0));
        Log.e("hash", hash);
    }
```

```
} catch (NameNotFoundException e1) {
    Log.e("name not found", e1.toString());
} catch (NoSuchAlgorithmException e) {
    Log.e("no such an algorithm", e.toString());
} catch (Exception e) {
    Log.e("exception", e.toString());
}
```

No inicie Google Maps cuando se hace clic en el mapa (modo lite)

Cuando se muestra un Google Map en modo lite, al hacer clic en un mapa se abrirá la aplicación Google Maps. Para deshabilitar esta funcionalidad, debe llamar a `setClickable(false)` en el `MapView` , *por ejemplo* :

```
final MapView mapView = (MapView)view.findViewById(R.id.map);
mapView.setClickable(false);
```

UISettings

Usando `UISettings` , se puede modificar la apariencia de Google Map.

Aquí hay un ejemplo de algunas configuraciones comunes:

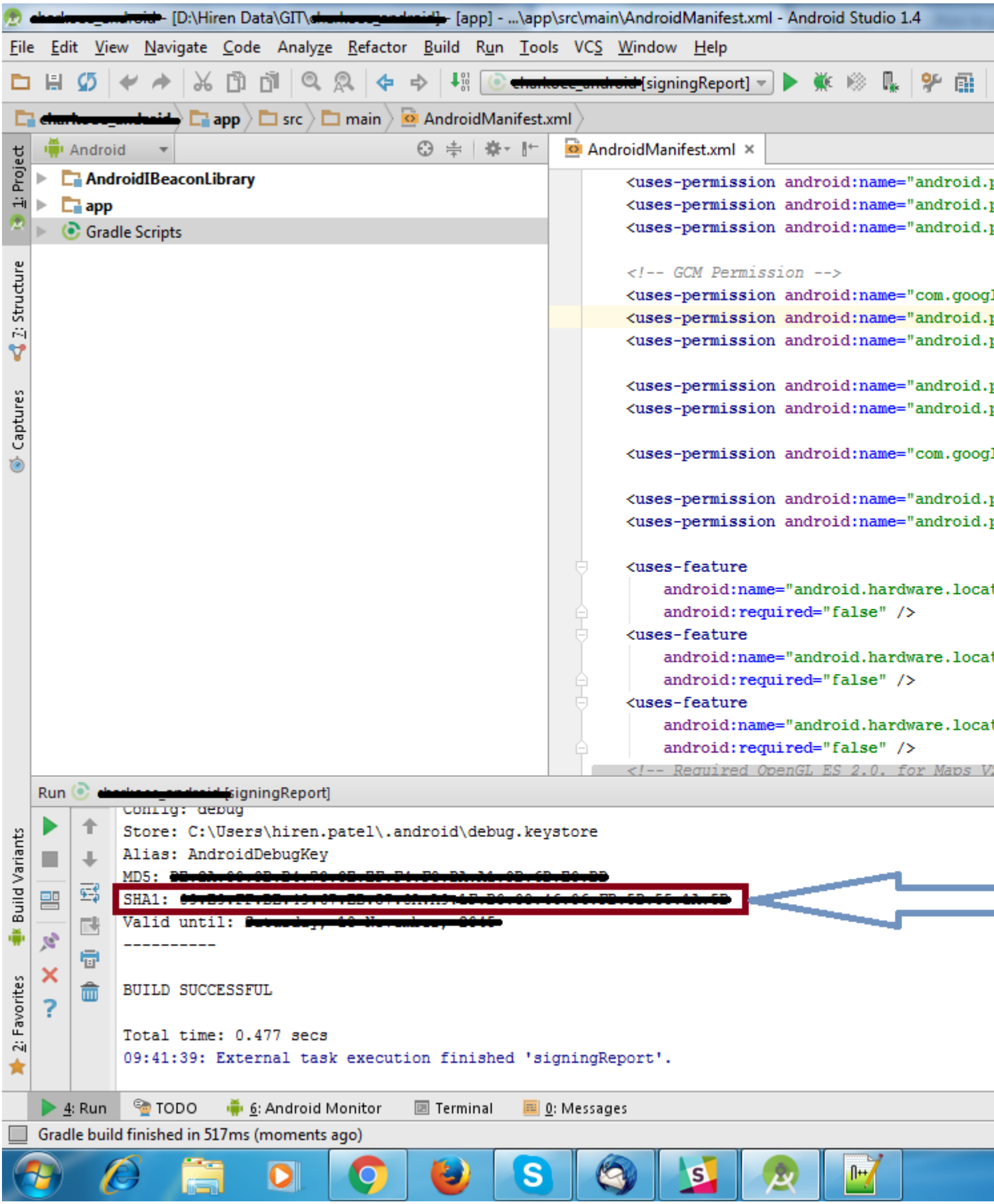
```
    mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
    mGoogleMap.getUiSettings().setMapToolbarEnabled(true);
    mGoogleMap.getUiSettings().setZoomControlsEnabled(true);
    mGoogleMap.getUiSettings().setCompassEnabled(true);
```

Resultado:



Obtener debug SHA1 huella digital

1. Abrir Android Studio
2. Abre tu proyecto
3. Haga clic en Gradle (en el panel lateral derecho, verá la **barra de Gradle**)
4. Haga clic en Actualizar (Haga clic en Actualizar desde la **barra** de **Gradle** , verá los scripts de la **lista de** Gradle de su proyecto)
5. Haga clic en Su proyecto (**Lista de** formularios de su nombre de proyecto (raíz))
6. Haga clic en Tareas
7. Haga clic en android
8. Haga doble clic en signarReport (obtendrá **SHA1** y **MD5** en la **barra de ejecución**)



InfoWindow Click Listener

Este es un ejemplo de cómo definir una acción diferente para cada evento de clic en la ventana

de InfoWindow.

Use un HashMap en el que la identificación del marcador sea la clave, y el valor sea la acción correspondiente que se debe realizar cuando se hace clic en la ventana de información.

Luego, use un `OnInfoWindowClickListener` para manejar el evento de un usuario que haga clic en la ventana de información, y use el HashMap para determinar qué acción tomar.

En este sencillo ejemplo, abriremos una Actividad diferente en función de la Ventana de Información del Marcador en la que se hizo clic.

Declare el HashMap como una variable de instancia de la Actividad o Fragmento:

```
 //Declare HashMap to store mapping of marker to Activity
HashMap<String, String> markerMap = new HashMap<String, String>();
```

Luego, cada vez que agregue un Marcador, cree una entrada en el HashMap con el ID de Marcador y la acción que debe tomar cuando se hace clic en InfoWindow.

Por ejemplo, agregando dos marcadores y definiendo una acción a realizar para cada uno:

```
Marker markerOne = googleMap.addMarker(new MarkerOptions().position(latLng1)
        .title("Marker One")
        .snippet("This is Marker One");
String idOne = markerOne.getId();
markerMap.put(idOne, "action_one");

Marker markerTwo = googleMap.addMarker(new MarkerOptions().position(latLng2)
        .title("Marker Two")
        .snippet("This is Marker Two");
String idTwo = markerTwo.getId();
markerMap.put(idTwo, "action_two");
```

En el detector de clics de InfoWindow, obtenga la acción del HashMap y abra la Actividad correspondiente en función de la acción del Marcador:

```
 mGoogleMap.setOnInfoWindowClickListener(new GoogleMap.OnInfoWindowClickListener() {
  @Override
  public void onInfoWindowClick(Marker marker) {

    String actionId = markerMap.get(marker.getId());

    if (actionId.equals("action_one")) {
      Intent i = new Intent(MainActivity.this, ActivityOne.class);
      startActivity(i);
    } else if (actionId.equals("action_two")) {
      Intent i = new Intent(MainActivity.this, ActivityTwo.class);
      startActivity(i);
    }
  }
});
```

Nota Si el código está en un fragmento, reemplace MainActivity.this con getActivity ().

Cambiar Offset

Al cambiar los valores de mappoint *x* e *y* según sea necesario, puede cambiar la posición de desplazamiento de google map, de forma predeterminada estará en el centro de la vista del mapa. Llama a continuación el método donde quieres cambiarlo! Es mejor usarlo dentro de `onLocationChanged` como `changeOffsetCenter(location.getLatitude(),location.getLongitude());`

```
 public void changeOffsetCenter(double latitude,double longitude) {
            Point mappoint = mGoogleMap.getProjection().toScreenLocation(new LatLng(latitude,
longitude));
            mappoint.set(mappoint.x, mappoint.y-100); // change these values as you need ,
just hard coded a value if you want you can give it based on a ratio like using DisplayMetrics
as well

mGoogleMap.animateCamera(CameraUpdateFactory.newLatLng(mGoogleMap.getProjection().fromScreenLocation(ma

        }
```

Lea API de Google Maps v2 para Android en línea: https://riptutorial.com/es/android/topic/170/api-de-google-maps-v2-para-android

Capítulo 24: API de la cámara 2

Parámetros

Parámetro	Detalles
`CameraCaptureSession`	Una sesión de captura configurada para un `CameraDevice` , usado para capturar imágenes de la cámara o reprocesar imágenes capturadas desde la cámara en la misma sesión anterior
`CameraDevice`	Una representación de una sola cámara conectada a un dispositivo Android
`CameraCharacteristics`	Las propiedades que describen un CameraDevice. Estas propiedades son fijas para un CameraDevice determinado y se pueden consultar a través de la interfaz de CameraManager con `getCameraCharacteristics(String)`
`CameraManager`	Un administrador de servicios del sistema para detectar, caracterizar y conectarse a `CameraDevices` . Puede obtener una instancia de esta clase llamando a `Context.getSystemService()`
`CaptureRequest`	Un paquete inmutable de configuraciones y salidas necesarias para capturar una sola imagen desde el dispositivo de la cámara. Contiene la configuración del hardware de captura (sensor, lente, flash), el proceso de procesamiento, los algoritmos de control y los buffers de salida. También contiene la lista de Superficies de destino para enviar datos de imagen para esta captura. Puede crearse utilizando una instancia de `CaptureRequest.Builder` , obtenida llamando a `createCaptureRequest(int)`
`CaptureResult`	El subconjunto de los resultados de una sola captura de imagen del sensor de imagen. Contiene un subconjunto de la configuración final para el hardware de captura (sensor, lente, flash), la tubería de procesamiento, los algoritmos de control y los buffers de salida. Es producido por un `CameraDevice` después de procesar una `CaptureRequest`

Observaciones

- Las API de Camera2 están disponibles en API 21+ (Lollipop y más allá)
- Incluso si un dispositivo Android tiene una ROM 21+ oficialmente, no hay garantía de que implemente las API de Camera2, el fabricante tiene la responsabilidad de implementarlo o no (por ejemplo, LG G2 tiene soporte oficial de Lollipop, pero no tiene API de Camera2)
- Con Camera2, la cámara ("Camera1") está en desuso

- Con gran poder viene una gran responsabilidad: es más fácil estropearlo cuando se utilizan estas API.
- Recuerde, si solo desea tomar una foto en su aplicación, y simplemente obtenerla, **no** necesita implementar Camera2, puede abrir la aplicación de la cámara del dispositivo a través de un Intent y volver a recibirla.

Examples

Vista previa de la cámara principal en un TextureView

En este caso, compilando contra la API 23, los permisos también se manejan.

Debe agregar en el Manifiesto el siguiente permiso (donde sea que esté usando el nivel de API):

```
<uses-permission android:name="android.permission.CAMERA"/>
```

Estamos a punto de crear una actividad (Camera2Activity.java) que llena un `TextureView` con la vista previa de la cámara del dispositivo.

La Actividad que vamos a usar es una AppCompatActivity típica:

```
public class Camera2Activity extends AppCompatActivity {
```

Atributos (Es posible que deba leer el ejemplo completo para comprenderlo)

El `MAX_PREVIEW_SIZE` garantizado por Camera2 API es 1920x1080

```
private static final int MAX_PREVIEW_WIDTH = 1920;
private static final int MAX_PREVIEW_HEIGHT = 1080;
```

`TextureView.SurfaceTextureListener` maneja varios eventos del ciclo de vida en un `TextureView` . En este caso, estamos escuchando esos eventos. Cuando la SurfaceTexture está lista, inicializamos la cámara. Cuando cambia el tamaño, configuramos la vista previa que viene de la cámara en consecuencia

```
private final TextureView.SurfaceTextureListener mSurfaceTextureListener
        = new TextureView.SurfaceTextureListener() {

    @Override
    public void onSurfaceTextureAvailable(SurfaceTexture texture, int width, int height) {
        openCamera(width, height);
    }

    @Override
    public void onSurfaceTextureSizeChanged(SurfaceTexture texture, int width, int height) {
        configureTransform(width, height);
    }

    @Override
    public boolean onSurfaceTextureDestroyed(SurfaceTexture texture) {
        return true;
```

```
    }

    @Override
    public void onSurfaceTextureUpdated(SurfaceTexture texture) {
    }

};
```

Un `CameraDevice` representa la cámara de un dispositivo físico. En este atributo, guardamos el ID del `CameraDevice` actual

```
private String mCameraId;
```

Esta es la vista (`TextureView`) que `TextureView` para "dibujar" la vista previa de la cámara

```
private TextureView mTextureView;
```

La `CameraCaptureSession` para la vista previa de la cámara

```
private CameraCaptureSession mCaptureSession;
```

Una referencia al `CameraDevice` abierto `CameraDevice`

```
private CameraDevice mCameraDevice;
```

El `Size` de la vista previa de la cámara.

```
private Size mPreviewSize;
```

`CameraDevice.StateCallback` se llama cuando `CameraDevice` cambia su estado

```
private final CameraDevice.StateCallback mStateCallback = new CameraDevice.StateCallback() {

    @Override
    public void onOpened(@NonNull CameraDevice cameraDevice) {
        // This method is called when the camera is opened.  We start camera preview here.
        mCameraOpenCloseLock.release();
        mCameraDevice = cameraDevice;
        createCameraPreviewSession();
    }

    @Override
    public void onDisconnected(@NonNull CameraDevice cameraDevice) {
        mCameraOpenCloseLock.release();
        cameraDevice.close();
        mCameraDevice = null;
    }

    @Override
    public void onError(@NonNull CameraDevice cameraDevice, int error) {
        mCameraOpenCloseLock.release();
        cameraDevice.close();
        mCameraDevice = null;
```

```
        finish();
    }

};
```

Un hilo adicional para ejecutar tareas que no deberían bloquear la interfaz de usuario

```
private HandlerThread mBackgroundThread;
```

Un `Handler` para ejecutar tareas en segundo plano

```
private Handler mBackgroundHandler;
```

Un `ImageReader` que maneja la captura de imágenes fijas

```
private ImageReader mImageReader;
```

`CaptureRequest.Builder` para la vista previa de la cámara

```
private CaptureRequest.Builder mPreviewRequestBuilder;
```

`CaptureRequest` generado por `mPreviewRequestBuilder`

```
private CaptureRequest mPreviewRequest;
```

Un `Semaphore` para evitar que la aplicación salga antes de cerrar la cámara.

```
private Semaphore mCameraOpenCloseLock = new Semaphore(1);
```

ID constante de la solicitud de permiso

```
private static final int REQUEST_CAMERA_PERMISSION = 1;
```

Métodos de ciclo de vida de Android

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_camera2);

    mTextureView = (TextureView) findViewById(R.id.texture);
}

@Override
public void onResume() {
    super.onResume();
    startBackgroundThread();

    // When the screen is turned off and turned back on, the SurfaceTexture is already
    // available, and "onSurfaceTextureAvailable" will not be called. In that case, we can
open
```

```
    // a camera and start preview from here (otherwise, we wait until the surface is ready in
    // the SurfaceTextureListener).
    if (mTextureView.isAvailable()) {
        openCamera(mTextureView.getWidth(), mTextureView.getHeight());
    } else {
        mTextureView.setSurfaceTextureListener(mSurfaceTextureListener);
    }
}

@Override
public void onPause() {
    closeCamera();
    stopBackgroundThread();
    super.onPause();
}
```

Camera2 métodos relacionados

Esos son métodos que utilizan las API de Camera2

```
private void openCamera(int width, int height) {
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
            != PackageManager.PERMISSION_GRANTED) {
        requestCameraPermission();
        return;
    }
    setUpCameraOutputs(width, height);
    configureTransform(width, height);
    CameraManager manager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
    try {
        if (!mCameraOpenCloseLock.tryAcquire(2500, TimeUnit.MILLISECONDS)) {
            throw new RuntimeException("Time out waiting to lock camera opening.");
        }
        manager.openCamera(mCameraId, mStateCallback, mBackgroundHandler);
    } catch (CameraAccessException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        throw new RuntimeException("Interrupted while trying to lock camera opening.", e);
    }
}
```

Cierra la cámara actual.

```
private void closeCamera() {
    try {
        mCameraOpenCloseLock.acquire();
        if (null != mCaptureSession) {
            mCaptureSession.close();
            mCaptureSession = null;
        }
        if (null != mCameraDevice) {
            mCameraDevice.close();
            mCameraDevice = null;
        }
        if (null != mImageReader) {
            mImageReader.close();
            mImageReader = null;
        }
```

```
    } catch (InterruptedException e) {
        throw new RuntimeException("Interrupted while trying to lock camera closing.", e);
    } finally {
        mCameraOpenCloseLock.release();
    }
}
```

Configura variables miembro relacionadas con la cámara.

```
private void setUpCameraOutputs(int width, int height) {
    CameraManager manager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
    try {
        for (String cameraId : manager.getCameraIdList()) {
            CameraCharacteristics characteristics
                    = manager.getCameraCharacteristics(cameraId);

            // We don't use a front facing camera in this sample.
            Integer facing = characteristics.get(CameraCharacteristics.LENS_FACING);
            if (facing != null && facing == CameraCharacteristics.LENS_FACING_FRONT) {
                continue;
            }

            StreamConfigurationMap map = characteristics.get(
                    CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP);
            if (map == null) {
                continue;
            }

            // For still image captures, we use the largest available size.
            Size largest = Collections.max(
                    Arrays.asList(map.getOutputSizes(ImageFormat.JPEG)),
                    new CompareSizesByArea());
            mImageReader = ImageReader.newInstance(largest.getWidth(), largest.getHeight(),
                    ImageFormat.JPEG, /*maxImages*/2);
            mImageReader.setOnImageAvailableListener(
                    null, mBackgroundHandler);

            Point displaySize = new Point();
            getWindowManager().getDefaultDisplay().getSize(displaySize);
            int rotatedPreviewWidth = width;
            int rotatedPreviewHeight = height;
            int maxPreviewWidth = displaySize.x;
            int maxPreviewHeight = displaySize.y;

            if (maxPreviewWidth > MAX_PREVIEW_WIDTH) {
                maxPreviewWidth = MAX_PREVIEW_WIDTH;
            }

            if (maxPreviewHeight > MAX_PREVIEW_HEIGHT) {
                maxPreviewHeight = MAX_PREVIEW_HEIGHT;
            }

            // Danger! Attempting to use too large a preview size could  exceed the camera
            // bus' bandwidth limitation, resulting in gorgeous previews but the storage of
            // garbage capture data.
            mPreviewSize = chooseOptimalSize(map.getOutputSizes(SurfaceTexture.class),
                    rotatedPreviewWidth, rotatedPreviewHeight, maxPreviewWidth,
                    maxPreviewHeight, largest);

            mCameraId = cameraId;
```

```
            return;
        }
    } catch (CameraAccessException e) {
        e.printStackTrace();
    } catch (NullPointerException e) {
        // Currently an NPE is thrown when the Camera2API is used but not supported on the
        // device this code runs.
        Toast.makeText(Camera2Activity.this, "Camera2 API not supported on this device",
Toast.LENGTH_LONG).show();
    }
}
```

Crea una nueva `CameraCaptureSession` para la vista previa de la cámara

```
private void createCameraPreviewSession() {
    try {
        SurfaceTexture texture = mTextureView.getSurfaceTexture();
        assert texture != null;

        // We configure the size of default buffer to be the size of camera preview we want.
        texture.setDefaultBufferSize(mPreviewSize.getWidth(), mPreviewSize.getHeight());

        // This is the output Surface we need to start preview.
        Surface surface = new Surface(texture);

        // We set up a CaptureRequest.Builder with the output Surface.
        mPreviewRequestBuilder
                = mCameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
        mPreviewRequestBuilder.addTarget(surface);

        // Here, we create a CameraCaptureSession for camera preview.
        mCameraDevice.createCaptureSession(Arrays.asList(surface, mImageReader.getSurface()),
                new CameraCaptureSession.StateCallback() {

                    @Override
                    public void onConfigured(@NonNull CameraCaptureSession
cameraCaptureSession) {
                        // The camera is already closed
                        if (null == mCameraDevice) {
                            return;
                        }

                        // When the session is ready, we start displaying the preview.
                        mCaptureSession = cameraCaptureSession;
                        try {
                            // Auto focus should be continuous for camera preview.
                            mPreviewRequestBuilder.set(CaptureRequest.CONTROL_AF_MODE,
                                    CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);

                            // Finally, we start displaying the camera preview.
                            mPreviewRequest = mPreviewRequestBuilder.build();
                            mCaptureSession.setRepeatingRequest(mPreviewRequest,
                                    null, mBackgroundHandler);
                        } catch (CameraAccessException e) {
                            e.printStackTrace();
                        }
                    }

                    @Override
                    public void onConfigureFailed(
```

```
                            @NonNull CameraCaptureSession cameraCaptureSession) {
                        showToast("Failed");
                    }
                }, null
        );
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
}
```

Métodos relacionados con permisos para Android API 23+

```
private void requestCameraPermission() {
    if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.CAMERA))
{
        new AlertDialog.Builder(Camera2Activity.this)
                .setMessage("R string request permission")
                .setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener()
{
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        ActivityCompat.requestPermissions(Camera2Activity.this,
                                new String[]{Manifest.permission.CAMERA},
                                REQUEST_CAMERA_PERMISSION);
                    }
                })
                .setNegativeButton(android.R.string.cancel,
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                finish();

                            }
                        })
                .create();

    } else {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA},
                REQUEST_CAMERA_PERMISSION);
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    if (requestCode == REQUEST_CAMERA_PERMISSION) {
        if (grantResults.length != 1 || grantResults[0] != PackageManager.PERMISSION_GRANTED)
{
            Toast.makeText(Camera2Activity.this, "ERROR: Camera permissions not granted",
Toast.LENGTH_LONG).show();
        }
    } else {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

Hilos de fondo / métodos de manejo

```
private void startBackgroundThread() {
```

```
    mBackgroundThread = new HandlerThread("CameraBackground");
    mBackgroundThread.start();
    mBackgroundHandler = new Handler(mBackgroundThread.getLooper());
}

private void stopBackgroundThread() {
    mBackgroundThread.quitSafely();
    try {
        mBackgroundThread.join();
        mBackgroundThread = null;
        mBackgroundHandler = null;
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

Métodos de utilidad

Dadas las opciones de `Size` admitidas por una cámara, elija la más pequeña que sea al menos tan grande como el tamaño de la vista de textura respectiva, y que sea tan grande como el tamaño máximo respectivo, y cuya relación de aspecto coincida con el valor especificado. Si no existe, elija el más grande que sea a lo sumo tan grande como el tamaño máximo respectivo, y cuya relación de aspecto coincida con el valor especificado

```
private static Size chooseOptimalSize(Size[] choices, int textureViewWidth,
                                      int textureViewHeight, int maxWidth, int maxHeight, Size
aspectRatio) {

    // Collect the supported resolutions that are at least as big as the preview Surface
    List<Size> bigEnough = new ArrayList<>();
    // Collect the supported resolutions that are smaller than the preview Surface
    List<Size> notBigEnough = new ArrayList<>();
    int w = aspectRatio.getWidth();
    int h = aspectRatio.getHeight();
    for (Size option : choices) {
        if (option.getWidth() <= maxWidth && option.getHeight() <= maxHeight &&
                option.getHeight() == option.getWidth() * h / w) {
            if (option.getWidth() >= textureViewWidth &&
                    option.getHeight() >= textureViewHeight) {
                bigEnough.add(option);
            } else {
                notBigEnough.add(option);
            }
        }
    }

    // Pick the smallest of those big enough. If there is no one big enough, pick the
    // largest of those not big enough.
    if (bigEnough.size() > 0) {
        return Collections.min(bigEnough, new CompareSizesByArea());
    } else if (notBigEnough.size() > 0) {
        return Collections.max(notBigEnough, new CompareSizesByArea());
    } else {
        Log.e("Camera2", "Couldn't find any suitable preview size");
        return choices[0];
    }
}
```

Este método configura la transformación `Matrix` necesaria para `mTextureView`

```
private void configureTransform(int viewWidth, int viewHeight) {
    if (null == mTextureView || null == mPreviewSize) {
        return;
    }
    int rotation = getWindowManager().getDefaultDisplay().getRotation();
    Matrix matrix = new Matrix();
    RectF viewRect = new RectF(0, 0, viewWidth, viewHeight);
    RectF bufferRect = new RectF(0, 0, mPreviewSize.getHeight(), mPreviewSize.getWidth());
    float centerX = viewRect.centerX();
    float centerY = viewRect.centerY();
    if (Surface.ROTATION_90 == rotation || Surface.ROTATION_270 == rotation) {
        bufferRect.offset(centerX - bufferRect.centerX(), centerY - bufferRect.centerY());
        matrix.setRectToRect(viewRect, bufferRect, Matrix.ScaleToFit.FILL);
        float scale = Math.max(
                (float) viewHeight / mPreviewSize.getHeight(),
                (float) viewWidth / mPreviewSize.getWidth());
        matrix.postScale(scale, scale, centerX, centerY);
        matrix.postRotate(90 * (rotation - 2), centerX, centerY);
    } else if (Surface.ROTATION_180 == rotation) {
        matrix.postRotate(180, centerX, centerY);
    }
    mTextureView.setTransform(matrix);
}
```

Este método compara dos `Size` basados en sus áreas.

```
static class CompareSizesByArea implements Comparator<Size> {

    @Override
    public int compare(Size lhs, Size rhs) {
        // We cast here to ensure the multiplications won't overflow
        return Long.signum((long) lhs.getWidth() * lhs.getHeight() -
                (long) rhs.getWidth() * rhs.getHeight());
    }
}
```

no hay mucho que ver aqui

```
/**
 * Shows a {@link Toast} on the UI thread.
 *
 * @param text The message to show
 */
private void showToast(final String text) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            Toast.makeText(Camera2Activity.this, text, Toast.LENGTH_SHORT).show();
        }
    });
}
```

Lea API de la cámara 2 en línea: https://riptutorial.com/es/android/topic/619/api-de-la-camara-2

Capítulo 25: API de Twitter

Examples

Crear login con el botón de twitter y adjuntarle una devolución

1. Dentro de su diseño, agregue un botón de inicio de sesión con el siguiente código:

```
<com.twitter.sdk.android.core.identity.TwitterLoginButton
    android:id="@+id/twitter_login_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"/>
```

2. En la Actividad o Fragmento que muestra el botón, debe crear y adjuntar una Devolución de llamada al Botón de inicio de sesión como sigue:

```
import com.twitter.sdk.android.core.Callback;
import com.twitter.sdk.android.core.Result;
import com.twitter.sdk.android.core.TwitterException;
import com.twitter.sdk.android.core.TwitterSession;
import com.twitter.sdk.android.core.identity.TwitterLoginButton;
...

loginButton = (TwitterLoginButton) findViewById(R.id.login_button);
loginButton.setCallback(new Callback<TwitterSession>() {
    @Override
    public void success(Result<TwitterSession> result) {
        Log.d(TAG, "userName: " + session.getUserName());
        Log.d(TAG, "userId: " + session.getUserId());
        Log.d(TAG, "authToken: " + session.getAuthToken());
        Log.d(TAG, "id: " + session.getId());
        Log.d(TAG, "authToken: " + session.getAuthToken().token);
        Log.d(TAG, "authSecret: " + session.getAuthToken().secret);
    }

    @Override
    public void failure(TwitterException exception) {
        // Do something on failure
    }
});
```

3. Pase el resultado de la actividad de autenticación de nuevo al botón:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    // Make sure that the loginButton hears the result from any
    // Activity that it triggered.
    loginButton.onActivityResult(requestCode, resultCode, data);
}
```

Tenga en cuenta que si usa el botón TwitterLoginButton en un fragmento, use los

siguientes pasos en su lugar:

```
 @Override
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
     super.onActivityResult(requestCode, resultCode, data);

     // Pass the activity result to the fragment, which will then pass the result to the
login
     // button.
     Fragment fragment = getFragmentManager().findFragmentById(R.id.your_fragment_id);
     if (fragment != null) {
         fragment.onActivityResult(requestCode, resultCode, data);
     }
 }
```

4. Agregue las siguientes líneas a sus dependencias de **build.gradle** :

```
 apply plugin: 'io.fabric'

 repositories {
     maven { url 'https://maven.fabric.io/public' }
 }

 compile('com.twitter.sdk.android:twitter:1.14.1@aar') {
     transitive = true;
 }
```

Lea API de Twitter en línea: https://riptutorial.com/es/android/topic/4801/api-de-twitter

Capítulo 26: API de Youtube

Observaciones

1. En primer lugar, debe descargar la última versión del siguiente enlace https://developers.google.com/youtube/android/player/downloads/
2. Necesitas incluir este frasco en tu proyecto. Copie y pegue este jar en la carpeta libs y no olvide agregarlo en las dependencias de archivos de gradle {compile los archivos ('libs / YouTubeAndroidPlayerApi.jar')}
3. Necesitas una clave api para acceder a los api de youtube. Siga este enlace: https://developers.google.com/youtube/android/player/register para generar su clave de api.
4. Limpia y construye tu proyecto. Ahora está listo para usar YoutubeAndroidPlayerApi Para reproducir un video de youtube, debe tener la identificación del video correspondiente para poder reproducirlo en youtube. Por ejemplo: https://www.youtube.com/watch?v=B08iLAtS3AQ , B08iLAtS3AQ es el ID de video que necesita para reproducirlo en youtube.

Examples

Lanzamiento de StandAlonePlayerActivity

1. Lanzar la actividad del jugador independiente

```
     Intent standAlonePlayerIntent = YouTubeStandalonePlayer.createVideoIntent((Activity)
context,
             Config.YOUTUBE_API_KEY, // which you have created in step 3
             videoId, // video which is to be played
             100,     //The time, in milliseconds, where playback should start in the
video
             true,    //autoplay or not
             false);   //lightbox mode or not; false will show in fullscreen
     context.startActivity(standAlonePlayerIntent);
```

Actividad que extiende YouTubeBaseActivity

```
public class CustomYouTubeActivity extends YouTubeBaseActivity implements
YouTubePlayer.OnInitializedListener, YouTubePlayer.PlayerStateChangeListener {

    private YouTubePlayerView mPlayerView;
    private YouTubePlayer mYouTubePlayer;
    private String mVideoId = "B08iLAtS3AQ";
    private String mApiKey;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mApiKey = Config.YOUTUBE_API_KEY;
        mPlayerView = new YouTubePlayerView(this);
        mPlayerView.initialize(mApiKey, this); // setting up OnInitializedListener
```

```
        addContentView(mPlayerView, new LayoutParams(LayoutParams.MATCH_PARENT,
                LayoutParams.MATCH_PARENT)); //show it in full screen
    }

    //Called when initialization of the player succeeds.
    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider,
                                        YouTubePlayer player,
                                        boolean wasRestored) {

        player.setPlayerStateChangeListener(this); // setting up the player state change
listener
        this.mYouTubePlayer = player;
        if (!wasRestored)
            player.cueVideo(mVideoId);
    }

        @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
                                        YouTubeInitializationResult errorReason) {

        Toast.makeText(this, "Error While initializing", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onAdStarted() {
    }

    @Override
    public void onLoaded(String videoId) { //video has been loaded
        if(!TextUtils.isEmpty(mVideoId) && !this.isFinishing() && mYouTubePlayer != null)
            mYouTubePlayer.play(); // if we dont call play then video will not auto play, but
user still has the option to play via play button
    }

    @Override
    public void onLoading() {
    }

    @Override
    public void onVideoEnded() {
    }

    @Override
    public void onVideoStarted() {

    }

    @Override
    public void onError(ErrorReason reason) {
        Log.e("onError", "onError : " + reason.name());
    }

}
```

YoutubePlayerFragmento en retrato Activty

El siguiente código implementa un YoutubePlayerFragment simple. El diseño de la actividad se bloquea en modo vertical y cuando cambia la orientación o el usuario hace clic en pantalla

completa en YoutubePlayer, se convierte en lansscape con YoutubePlayer llenando la pantalla. El YoutubePlayerFragment no necesita extender una actividad proporcionada por la biblioteca de Youtube. Necesita implementar YouTubePlayer.OnInitializedListener para poder inicializar YoutubePlayer. Así que la clase de nuestra actividad es la siguiente.

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.widget.Toast;

import com.google.android.youtube.player.YouTubeInitializationResult;
import com.google.android.youtube.player.YouTubePlayer;
import com.google.android.youtube.player.YouTubePlayerFragment;

public class MainActivity extends AppCompatActivity implements
YouTubePlayer.OnInitializedListener {

    public static final String API_KEY ;
    public static final String VIDEO_ID = "B08iLAtS3AQ";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        YouTubePlayerFragment youTubePlayerFragment = (YouTubePlayerFragment)
getFragmentManager()
                .findFragmentById(R.id.youtubeplayerfragment);

        youTubePlayerFragment.initialize(API_KEY, this);

    }

    /**
     *
     * @param provider The provider which was used to initialize the YouTubePlayer
     * @param youTubePlayer A YouTubePlayer which can be used to control video playback in the
provider.
     * @param wasRestored Whether the player was restored from a previously saved state, as
part of the YouTubePlayerView
     *                    or YouTubePlayerFragment restoring its state. true usually means
playback is resuming from where
     *                    the user expects it would, and that a new video should not be loaded
     */
    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider, YouTubePlayer
youTubePlayer, boolean wasRestored) {

youTubePlayer.setFullscreenControlFlags(YouTubePlayer.FULLSCREEN_FLAG_CONTROL_ORIENTATION |
                    YouTubePlayer.FULLSCREEN_FLAG_ALWAYS_FULLSCREEN_IN_LANDSCAPE);


        if(!wasRestored) {
            youTubePlayer.cueVideo(VIDEO_ID);
        }
    }

    /**
     *
     * @param provider The provider which failed to initialize a YouTubePlayer.
```

```
     * @param error The reason for this failure, along with potential resolutions to this
failure.
     */
    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
YouTubeInitializationResult error) {

        final int REQUEST_CODE = 1;

        if(error.isUserRecoverableError()) {
            error.getErrorDialog(this,REQUEST_CODE).show();
        } else {
            String errorMessage = String.format("There was an error initializing the
YoutubePlayer (%1$s)", error.toString());
            Toast.makeText(this, errorMessage, Toast.LENGTH_LONG).show();
        }
    }
}
```

Un YoutubePlayerFragment se puede agregar al diseño de la actividad xaml como se indica a continuación.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:tools="http://schemas.android.com/tools"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical"
              android:paddingBottom="@dimen/activity_vertical_margin"
              android:paddingLeft="@dimen/activity_horizontal_margin"
              android:paddingRight="@dimen/activity_horizontal_margin"
              android:paddingTop="@dimen/activity_vertical_margin"
              tools:context=".MainActivity">

    <fragment
        android:id="@+id/youtubeplayerfragment"
        android:name="com.google.android.youtube.player.YouTubePlayerFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>


    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">


            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>

            <TextView
                android:layout_width="wrap_content"
```

```
                android:layout_height="wrap_content"

                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>

        </LinearLayout>
    </ScrollView>

</LinearLayout>
```

Por último, debe agregar los siguientes atributos en su archivo de manifiesto dentro de la etiqueta de la actividad

```
android:configChanges="keyboardHidden|orientation|screenSize"
android:screenOrientation="portrait"
```

API de reproductor de YouTube

Obteniendo la clave API de Android:

Primero necesitará obtener la huella dactilar SHA-1 en su máquina usando la herramienta de teclado Java. Ejecute el siguiente comando en cmd / terminal para obtener la huella dactilar SHA-1.

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android
-keypass android
```

MainActivity.java

```
public class Activity extends YouTubeBaseActivity implements
YouTubePlayer.OnInitializedListener {

    private static final int RECOVERY_DIALOG_REQUEST = 1;

    // YouTube player view
    private YouTubePlayerView youTubeView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setContentView(R.layout.activity_main);

        youTubeView = (YouTubePlayerView) findViewById(R.id.youtube_view);

        // Initializing video player with developer key
        youTubeView.initialize(Config.DEVELOPER_KEY, this);

    }

    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
                                        YouTubeInitializationResult errorReason) {
        if (errorReason.isUserRecoverableError()) {
            errorReason.getErrorDialog(this, RECOVERY_DIALOG_REQUEST).show();
        } else {
            String errorMessage = String.format(
                    getString(R.string.error_player), errorReason.toString());
            Toast.makeText(this, errorMessage, Toast.LENGTH_LONG).show();
        }
    }

    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider,
                                        YouTubePlayer player, boolean wasRestored) {
        if (!wasRestored) {

            // loadVideo() will auto play video
            // Use cueVideo() method, if you don't want to play it automatically
            player.loadVideo(Config.YOUTUBE_VIDEO_CODE);

            // Hiding player controls
            player.setPlayerStyle(YouTubePlayer.PlayerStyle.CHROMELESS);
```

```
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == RECOVERY_DIALOG_REQUEST) {
            // Retry initialization if user performed a recovery action
            getYouTubePlayerProvider().initialize(Config.DEVELOPER_KEY, this);
        }
    }

    private YouTubePlayer.Provider getYouTubePlayerProvider() {
        return (YouTubePlayerView) findViewById(R.id.youtube_view);
    }
}
```

Ahora crea el archivo `Config.java` . Este archivo contiene la clave de desarrollador de la API de la Consola de Google y el ID de video de YouTube

Config.java

```
public class Config {

    // Developer key
    public static final String DEVELOPER_KEY = "AIzaSyDZtE10od_hXM5aXYEh6Zn7c6brV9ZjKuk";

    // YouTube video id
    public static final String YOUTUBE_VIDEO_CODE = "_oEA18Y8gM0";
}
```

archivo xml

```
<com.google.android.youtube.player.YouTubePlayerView
                android:id="@+id/youtube_view"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="30dp" />
```

Consumiendo API de datos de YouTube en Android

Este ejemplo lo guiará sobre cómo obtener datos de la lista de reproducción utilizando la API de datos de YouTube en Android.

Huella digital SHA-1

Primero necesita obtener una huella dactilar SHA-1 para su máquina. Hay varios métodos para recuperarlo. Puede elegir cualquier método proporcionado en esta Q&A .

Consola de API de Google y clave de YouTube para Android

Ahora que tiene una huella dactilar SHA-1, abra la consola de la API de Google y cree un proyecto. Vaya a esta página y cree un proyecto con esa clave SHA-1 y habilite la API de datos de YouTube. Ahora obtendrás una llave. Esta clave se utilizará para enviar solicitudes desde

Android y recuperar datos.

Parte de Gradle

Deberá agregar las siguientes líneas a su archivo de Gradle para la API de datos de YouTube:

```
compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
```

Para usar el cliente nativo de YouTube para enviar solicitudes, debemos agregar las siguientes líneas en Gradle:

```
compile 'com.google.http-client:google-http-client-android:+'
compile 'com.google.api-client:google-api-client-android:+'
compile 'com.google.api-client:google-api-client-gson:+'
```

La siguiente configuración también debe agregarse en Gradle para evitar conflictos:

```
configurations.all {
    resolutionStrategy.force 'com.google.code.findbugs:jsr305:3.0.2'
}
```

A continuación se muestra cómo se *vería* finalmente el *gradle.build* .

construir.gradle

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "com.aam.skillschool"
        minSdkVersion 19
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    configurations.all {
        resolutionStrategy.force 'com.google.code.findbugs:jsr305:3.0.2'
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
    compile 'com.android.support:appcompat-v7:25.3.1'
```

```
    compile 'com.android.support:support-v4:25.3.1'
    compile 'com.google.http-client:google-http-client-android:+'
    compile 'com.google.api-client:google-api-client-android:+'
    compile 'com.google.api-client:google-api-client-gson:+'
}
```

Ahora viene la parte de Java. Ya que `HttpTransport` para redes y `GsonFactory` para convertir JSON en POJO, no necesitamos ninguna otra biblioteca para enviar ninguna solicitud.

Ahora quiero mostrar cómo obtener listas de reproducción a través de la API de YouTube proporcionando los ID de lista de reproducción. Para esta tarea utilizaré `AsyncTask` . Para comprender cómo solicitamos los parámetros y para comprender el flujo, eche un vistazo a la API de datos de YouTube .

```
public class GetPlaylistDataAsyncTask extends AsyncTask<String[], Void, PlaylistListResponse>
{
    private static final String YOUTUBE_PLAYLIST_PART = "snippet";
    private static final String YOUTUBE_PLAYLIST_FIELDS = "items(id,snippet(title))";

    private YouTube mYouTubeDataApi;

    public GetPlaylistDataAsyncTask(YouTube api) {
        mYouTubeDataApi = api;
    }

    @Override
    protected PlaylistListResponse doInBackground(String[]... params) {

        final String[] playlistIds = params[0];

        PlaylistListResponse playlistListResponse;
        try {
            playlistListResponse = mYouTubeDataApi.playlists()
                    .list(YOUTUBE_PLAYLIST_PART)
                    .setId(TextUtils.join(",", playlistIds))
                    .setFields(YOUTUBE_PLAYLIST_FIELDS)
                    .setKey(AppConstants.YOUTUBE_KEY) //Here you will have to provide the keys
                    .execute();
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }

        return playlistListResponse;
    }
}
```

La tarea asíncrona anterior devolverá una instancia de `PlaylistListResponse` que es una clase incorporada del SDK de YouTube. Tiene todos los campos requeridos, por lo que no tenemos que crear POJOs nosotros mismos.

Finalmente, en nuestra `MainActivity` tendremos que hacer lo siguiente:

```
public class MainActivity extends AppCompatActivity {
    private YouTube mYoutubeDataApi;
    private final GsonFactory mJsonFactory = new GsonFactory();
```

```
    private final HttpTransport mTransport = AndroidHttp.newCompatibleTransport();
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_review);
        mYoutubeDataApi = new YouTube.Builder(mTransport, mJsonFactory, null)
                .setApplicationName(getResources().getString(R.string.app_name))
                .build();
        String[] ids = {"some playlists ids here seperated by "," };
        new GetPlaylistDataAsyncTask(mYoutubeDataApi) {
            ProgressDialog progressDialog = new ProgressDialog(getActivity());

            @Override
            protected void onPreExecute() {
                progressDialog.setTitle("Please wait.....");
                progressDialog.show();
                super.onPreExecute();
            }

            @Override
            protected void onPostExecute(PlaylistListResponse playlistListResponse) {
                super.onPostExecute(playlistListResponse);
                //Here we get the playlist data
                progressDialog.dismiss();
                Log.d(TAG, playlistListResponse.toString());
            }
        }.execute(ids);
    }
}
```

Lea API de Youtube en línea: https://riptutorial.com/es/android/topic/7587/api-de-youtube

Capítulo 27: Archivo zip en android

Examples

Archivo zip en Android

```
import android.util.Log;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;


  public class Compress {
      private static final int BUFFER = 2048;

      private String[] _files;
      private String _zipFile;

      public Compress(String[] files, String zipFile) {
        _files = files;
        _zipFile = zipFile;
      }

      public void zip() {
        try  {
          BufferedInputStream origin = null;
          FileOutputStream dest = new FileOutputStream(_zipFile);

          ZipOutputStream out = new ZipOutputStream(new BufferedOutputStream(dest));

          byte data[] = new byte[BUFFER];

          for(int i=0; i < _files.length; i++) {
            Log.v("Compress", "Adding: " + _files[i]);
            FileInputStream fi = new FileInputStream(_files[i]);
            origin = new BufferedInputStream(fi, BUFFER);
            ZipEntry entry = new ZipEntry(_files[i].substring(_files[i].lastIndexOf("/") +
1));
            out.putNextEntry(entry);
            int count;
            while ((count = origin.read(data, 0, BUFFER)) != -1) {
              out.write(data, 0, count);
            }
            origin.close();
          }

          out.close();
        } catch(Exception e) {
          e.printStackTrace();
        }

      }

    }
```

Lea Archivo zip en android en línea: https://riptutorial.com/es/android/topic/8137/archivo-zip-en-android

Capítulo 28: Arquitectura MVP

Introducción

Este tema proporcionará la arquitectura de Android de Modelo-Vista-Presentador (MVP) con varios ejemplos.

Observaciones

Hay muchas maneras de diseñar una aplicación de Android. Pero no todos son verificables y nos permiten estructurar nuestro código para que la aplicación sea fácil de probar. La idea clave de una arquitectura comprobable es la separación de partes de la aplicación, lo que facilita su mantenimiento, extensión y prueba por separado.

Definición de MVP

Modelo

En una aplicación con una buena arquitectura en capas, este modelo solo sería la puerta de entrada a la capa de dominio o lógica empresarial. Véalo como el proveedor de los datos que queremos mostrar en la vista.

Ver

La Vista, generalmente implementada por una `Activity` o `Fragment` , contendrá una referencia al *presentador* . Lo único que hará la vista es llamar a un método desde el Presentador cada vez que haya una acción de interfaz.

Presentador

El presentador es responsable de actuar como intermediario entre View y Model. Recupera datos del modelo y los devuelve formateados a la vista. Pero a diferencia del MVC típico, también decide qué sucede cuando interactúas con la Vista.

* Definiciones del artículo de Antonio Leiva.

Estructura de aplicación *recomendada* (no requerida)

La aplicación debe estar estructurada por paquete *por función* . Esto mejora la legibilidad y modulariza la aplicación de manera que partes de ella se pueden cambiar de forma independiente entre sí. Cada característica clave de la aplicación está en su propio paquete de Java.

Examples

Ejemplo de inicio de sesión en el patrón de Model View Presenter (MVP)

Veamos MVP en acción usando una simple pantalla de inicio de sesión. Hay dos `Button` : uno para la acción de inicio de sesión y otro para una pantalla de registro; dos `EditText` s: uno para el correo electrónico y otro para la contraseña.

LoginFragment (la vista)

```
public class LoginFragment extends Fragment implements LoginContract.PresenterToView,
View.OnClickListener {

    private View view;
    private EditText email, password;
    private Button login, register;

    private LoginContract.ToPresenter presenter;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        return inflater.inflate(R.layout.fragment_login, container, false);
    }

    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        email = (EditText) view.findViewById(R.id.email_et);
        password = (EditText) view.findViewById(R.id.password_et);
        login = (Button) view.findViewById(R.id.login_btn);
        login.setOnClickListener(this);
        register = (Button) view.findViewById(R.id.register_btn);
        register.setOnClickListener(this);

        presenter = new LoginPresenter(this);

        presenter.isLoggedIn();

    }

    @Override
    public void onLoginResponse(boolean isLoginSuccess) {
        if (isLoginSuccess) {
            startActivity(new Intent(getActivity(), MapActivity.class));
            getActivity().finish();
        }
    }

    @Override
    public void onError(String message) {
        Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void isLoggedIn(boolean isLoggedIn) {
        if (isLoggedIn) {
```

```
            startActivity(new Intent(getActivity(), MapActivity.class));
            getActivity().finish();
        }
    }

    @Override
    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.login_btn:
                LoginItem loginItem = new LoginItem();
                loginItem.setPassword(password.getText().toString().trim());
                loginItem.setEmail(email.getText().toString().trim());
                presenter.login(loginItem);
                break;
            case R.id.register_btn:
                startActivity(new Intent(getActivity(), RegisterActivity.class));
                getActivity().finish();
                break;
        }
    }
}
```

LoginPresenter (El Presentador)

```
public class LoginPresenter implements LoginContract.ToPresenter {

    private LoginContract.PresenterToModel model;
    private LoginContract.PresenterToView view;

    public LoginPresenter(LoginContract.PresenterToView view) {
        this.view = view;
        model = new LoginModel(this);
    }

    @Override
    public void login(LoginItem userCredentials) {
        model.login(userCredentials);
    }

    @Override
    public void isLoggedIn() {
        model.isLoggedIn();
    }

    @Override
    public void onLoginResponse(boolean isLoginSuccess) {
        view.onLoginResponse(isLoginSuccess);
    }

    @Override
    public void onError(String message) {
        view.onError(message);
    }

    @Override
    public void isloggedIn(boolean isLoggedin) {
        view.isLoggedIn(isLoggedin);
    }
}
```

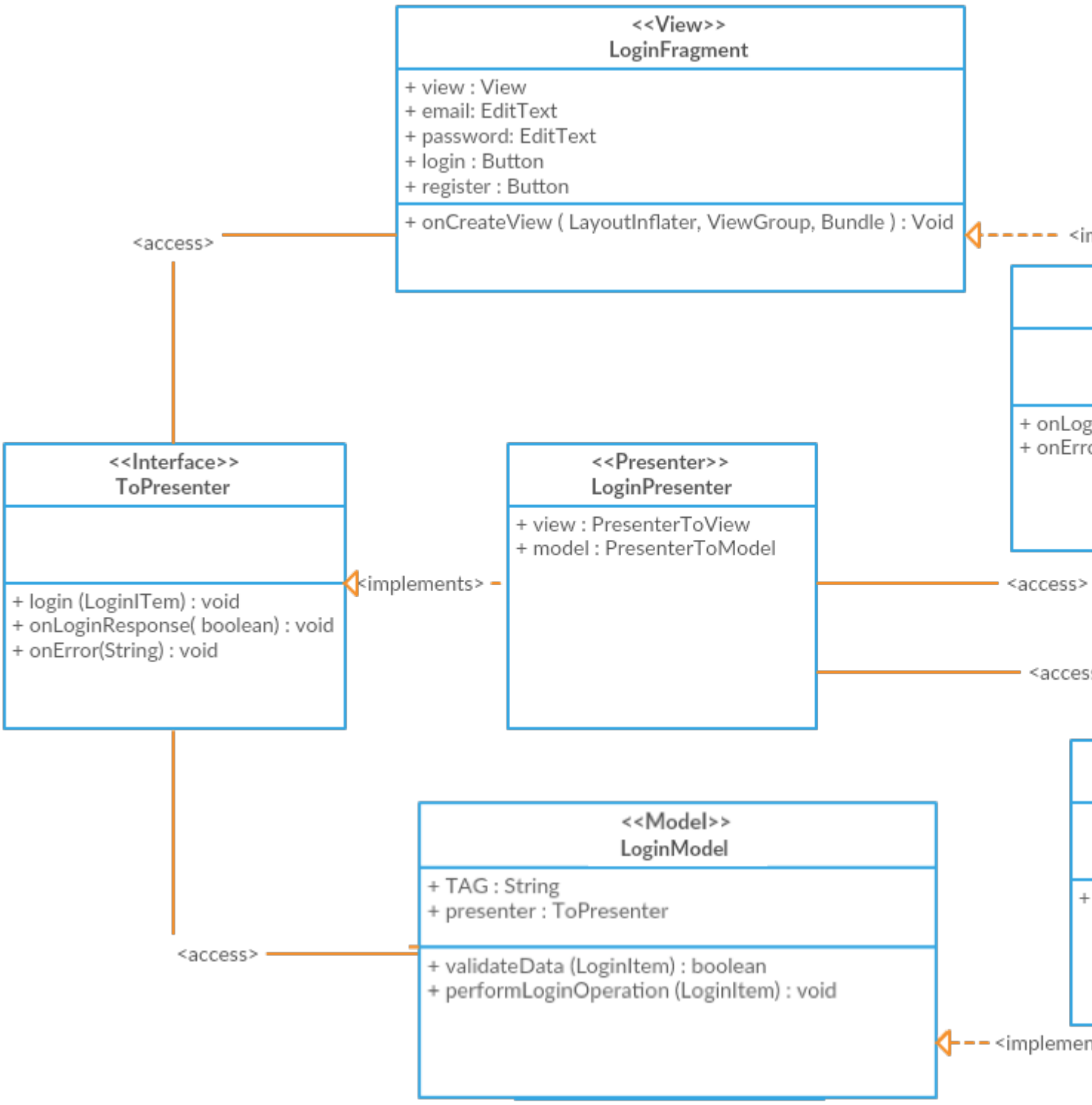
LoginModel (El Modelo)

```
public class LoginModel implements LoginContract.PresenterToModel,
ResponseErrorListener.ErrorListener {

    private static final String TAG = LoginModel.class.getSimpleName();
    private LoginContract.ToPresenter presenter;

    public LoginModel(LoginContract.ToPresenter presenter) {
        this.presenter = presenter;
    }

    @Override
    public void login(LoginItem userCredentials) {
        if (validateData(userCredentials)) {
            try {
                performLoginOperation(userCredentials);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        } else {

presenter.onError(BaseContext.getContext().getString(R.string.error_login_field_validation));
        }
    }

    @Override
    public void isLoggedIn() {
        DatabaseHelper database = new DatabaseHelper(BaseContext.getContext());
        presenter.isloggedIn(database.isLoggedIn());
    }

    private boolean validateData(LoginItem userCredentials) {
        return Patterns.EMAIL_ADDRESS.matcher(userCredentials.getEmail()).matches()
                && !userCredentials.getPassword().trim().equals("");
    }

    private void performLoginOperation(final LoginItem userCredentials) throws JSONException {

        JSONObject postData = new JSONObject();
        postData.put(Constants.EMAIL, userCredentials.getEmail());
        postData.put(Constants.PASSWORD, userCredentials.getPassword());

        JsonObjectRequest request = new JsonObjectRequest(Request.Method.POST, Url.AUTH,
postData,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        try {
                            String token = response.getString(Constants.ACCESS_TOKEN);
                            DatabaseHelper databaseHelper = new
DatabaseHelper(BaseContext.getContext());
                            databaseHelper.login(token);
                            Log.d(TAG, "onResponse: " + token);
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                        presenter.onLoginResponse(true);
                    }
                }, new ErrorResponse(this));
```

```
        RequestQueue queue = Volley.newRequestQueue(BaseContext.getContext());
        queue.add(request);
    }

    @Override
    public void onError(String message) {
        presenter.onError(message);
    }
}
```

Diagrama de clase

Veamos la acción en forma de diagrama de clase.

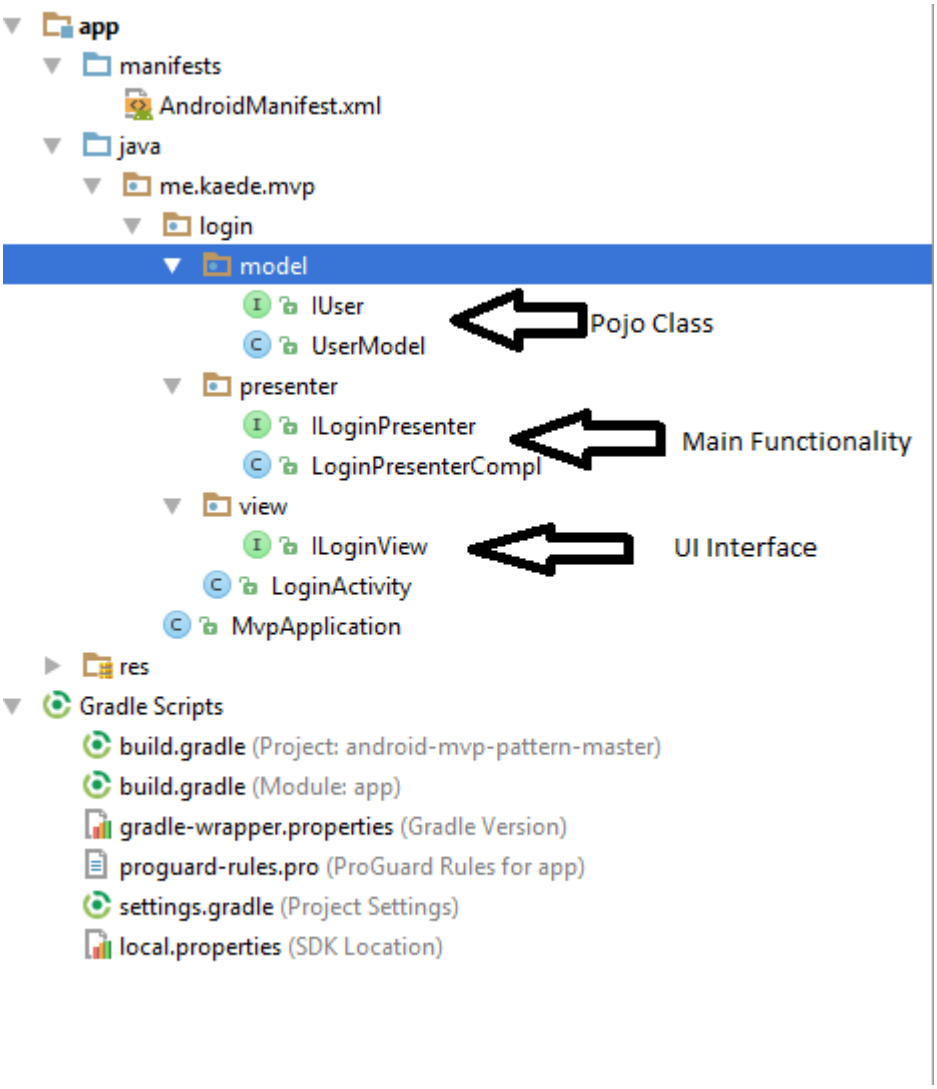


Notas:

- Este ejemplo utiliza Volley para la comunicación de red, pero esta biblioteca no es necesaria para MVP
- `UrlUtils` es una clase que contiene todos los enlaces para mis `UrlUtils` API
- `ResponseErrorListener.ErrorListener` es una `interface` que escucha el error en `ErrorResponse` que `implements Response.ErrorListener` de Volley; estas clases no se incluyen aquí, ya que no forman parte directamente de este ejemplo

Ejemplo de inicio de sesión simple en MVP

Estructura del paquete requerido



XML activity_login

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <EditText
        android:id="@+id/et_login_username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="USERNAME" />

    <EditText
```

```
        android:id="@+id/et_login_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="PASSWORD" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btn_login_login"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginRight="4dp"
            android:layout_weight="1"
            android:text="Login" />

        <Button
            android:id="@+id/btn_login_clear"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="4dp"
            android:layout_weight="1"
            android:text="Clear" />
    </LinearLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="3dp"
        android:text="correct user: mvp, mvp" />

    <ProgressBar
        android:id="@+id/progress_login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp" />

</LinearLayout>
```

Actividad Clase LoginActivity.class

```
public class LoginActivity extends AppCompatActivity implements ILoginView,
View.OnClickListener {
    private EditText editUser;
    private EditText editPass;
    private Button   btnLogin;
    private Button   btnClear;
    private ILoginPresenter loginPresenter;
    private ProgressBar progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        //find view
```

```
        editUser = (EditText) this.findViewById(R.id.et_login_username);
        editPass = (EditText) this.findViewById(R.id.et_login_password);
        btnLogin = (Button) this.findViewById(R.id.btn_login_login);
        btnClear = (Button) this.findViewById(R.id.btn_login_clear);
        progressBar = (ProgressBar) this.findViewById(R.id.progress_login);

        //set listener
        btnLogin.setOnClickListener(this);
        btnClear.setOnClickListener(this);

        //init
        loginPresenter = new LoginPresenterCompl(this);
        loginPresenter.setProgressBarVisiblity(View.INVISIBLE);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btn_login_clear:
                loginPresenter.clear();
                break;
            case R.id.btn_login_login:
                loginPresenter.setProgressBarVisiblity(View.VISIBLE);
                btnLogin.setEnabled(false);
                btnClear.setEnabled(false);
                loginPresenter.doLogin(editUser.getText().toString(),
editPass.getText().toString());
                break;
        }
    }

    @Override
    public void onClearText() {
        editUser.setText("");
        editPass.setText("");
    }

    @Override
    public void onLoginResult(Boolean result, int code) {
        loginPresenter.setProgressBarVisiblity(View.INVISIBLE);
        btnLogin.setEnabled(true);
        btnClear.setEnabled(true);
        if (result){
            Toast.makeText(this,"Login Success",Toast.LENGTH_SHORT).show();
        }
        else
            Toast.makeText(this,"Login Fail, code = " + code,Toast.LENGTH_SHORT).show();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

    @Override
    public void onSetProgressBarVisibility(int visibility) {
        progressBar.setVisibility(visibility);
    }
}
```

Creando una interfaz ILoginView

Cree una interfaz `ILoginView` para actualizar la información de Presenter en la carpeta de vista de la siguiente manera:

```
public interface ILoginView {
    public void onClearText();
    public void onLoginResult(Boolean result, int code);
    public void onSetProgressBarVisibility(int visibility);
}
```

Creando una interfaz ILoginPresenter

Cree una interfaz `ILoginPresenter` para comunicarse con `LoginActivity` (Vistas) y cree la clase `LoginPresenterCompl` para manejar la funcionalidad de inicio de sesión e informar a la Actividad. La clase `LoginPresenterCompl` implementa la interfaz `ILoginPresenter` :

ILoginPresenter.class

```
public interface ILoginPresenter {
    void clear();
    void doLogin(String name, String passwd);
    void setProgressBarVisiblity(int visiblity);
}
```

LoginPresenterCompl.class

```
public class LoginPresenterCompl implements ILoginPresenter {
    ILoginView iLoginView;
    IUser user;
    Handler handler;

    public LoginPresenterCompl(ILoginView iLoginView) {
        this.iLoginView = iLoginView;
        initUser();
        handler = new Handler(Looper.getMainLooper());
    }

    @Override
    public void clear() {
        iLoginView.onClearText();
    }

    @Override
    public void doLogin(String name, String passwd) {
        Boolean isLoginSuccess = true;
        final int code = user.checkUserValidity(name,passwd);
        if (code!=0) isLoginSuccess = false;
        final Boolean result = isLoginSuccess;
        handler.postDelayed(new Runnable() {
```

```
            @Override
            public void run() {
                    iLoginView.onLoginResult(result, code);
            }
        }, 5000);
    }

    @Override
    public void setProgressBarVisiblity(int visiblity){
        iLoginView.onSetProgressBarVisibility(visiblity);
    }

    private void initUser(){
        user = new UserModel("mvp","mvp");
    }
}
```

Creando un UserModel

Cree un `UserModel` que sea como una clase Pojo para `LoginActivity` . Cree una interfaz `IUser` para las validaciones de Pojo:

UserModel.class

```
public class UserModel implements IUser {
String name;
String passwd;

public UserModel(String name, String passwd) {
    this.name = name;
    this.passwd = passwd;
}

@Override
public String getName() {
    return name;
}

@Override
public String getPasswd() {
    return passwd;
}

@Override
public int checkUserValidity(String name, String passwd){
    if (name==null||passwd==null||!name.equals(getName())||!passwd.equals(getPasswd())){
        return -1;
    }
    return 0;
}
```

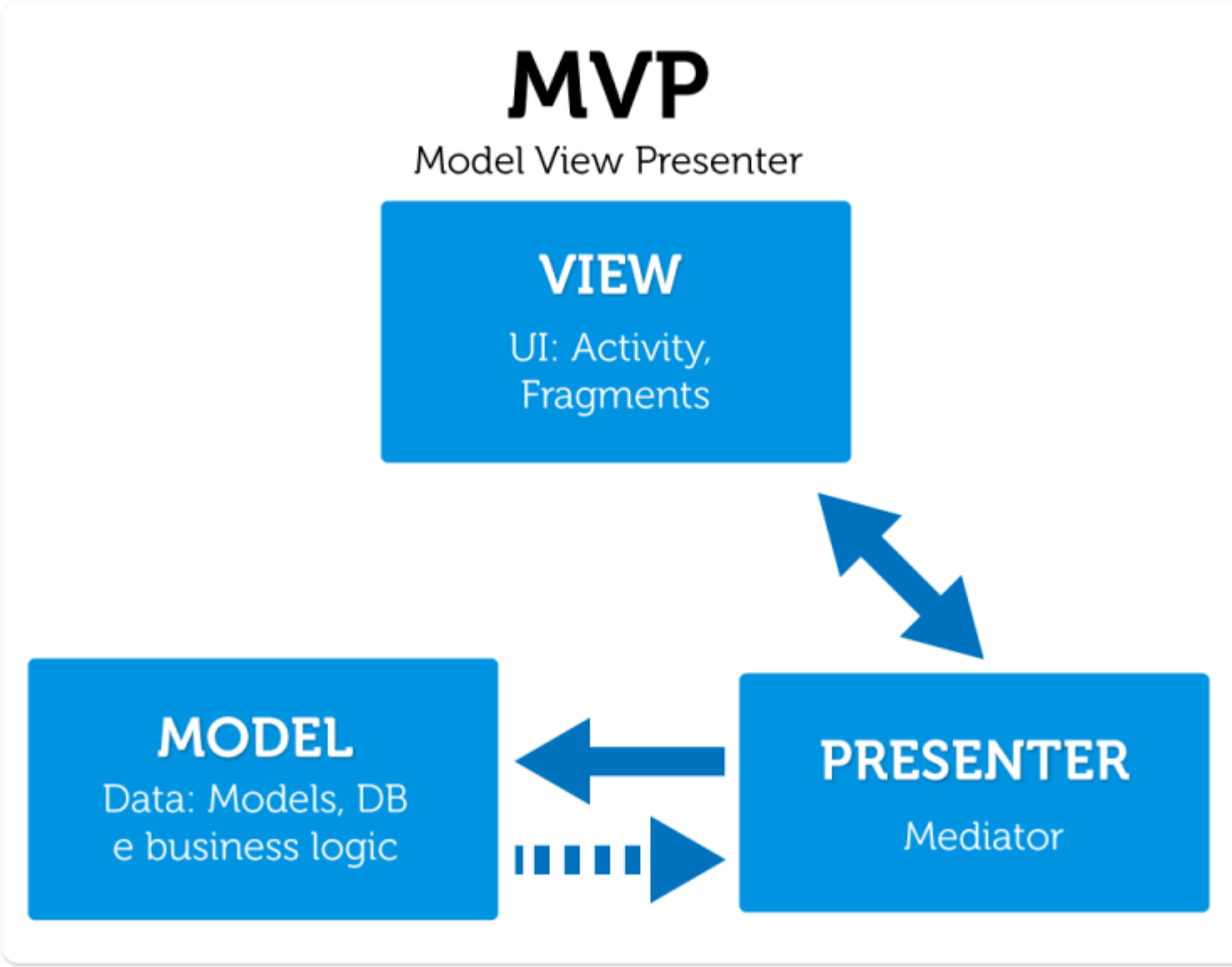
Clase de usuario

```
public interface IUser {
    String getName();

    String getPasswd();

    int checkUserValidity(String name, String passwd);
}
```

MVP

Un modelo-vista-presentador (MVP) es una derivación del modelo arquitectónico modelo-vista-controlador (MVC). Se utiliza principalmente para crear interfaces de usuario y ofrece los siguientes beneficios:

- Las vistas están más separadas de los modelos. El presentador es el mediador entre el modelo y la vista.
- Es más fácil crear pruebas unitarias.
- En general, existe una asignación uno a uno entre View y Presenter, con la posibilidad de usar varios Presenters para vistas complejas.



Lea Arquitectura MVP en línea: https://riptutorial.com/es/android/topic/4615/arquitectura-mvp

Capítulo 29: AsyncTask

Parámetros

Parámetro	Detalles
Parámetros	el tipo de los parámetros enviados a la tarea en la ejecución.
Progreso	El tipo de unidades de progreso publicadas durante el cómputo de fondo.
Resultado	El tipo del resultado del cálculo de fondo.

Examples

Uso básico

En Actividades y servicios de Android, la mayoría de las devoluciones de llamada se ejecutan en el hilo principal . Esto facilita la actualización de la interfaz de usuario, pero la ejecución de tareas pesadas de procesador o de E / S en el subproceso principal puede hacer que su interfaz de usuario se detenga y deje de responder (documentación oficial sobre lo que sucede).

Puedes remediar esto poniendo estas tareas más pesadas en un hilo de fondo.

Una forma de hacerlo es usar una AsyncTask , que proporciona un marco para facilitar el uso de un subproceso en segundo plano, y también realizar tareas de subprocesos en la interfaz de usuario antes, durante y después de que el subproceso en segundo plano haya completado su trabajo.

Métodos que se pueden anular al extender `AsyncTask` :

- `onPreExecute()` : invocado en el **subproceso de** la **interfaz de usuario** antes de que se ejecute la tarea
- `doInBackground()` : se invoca en **el subproceso en segundo plano** inmediatamente después de que `onPreExecute()` termine de ejecutarse.
- `onProgressUpdate()` : se invoca en el **subproceso de** la **interfaz de usuario** después de una llamada a `publishProgress(Progress...)` .
- `onPostExecute()` : invocado en el **subproceso de** la **interfaz de usuario** después de que finalice el cálculo en segundo plano

Ejemplo

```
public class MyCustomAsyncTask extends AsyncTask<File, Void, String> {
```

```
    @Override
    protected void onPreExecute(){
        // This runs on the UI thread before the background thread executes.
        super.onPreExecute();
        // Do pre-thread tasks such as initializing variables.
        Log.v("myBackgroundTask", "Starting Background Task");
    }

    @Override
    protected String doInBackground(File... params) {
        // Disk-intensive work. This runs on a background thread.
        // Search through a file for the first line that contains "Hello", and return
        // that line.
        try (Scanner scanner = new Scanner(params[0])) {
            while (scanner.hasNextLine()) {
                final String line = scanner.nextLine();
                publishProgress(); // tell the UI thread we made progress

                if (line.contains("Hello")) {
                    return line;
                }
            }
            return null;
        }
    }

    @Override
    protected void onProgressUpdate(Void...p) {
        // Runs on the UI thread after publishProgress is invoked
        Log.v("Read another line!")
    }

    @Override
    protected void onPostExecute(String s) {
        // This runs on the UI thread after complete execution of the doInBackground() method
        // This function receives result(String s) returned from the doInBackground() method.
        // Update UI with the found string.
        TextView view = (TextView) findViewById(R.id.found_string);
        if (s != null) {
            view.setText(s);
        } else {
            view.setText("Match not found.");
        }
    }

}
```

Uso:

```
MyCustomAsyncTask asyncTask = new MyCustomAsyncTask<File, Void, String>();
// Run the task with a user supplied filename.
asyncTask.execute(userSuppliedFilename);
```

o simplemente:

```
new MyCustomAsyncTask().execute(userSuppliedFilename);
```

Nota

Al definir una `AsyncTask` podemos pasar tres tipos entre corchetes `< >` .
Definido como `<Params, Progress, Result>` **Parámetros** `<Params, Progress, Result>` (vea la **sección Parámetros**)

En el ejemplo anterior, hemos utilizado los tipos `<File, Void, String>` :

```
AsyncTask<File, Void, String>
// Params has type File
// Progress has unused type
// Result has type String
```

`Void` se utiliza cuando desea marcar un tipo como no utilizado.

Tenga en cuenta que no puede pasar tipos primitivos (es decir, `int` , `float` y otros 6) como parámetros. En tales casos, debe pasar sus clases de envoltorio , por ejemplo, `Integer` lugar de `int` , o `Float` lugar de `float` .

El ciclo de vida de AsyncTask y Activity

AsyncTasks no sigue el ciclo de vida de las instancias de la actividad. Si inicia una AsyncTask dentro de una actividad y gira el dispositivo, la actividad se destruirá y se creará una nueva instancia. Pero la AsyncTask no morirá. Seguirá viviendo hasta que se complete.

Solución: AsyncTaskLoader

Una subclase de cargadores es el AsyncTaskLoader. Esta clase realiza la misma función que AsyncTask, pero mucho mejor. Puede manejar los cambios de configuración de la actividad más fácilmente, y se comporta dentro de los ciclos de vida de Fragmentos y Actividades. Lo bueno es que AsyncTaskLoader se puede usar en cualquier situación en la que se esté utilizando AsyncTask. En cualquier momento, los datos deben cargarse en la memoria para que la Actividad / Fragmento los maneje, AsyncTaskLoader puede hacer el trabajo mejor.

Cancelando AsyncTask

```
YourAsyncTask task = new YourAsyncTask();
task.execute();
task.cancel();
```

Esto no detiene su tarea si estaba en progreso, solo establece el indicador cancelado que puede verificarse verificando el valor de retorno de `isCancelled()` (asumiendo que su código se está ejecutando actualmente) haciendo esto:

```
class YourAsyncTask extends AsyncTask<Void, Void, Void> {
    @Override
    protected Void doInBackground(Void... params) {
        while(!isCancelled()) {
            ... doing long task stuff
```

```
            //Do something, you need, upload part of file, for example
            if (isCancelled()) {
                return null; // Task was detected as canceled
            }
            if (yourTaskCompleted) {
                return null;
            }
        }
    }
}
```

Nota

Si una AsyncTask se cancela mientras `doInBackground(Params... params)` aún se está ejecutando, el método `onPostExecute(Result result)` **NO** se llamará después de que `doInBackground(Params... params)` . AsyncTask llamará a `onCancelled(Result result)` para indicar que la tarea se canceló durante la ejecución.

Progreso de publicación

A veces, necesitamos actualizar el progreso del cálculo realizado por una `AsyncTask` . Este progreso podría representarse por una cadena, un entero, etc. Para hacer esto, tenemos que usar dos funciones. Primero, debemos configurar la función `onProgressUpdate` cuyo tipo de parámetro sea el mismo que el segundo parámetro de tipo de nuestra `AsyncTask` .

```
class YourAsyncTask extends AsyncTask<URL, Integer, Long> {
    @Override
    protected void onProgressUpdate(Integer... args) {
        setProgressPercent(args[0])
    }
}
```

Segundo, tenemos que usar la función `publishProgress` necesariamente en la función `doInBackground` , y eso es todo, el método anterior hará todo el trabajo.

```
protected Long doInBackground(URL... urls) {
     int count = urls.length;
     long totalSize = 0;
     for (int i = 0; i < count; i++) {
         totalSize += Downloader.downloadFile(urls[i]);
         publishProgress((int) ((i / (float) count) * 100));
     }
     return totalSize;
 }
```

Descarga la imagen usando AsyncTask en Android

Este tutorial explica cómo descargar la imagen usando AsyncTask en Android. El siguiente ejemplo descarga la imagen mientras muestra la barra de progreso mientras se descarga.

Entendiendo Android AsyncTask

La tarea asíncrona le permite implementar MultiThreading sin ensuciarse las manos en hilos. AsyncTask permite el uso correcto y fácil del hilo de la interfaz de usuario. Permite realizar operaciones en segundo plano y pasar los resultados en el subproceso de la interfaz de usuario. Si está haciendo algo aislado relacionado con la IU, por ejemplo, descargando datos para presentarlos en una lista, siga adelante y use AsyncTask.

- Las AsyncTasks deberían usarse idealmente para operaciones cortas (unos segundos como máximo).
- Una tarea asíncrona se define mediante 3 tipos genéricos, llamados Parámetros, Progreso y Resultado, y 4 pasos, llamados `onPreExecute()` , `doInBackground()` , `onProgressUpdate()` y `onPostExecute()` .
- En `onPreExecute()` puede definir el código, que debe ejecutarse antes de que comience el procesamiento en segundo plano.
- doInBackground tiene un código que debe ejecutarse en segundo plano, aquí en `doInBackground()` podemos enviar resultados varias veces al hilo de eventos mediante el método publishProgress (), para notificar que se ha completado el procesamiento en segundo plano, podemos devolver los resultados de manera simple.
- `onProgressUpdate()` método `onProgressUpdate()` recibe actualizaciones de progreso del método `doInBackground()` , que se publica a través del método `publishProgress()` , y este método puede usar esta actualización de progreso para actualizar el hilo de eventos
- `onPostExecute()` método `onPostExecute()` maneja los resultados devueltos por el método `doInBackground()` .
- Los tipos genéricos utilizados son
    - Parámetros, el tipo de los parámetros enviados a la tarea en la ejecución
    - Progreso, el tipo de las unidades de progreso publicadas durante el cálculo de fondo.
    - Resultado, el tipo de resultado del cálculo de fondo.
- Si una tarea asíncrona no utiliza ningún tipo, puede marcarse como Tipo de vacío.
- Una tarea asíncrona en ejecución puede cancelarse llamando al método de `cancel(boolean)` .

Descarga de imágenes usando Android AsyncTask

su diseño .xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

<Button
    android:id="@+id/downloadButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
    android:text="Click Here to Download" />

<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:contentDescription="Your image will appear here" />

</LinearLayout>
```

clase .java

```
package com.javatechig.droid;

import java.io.InputStream;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import android.app.Activity;
import android.app.ProgressDialog;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;

public class ImageDownladerActivity extends Activity {

    private ImageView downloadedImg;
    private ProgressDialog simpleWaitDialog;
    private String downloadUrl = "http://www.9ori.com/store/media/images/8ab579a656.jpg";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.asynch);
        Button imageDownloaderBtn = (Button) findViewById(R.id.downloadButton);

        downloadedImg = (ImageView) findViewById(R.id.imageView);

        imageDownloaderBtn.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                new ImageDownloader().execute(downloadUrl);
            }

        });
    }

    private class ImageDownloader extends AsyncTask {

        @Override
```

```
        protected Bitmap doInBackground(String... param) {
            // TODO Auto-generated method stub
            return downloadBitmap(param[0]);
        }

        @Override
        protected void onPreExecute() {
            Log.i("Async-Example", "onPreExecute Called");
            simpleWaitDialog = ProgressDialog.show(ImageDownladerActivity.this,
                    "Wait", "Downloading Image");

        }

        @Override
        protected void onPostExecute(Bitmap result) {
            Log.i("Async-Example", "onPostExecute Called");
            downloadedImg.setImageBitmap(result);
            simpleWaitDialog.dismiss();

        }

        private Bitmap downloadBitmap(String url) {
            // initilize the default HTTP client object
            final DefaultHttpClient client = new DefaultHttpClient();

            //forming a HttpGet request
            final HttpGet getRequest = new HttpGet(url);
            try {

                HttpResponse response = client.execute(getRequest);

                //check 200 OK for success
                final int statusCode = response.getStatusLine().getStatusCode();

                if (statusCode != HttpStatus.SC_OK) {
                    Log.w("ImageDownloader", "Error " + statusCode +
                            " while retrieving bitmap from " + url);
                    return null;

                }

                final HttpEntity entity = response.getEntity();
                if (entity != null) {
                    InputStream inputStream = null;
                    try {
                        // getting contents from the stream
                        inputStream = entity.getContent();

                        // decoding stream data back into image Bitmap that android
understands
                        final Bitmap bitmap = BitmapFactory.decodeStream(inputStream);

                        return bitmap;
                    } finally {
                        if (inputStream != null) {
                            inputStream.close();
                        }
                        entity.consumeContent();
                    }
                }
            } catch (Exception e) {
```

```
                // You Could provide a more explicit error message for IOException
                getRequest.abort();
                Log.e("ImageDownloader", "Something went wrong while" +
                        " retrieving bitmap from " + url + e.toString());
            }

            return null;
        }
    }
}
```

Como actualmente no hay un campo de comentarios para los ejemplos (o no lo he encontrado o no tengo permiso para ello), aquí hay algunos comentarios al respecto:

Este es un buen ejemplo de lo que se puede hacer con AsyncTask.

Sin embargo, el ejemplo actualmente tiene problemas con

- posibles fugas de memoria
- La aplicación se bloquea si se produce una rotación de pantalla poco antes de que finalice la tarea asíncrona.

Para más detalles ver:

- Pase la actividad como WeakReference para evitar pérdidas de memoria
- http://stackoverflow.com/documentation/android/117/asynctask/5377/possible-problems-with-inner-async-tasks
- Evite las actividades con fugas con AsyncTask

Pase la actividad como WeakReference para evitar pérdidas de memoria

Es común que una AsyncTask requiera una referencia a la Actividad que la llamó.

Si la AsyncTask es una clase interna de la Actividad, puede hacer referencia a ella y a cualquier variable / método miembro directamente.

Sin embargo, si la AsyncTask no es una clase interna de la Actividad, deberá pasar una referencia de la Actividad a la AsyncTask. Cuando haga esto, un problema potencial que puede surgir es que AsyncTask mantendrá la referencia de la Actividad hasta que AsyncTask haya completado su trabajo en su hilo de fondo. Si la Actividad finaliza o se cancela antes de que se realice el trabajo de subproceso de fondo de AsyncTask, la AsyncTask seguirá teniendo su referencia a la Actividad y, por lo tanto, no se puede recolectar la basura.

Como resultado, esto causará una pérdida de memoria.

Para evitar que esto suceda, use una WeakReference en la AsyncTask en lugar de tener una referencia directa a la Actividad.

Aquí hay un ejemplo de AsyncTask que utiliza una WeakReference:

```
private class MyAsyncTask extends AsyncTask<String, Void, Void> {

    private WeakReference<Activity> mActivity;

    public MyAsyncTask(Activity activity) {
        mActivity = new WeakReference<Activity>(activity);
    }

    @Override
    protected void onPreExecute() {
        final Activity activity = mActivity.get();
        if (activity != null) {
            ....
        }
    }

    @Override
    protected Void doInBackground(String... params) {
        //Do something
        String param1 = params[0];
        String param2 = params[1];
        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        final Activity activity = mActivity.get();
        if (activity != null) {
            activity.updateUI();
        }
    }
}
```

Llamando a la AsyncTask desde una actividad:

```
new MyAsyncTask(this).execute("param1", "param2");
```

Llamando a la AsyncTask desde un Fragmento:

```
new MyAsyncTask(getActivity()).execute("param1", "param2");
```

Orden de ejecución

Cuando se introdujo por primera vez, las `AsyncTasks` se ejecutaron en serie en un solo hilo de fondo. Comenzando con `DONUT` , esto se cambió a un grupo de subprocesos permitiendo que múltiples tareas funcionen en paralelo. A partir de `HONEYCOMB` , las tareas se ejecutan en un solo hilo para evitar errores comunes de aplicación causados por la ejecución paralela.

Si realmente desea una ejecución paralela, puede invocar `executeOnExecutor(java.util.concurrent.Executor, Object[])` con `THREAD_POOL_EXECUTOR` .

> SERIAL_EXECUTOR -> Un Ejecutor que ejecuta las tareas de una en una en orden serial.

THREAD_POOL_EXECUTOR -> Un Executor que se puede utilizar para ejecutar tareas en paralelo.

muestra:

```
Task task = new Task();
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB)
    task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR, data);
else
    task.execute(data);
```

AsyncTask: Ejecución en serie y ejecución paralela de tareas

AsyncTask es una clase abstracta y no hereda la clase `Thread` . Tiene un método **abstracto** `doInBackground(Params... params)` , que se reemplaza para realizar la tarea. Este método se llama desde `AsyncTask.call()` .

El ejecutor es parte del paquete `java.util.concurrent` .

Por otra parte, AsyncTask contiene 2 `Executor` s

`THREAD_POOL_EXECUTOR`

Utiliza hilos de trabajo para ejecutar las tareas en paralelo.

```
public static final Executor THREAD_POOL_EXECUTOR = new ThreadPoolExecutor(CORE_POOL_SIZE,
MAXIMUM_POOL_SIZE, KEEP_ALIVE, TimeUnit.SECONDS, sPoolWorkQueue, sThreadFactory);
```

`SERIAL_EXECUTOR`

Ejecuta la tarea en serie, es decir, uno por uno.

```
private static class SerialExecutor implements Executor { }
```

Los dos `Executor` son **estáticos** , por lo tanto, solo `THREAD_POOL_EXECUTOR` un objeto `THREAD_POOL_EXECUTOR` y un objeto `SerialExecutor` , pero puede crear varios objetos `AsyncTask` .

Por lo tanto, si intenta realizar varias tareas en segundo plano con el Ejecutor predeterminado (`SerialExecutor`), estas tareas se pondrán en cola y se ejecutarán en serie.

Si intenta realizar varias tareas en segundo plano con `THREAD_POOL_EXECUTOR` , entonces se ejecutarán en paralelo.

Ejemplo:

```
public class MainActivity extends Activity {
    private Button bt;
    private int CountTask = 0;
    private static final String TAG = "AsyncTaskExample";
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    bt = (Button) findViewById(R.id.button);
    bt.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            BackgroundTask backgroundTask = new BackgroundTask ();
            Integer data[] = { ++CountTask, null, null };

            // Task Executed in thread pool ( 1 )
            backgroundTask.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, data);

            // Task executed Serially ( 2 )
            // Uncomment the below code and comment the above code of Thread
            // pool Executor and check
            // backgroundTask.execute(data);
            Log.d(TAG, "Task = " + (int) CountTask + " Task Queued");

        }
    });

}

private class BackgroundTask extends AsyncTask<Integer, Integer, Integer> {
    int taskNumber;

    @Override
    protected Integer doInBackground(Integer... integers) {
        taskNumber = integers[0];

        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        Log.d(TAG, "Task = " + taskNumber + " Task Running in Background");

        publishProgress(taskNumber);
        return null;
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected void onPostExecute(Integer aLong) {
        super.onPostExecute(aLong);
    }

    @Override
    protected void onProgressUpdate(Integer... values) {
        super.onProgressUpdate(values);
        Log.d(TAG, "Task = " + (int) values[0]
                + " Task Execution Completed");
    }
```

```
    }
}
```

Haga clic en el botón varias veces para iniciar una tarea y ver el resultado.

Tarea ejecutada en grupo de subprocesos (1)

Cada tarea tarda 1000 ms en completarse.

En t = 36s, las tareas 2, 3 y 4 se ponen en cola y comienzan a ejecutarse también porque se ejecutan en paralelo.

```
08-02 19:48:35.815: D/AsyncTaskExample(11693): Task = 1 Task Queued
08-02 19:48:35.815: D/AsyncTaskExample(11693): Task = 1 Task Running in Background
08-02 19:48:**36.025**: D/AsyncTaskExample(11693): Task = 2 Task Queued
08-02 19:48:**36.025**: D/AsyncTaskExample(11693): Task = 2 Task Running in Background
08-02 19:48:**36.165**: D/AsyncTaskExample(11693): Task = 3 Task Queued
08-02 19:48:**36.165**: D/AsyncTaskExample(11693): Task = 3 Task Running in Background
08-02 19:48:**36.325**: D/AsyncTaskExample(11693): Task = 4 Task Queued
08-02 19:48:**36.325**: D/AsyncTaskExample(11693): Task = 4 Task Running in Background
08-02 19:48:**36.815**: D/AsyncTaskExample(11693): Task = 1 Task Execution Completed
08-02 19:48:**36.915**: D/AsyncTaskExample(11693): Task = 5 Task Queued
08-02 19:48:**36.915**: D/AsyncTaskExample(11693): Task = 5 Task Running in Background
08-02 19:48:37.025: D/AsyncTaskExample(11693): Task = 2 Task Execution Completed
08-02 19:48:37.165: D/AsyncTaskExample(11693): Task = 3 Task Execution Completed
----------
```

La `Task Executed in thread pool` comentario se `Task Executed in thread pool` (1) y la `Task executed Serially` descomentar se `Task executed Serially` (2).

Haga clic en el botón varias veces para iniciar una tarea y ver el resultado.

Está ejecutando la tarea en serie, por lo que cada tarea se inicia después de que la tarea actual se haya completado. Por lo tanto, cuando se completa la ejecución de la Tarea 1, solo la Tarea 2 comienza a ejecutarse en segundo plano. Viceversa.

```
08-02 19:42:57.505: D/AsyncTaskExample(10299): Task = 1 Task Queued
08-02 19:42:57.505: D/AsyncTaskExample(10299): Task = 1 Task Running in Background
08-02 19:42:57.675: D/AsyncTaskExample(10299): Task = 2 Task Queued
08-02 19:42:57.835: D/AsyncTaskExample(10299): Task = 3 Task Queued
08-02 19:42:58.005: D/AsyncTaskExample(10299): Task = 4 Task Queued
08-02 19:42:58.155: D/AsyncTaskExample(10299): Task = 5 Task Queued
08-02 19:42:58.505: D/AsyncTaskExample(10299): Task = 1 Task Execution Completed
08-02 19:42:58.505: D/AsyncTaskExample(10299): Task = 2 Task Running in Background
08-02 19:42:58.755: D/AsyncTaskExample(10299): Task = 6 Task Queued
08-02 19:42:59.295: D/AsyncTaskExample(10299): Task = 7 Task Queued
08-02 19:42:59.505: D/AsyncTaskExample(10299): Task = 2 Task Execution Completed
08-02 19:42:59.505: D/AsyncTaskExample(10299): Task = 3 Task Running in Background
08-02 19:43:00.035: D/AsyncTaskExample(10299): Task = 8 Task Queued
08-02 19:43:00.505: D/AsyncTaskExample(10299): Task = 3 Task Execution Completed
08-02 19:43:**00.505**: D/AsyncTaskExample(10299): Task = 4 Task Running in Background
08-02 19:43:**01.505**: D/AsyncTaskExample(10299): Task = 4 Task Execution Completed
08-02 19:43:**01.515**: D/AsyncTaskExample(10299): Task = 5 Task Running in Background
08-02 19:43:**02.515**: D/AsyncTaskExample(10299): Task = 5 Task Execution Completed
08-02 19:43:**02.515**: D/AsyncTaskExample(10299): Task = 6 Task Running in Background
```

```
08-02 19:43:**03.515**: D/AsyncTaskExample(10299): Task = 7 Task Running in Background
08-02 19:43:**03.515**: D/AsyncTaskExample(10299): Task = 6 Task Execution Completed
08-02 19:43:04.515: D/AsyncTaskExample(10299): Task = 8 Task Running in Background
08-02 19:43:**04.515**: D/AsyncTaskExample(10299): Task = 7 Task Execution Completed
```

Lea AsyncTask en línea: https://riptutorial.com/es/android/topic/117/asynctask

Capítulo 30: AudioManager

Examples

Solicitud de enfoque de audio transitorio

```
audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

audioManager.requestAudioFocus(audioListener, AudioManager.STREAM_MUSIC,
AudioManager.AUDIOFOCUS_GAIN_TRANSIENT);

changedListener = new AudioManager.OnAudioFocusChangeListener() {
    @Override
    public void onAudioFocusChange(int focusChange) {
        if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
            // You now have the audio focus and may play sound.
            // When the sound has been played you give the focus back.
            audioManager.abandonAudioFocus(changedListener);
        }
    }
}
```

Solicitando Audio Focus

```
audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

audioManager.requestAudioFocus(audioListener, AudioManager.STREAM_MUSIC,
AudioManager.AUDIOFOCUS_GAIN);

changedListener = new AudioManager.OnAudioFocusChangeListener() {
    @Override
    public void onAudioFocusChange(int focusChange) {
        if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
            // You now have the audio focus and may play sound.
        }
        else if (focusChange == AudioManager.AUDIOFOCUS_REQUEST_FAILED) {
            // Handle the failure.
        }
    }
}
```

Lea AudioManager en línea: https://riptutorial.com/es/android/topic/6798/audiomanager

Capítulo 31: Autentificador de Android

Examples

Servicio Autenticador de Cuenta Básico

El sistema de autenticación de cuenta de Android se puede utilizar para que el cliente se autentique con un servidor remoto. Se requieren tres piezas de información:

- Un servicio, activado por `android.accounts.AccountAuthenticator` . Su método `onBind` debería devolver una subclase de `AbstractAccountAuthenticator` .
- Una actividad para solicitar al usuario las credenciales (actividad de inicio de sesión)
- Un archivo de recursos xml para describir la cuenta.

1. El servicio:

Coloque los siguientes permisos en su AndroidManifest.xml:

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
```

Declara el servicio en el archivo manifiesto:

```
<service android:name="com.example.MyAuthenticationService">
    <intent-filter>
        <action android:name="android.accounts.AccountAuthenticator" />
    </intent-filter>
    <meta-data
        android:name="android.accounts.AccountAuthenticator"
        android:resource="@xml/authenticator" />
</service>
```

Tenga en cuenta que `android.accounts.AccountAuthenticator` está incluido dentro de la etiqueta `intent-filter` . El recurso xml (denominado `authenticator` aquí) se especifica en la etiqueta de `meta-data` .

La clase de servicio:

```
public class MyAuthenticationService extends Service {

    private static final Object lock = new Object();
    private MyAuthenticator mAuthenticator;

    public MyAuthenticationService() {
        super();
    }

    @Override
```

```
    public void onCreate() {
        super.onCreate();

        synchronized (lock) {
            if (mAuthenticator == null) {
                mAuthenticator = new MyAuthenticator(this);
            }
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mAuthenticator.getIBinder();
    }

}
```

2. El recurso xml:

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.example.account"
    android:icon="@drawable/appicon"
    android:smallIcon="@drawable/appicon"
    android:label="@string/app_name" />
```

No asigne directamente una cadena a `android:label` o asigne los elementos dibujables que faltan. Se estrellará sin previo aviso.

3. Extienda la clase AbstractAccountAuthenticator:

```
public class MyAuthenticator extends AbstractAccountAuthenticator {

    private Context mContext;

    public MyAuthenticator(Context context) {
        super(context);
        mContext = context;
    }

    @Override
    public Bundle addAccount(AccountAuthenticatorResponse response,
                             String accountType,
                             String authTokenType,
                             String[] requiredFeatures,
                             Bundle options) throws NetworkErrorException {

        Intent intent = new Intent(mContext, LoginActivity.class);
        intent.putExtra(AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE, response);

        Bundle bundle = new Bundle();
        bundle.putParcelable(AccountManager.KEY_INTENT, intent);

        return bundle;
    }

    @Override
    public Bundle confirmCredentials(AccountAuthenticatorResponse response, Account account,
Bundle options) throws NetworkErrorException {
```

```
        return null;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse response, String accountType) {
        return null;
    }

    @Override
    public Bundle getAuthToken(AccountAuthenticatorResponse response, Account account, String
authTokenType, Bundle options) throws NetworkErrorException {
        return null;
    }

    @Override
    public String getAuthTokenLabel(String authTokenType) {
        return null;
    }

    @Override
    public Bundle hasFeatures(AccountAuthenticatorResponse response, Account account, String[]
features) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle updateCredentials(AccountAuthenticatorResponse response, Account account,
String authTokenType, Bundle options) throws NetworkErrorException {
        return null;
    }
}
```

El método `addAccount()` en la clase `AbstractAccountAuthenticator` es importante ya que se llama a este método cuando se agrega una cuenta desde la pantalla "Agregar cuenta" en la configuración debajo de. `AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE` es importante, ya que incluirá el objeto AccountAuthenticatorResponse que se necesita para devolver las claves de la cuenta luego de la verificación exitosa del usuario.

Lea Autentificador de Android en línea: https://riptutorial.com/es/android/topic/6759/autentificador-de-android

Capítulo 32: AutocompletarTextView

Observaciones

Si desea ofrecer sugerencias al usuario cuando escribe un campo de texto editable, puede usar un `AutoCompleteTextView` . Proporciona sugerencias automáticamente cuando el usuario está escribiendo. La lista de sugerencias se muestra en un menú desplegable desde el cual el usuario puede seleccionar una para reemplazar el contenido del cuadro de edición.

Examples

Autocompletar, autocompletar, ver texto

Diseño (layout XML):

```
<AutoCompleteTextView
   android:id="@+id/autoCompleteTextView1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentTop="true"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="65dp"
   android:ems="10" />
```

Busque la vista en el código después de `setContentView()` (o su fragmento o equivalente de vista personalizada):

```
final AutoCompleteTextView myAutoCompleteTextView =
    (AutoCompleteTextView) findViewById(R.id.autoCompleteTextView1);
```

Proporcionar datos codificados a través de un adaptador:

```
String[] countries = getResources().getStringArray(R.array.list_of_countries);
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,countries);
myAutoCompleteTextView.setAdapter(adapter);
```

Consejo: aunque la forma preferida sería proporcionar datos a través de un `Loader` de algún tipo en lugar de una lista codificada como esta.

Autocompletar con CustomAdapter, ClickListener y Filter

Diseño principal: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <AutoCompleteTextView
        android:id="@+id/auto_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:completionThreshold="2"
        android:hint="@string/hint_enter_name" />
</LinearLayout>
```

Diseño de fila row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/lbl_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="16dp"
        android:paddingLeft="8dp"
        android:paddingRight="8dp"
        android:paddingTop="16dp"
        android:text="Medium Text"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</RelativeLayout>
```

strings.xml

```
<resources>
    <string name="hint_enter_name">Enter Name</string>
</resources>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    AutoCompleteTextView txtSearch;
    List<People> mList;
    PeopleAdapter adapter;
    private People selectedPerson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mList = retrievePeople();
        txtSearch = (AutoCompleteTextView) findViewById(R.id.auto_name);
        adapter = new PeopleAdapter(this, R.layout.activity_main, R.id.lbl_name, mList);
        txtSearch.setAdapter(adapter);
        txtSearch.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int pos, long id) {
```

```
                //this is the way to find selected object/item
                selectedPerson = (People) adapterView.getItemAtPosition(pos);
            }
        });
    }

    private List<People> retrievePeople() {
        List<People> list = new ArrayList<People>();
        list.add(new People("James", "Bond", 1));
        list.add(new People("Jason", "Bourne", 2));
        list.add(new People("Ethan", "Hunt", 3));
        list.add(new People("Sherlock", "Holmes", 4));
        list.add(new People("David", "Beckham", 5));
        list.add(new People("Bryan", "Adams", 6));
        list.add(new People("Arjen", "Robben", 7));
        list.add(new People("Van", "Persie", 8));
        list.add(new People("Zinedine", "Zidane", 9));
        list.add(new People("Luis", "Figo", 10));
        list.add(new People("John", "Watson", 11));
        return list;
    }
}
```

Clase de modelo: People.java

```
public class People {

    private String name, lastName;
    private int id;

    public People(String name, String lastName, int id) {
        this.name = name;
        this.lastName = lastName;
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getlastName() {
        return lastName;
    }

    public void setlastName(String lastName) {
        this.lastName = lastName;
    }
```

```
}
```

Clase de adaptador: PeopleAdapter.java

```
public class PeopleAdapter extends ArrayAdapter<People> {

    Context context;
    int resource, textViewResourceId;
    List<People> items, tempItems, suggestions;

    public PeopleAdapter(Context context, int resource, int textViewResourceId, List<People>
items) {
        super(context, resource, textViewResourceId, items);
        this.context = context;
        this.resource = resource;
        this.textViewResourceId = textViewResourceId;
        this.items = items;
        tempItems = new ArrayList<People>(items); // this makes the difference.
        suggestions = new ArrayList<People>();
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View view = convertView;
        if (convertView == null) {
            LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            view = inflater.inflate(R.layout.row, parent, false);
        }
        People people = items.get(position);
        if (people != null) {
            TextView lblName = (TextView) view.findViewById(R.id.lbl_name);
            if (lblName != null)
                lblName.setText(people.getName());
        }
        return view;
    }

    @Override
    public Filter getFilter() {
        return nameFilter;
    }

    /**
     * Custom Filter implementation for custom suggestions we provide.
     */
    Filter nameFilter = new Filter() {
        @Override
        public CharSequence convertResultToString(Object resultValue) {
            String str = ((People) resultValue).getName();
            return str;
        }

        @Override
        protected FilterResults performFiltering(CharSequence constraint) {
            if (constraint != null) {
                suggestions.clear();
                for (People people : tempItems) {
                    if
```

```
(people.getName().toLowerCase().contains(constraint.toString().toLowerCase())) {
                        suggestions.add(people);
                    }
                }
                FilterResults filterResults = new FilterResults();
                filterResults.values = suggestions;
                filterResults.count = suggestions.size();
                return filterResults;
            } else {
                return new FilterResults();
            }
        }

        @Override
        protected void publishResults(CharSequence constraint, FilterResults results) {
            List<People> filterList = (ArrayList<People>) results.values;
            if (results != null && results.count > 0) {
                clear();
                for (People people : filterList) {
                    add(people);
                    notifyDataSetChanged();
                }
            }
        }
    };
}
```

Lea AutocompletarTextView en línea:
https://riptutorial.com/es/android/topic/5300/autocompletartextview

Capítulo 33: Autosize TextViews

Introducción

Un TextView que automáticamente cambia el tamaño del texto para que se ajuste perfectamente a sus límites.

Android O le permite indicar a TextView que permita que el tamaño del texto se expanda o se contraiga automáticamente para completar su diseño según las características y los límites de TextView.

Puede configurar el tamaño automático de TextView en código o XML.

Hay dos formas de configurar TextView de tamaño automático: **granularidad** y **tamaños preestablecidos**

Examples

Granularidad

En Java:

Llame al método `setAutoSizeTextTypeUniformWithConfiguration()` :

```
setAutoSizeTextTypeUniformWithConfiguration(int autoSizeMinTextSize, int autoSizeMaxTextSize,
int autoSizeStepGranularity, int unit)
```

En XML:

Utilice los atributos `autoSizeMinTextSize` , `autoSizeMaxTextSize` y `autoSizeStepGranularity` para establecer las dimensiones de tamaño automático en el archivo XML de diseño:

```
<TextView android:id="@+id/autosizing_textview_presetsize"
        android:layout_width="wrap_content"
        android:layout_height="250dp"
        android:layout_marginLeft="0dp"
        android:layout_marginTop="0dp"
        android:autoSizeMaxTextSize="100sp"
        android:autoSizeMinTextSize="12sp"
        android:autoSizeStepGranularity="2sp"
        android:autoSizeText="uniform"
        android:text="Hello World!"
        android:textSize="100sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Vea la demostración de AutosizingTextViews en GitHub para más detalles.

Tamaños preestablecidos

En Java:

Llame al método `setAutoSizeTextTypeUniformWithPresetSizes()` :

```
setAutoSizeTextTypeUniformWithPresetSizes(int[] presetSizes, int unit)
```

En XML:

Use el atributo `autoSizePresetSizes` en el archivo XML de diseño:

```
<TextView android:id="@+id/autosizing_textview_presetsize"
        android:layout_width="wrap_content"
        android:layout_height="250dp"
        android:layout_marginLeft="0dp"
        android:layout_marginTop="0dp"
        android:autoSizeText="uniform"
        android:autoSizePresetSizes="@array/autosize_text_sizes"
        android:text="Hello World!"
        android:textSize="100sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Para acceder a la matriz como un recurso, defina la matriz en el archivo *res / values / arrays.xml* :

```
<array name="autosize_text_sizes">
    <item>10sp</item>
    <item>12sp</item>
    <item>20sp</item>
    <item>40sp</item>
    <item>100sp</item>
</array>
```

Vea la demostración de AutosizingTextViews en GitHub para más detalles.

Lea Autosize TextViews en línea: https://riptutorial.com/es/android/topic/9652/autosize-textviews

Capítulo 34: Barra de progreso

Observaciones

Documentación oficial: ProgressBar

Examples

Barra de progreso indeterminado

Una barra de progreso indeterminada muestra una animación cíclica sin una indicación de progreso.

Barra de progreso indeterminada básica (rueda giratoria)

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Barra de progreso horizontal indeterminada (barra plana)

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Otros estilos incorporados de ProgressBar

```
style="@android:style/Widget.ProgressBar.Small"
style="@android:style/Widget.ProgressBar.Large"
style="@android:style/Widget.ProgressBar.Inverse"
style="@android:style/Widget.ProgressBar.Small.Inverse"
style="@android:style/Widget.ProgressBar.Large.Inverse"
```

Para usar la barra de progreso indeterminada en una actividad

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setVisibility(View.VISIBLE);
progressBar.setVisibility(View.GONE);
```

Barra de progreso determinada

Una barra de progreso determinada muestra el progreso actual hacia un valor máximo específico.

Barra de progreso horizontal determinada

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="10dp"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Barra de progreso vertical determinada

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="10dp"
    android:layout_height="match_parent"
    android:progressDrawable="@drawable/progress_vertical"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

res / drawable / progress_vertical.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/background">
        <shape>
            <corners android:radius="3dp"/>
            <solid android:color="@android:color/darker_gray"/>
        </shape>
    </item>
    <item android:id="@android:id/secondaryProgress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
                <corners android:radius="3dp"/>
                <solid android:color="@android:color/holo_blue_light"/>
            </shape>
        </clip>
    </item>
    <item android:id="@android:id/progress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
                <corners android:radius="3dp"/>
                <solid android:color="@android:color/holo_blue_dark"/>
            </shape>
        </clip>
    </item>
</layer-list>
```

Anillo determinado ProgressBar

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:progressDrawable="@drawable/progress_ring"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

res / drawable / progress_ring.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/secondaryProgress">
        <shape
            android:shape="ring"
            android:useLevel="true"
            android:thicknessRatio="24"
            android:innerRadiusRatio="2.2">
            <corners android:radius="3dp"/>
            <solid android:color="#0000FF"/>
        </shape>
    </item>

    <item android:id="@android:id/progress">
        <shape
            android:shape="ring"
            android:useLevel="true"
            android:thicknessRatio="24"
            android:innerRadiusRatio="2.2">
            <corners android:radius="3dp"/>
            <solid android:color="#FFFFFF"/>
        </shape>
    </item>
</layer-list>
```

Para utilizar el ProgressBar determinado en una actividad.

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setSecondaryProgress(100);
progressBar.setProgress(10);
progressBar.setMax(100);
```

Barra de progreso personalizada

CustomProgressBarActivity.java :

```
public class CustomProgressBarActivity extends AppCompatActivity {

    private TextView txtProgress;
    private ProgressBar progressBar;
    private int pStatus = 0;
    private Handler handler = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_custom_progressbar);

        txtProgress = (TextView) findViewById(R.id.txtProgress);
        progressBar = (ProgressBar) findViewById(R.id.progressBar);

        new Thread(new Runnable() {
            @Override
            public void run() {
                while (pStatus <= 100) {
```

```
                    handler.post(new Runnable() {
                        @Override
                        public void run() {
                            progressBar.setProgress(pStatus);
                            txtProgress.setText(pStatus + " %");
                        }
                    });
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    pStatus++;
                }
            }
        }).start();

    }
}
```

activity_custom_progressbar.xml :

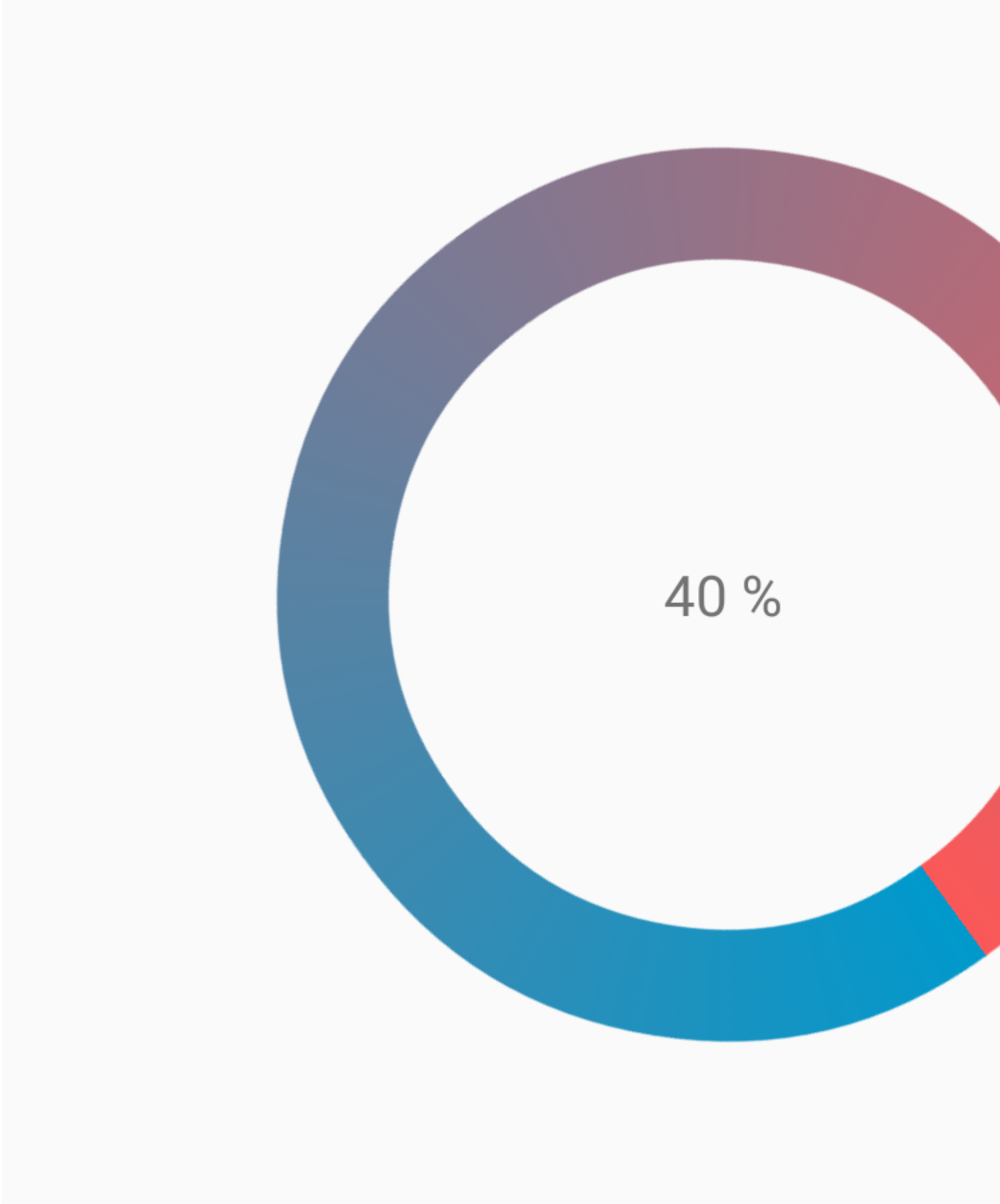
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.skholingua.android.custom_progressbar_circular.MainActivity" >


    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_centerInParent="true"
        android:layout_height="wrap_content">

        <ProgressBar
            android:id="@+id/progressBar"
            style="?android:attr/progressBarStyleHorizontal"
            android:layout_width="250dp"
            android:layout_height="250dp"
            android:layout_centerInParent="true"
            android:indeterminate="false"
            android:max="100"
            android:progress="0"
            android:progressDrawable="@drawable/custom_progressbar_drawable"
            android:secondaryProgress="0" />


        <TextView
            android:id="@+id/txtProgress"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBottom="@+id/progressBar"
            android:layout_centerInParent="true"
            android:textAppearance="?android:attr/textAppearanceSmall" />
    </RelativeLayout>
```

```
</RelativeLayout>
```

custom_progressbar_drawable.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromDegrees="-90"
    android:pivotX="50%"
    android:pivotY="50%"
    android:toDegrees="270" >

    <shape
        android:shape="ring"
        android:useLevel="false" >
        <gradient
            android:centerY="0.5"
            android:endColor="#FA5858"
            android:startColor="#0099CC"
            android:type="sweep"
            android:useLevel="false" />
    </shape>

</rotate>
```

Captura de pantalla de referencia:



Barra de progreso de tintado

Usando un tema de AppCompat, el color de `ProgressBar` será el `colorAccent` que haya definido.

5.0

Para cambiar el color de la `ProgressBar` sin cambiar el color de acento, puede usar el atributo `android:theme` invalida el color de acento:

```
<ProgressBar
    android:theme="@style/MyProgress"
    style="@style/Widget.AppCompat.ProgressBar" />

<!-- res/values/styles.xml -->
<style name="MyProgress" parent="Theme.AppCompat.Light">
    <item name="colorAccent">@color/myColor</item>
</style>
```

Para teñir la Barra de `ProgressBar` , puede usar en el archivo xml los atributos `android:indeterminateTintMode` y `android:indeterminateTint`

```
<ProgressBar
    android:indeterminateTintMode="src_in"
    android:indeterminateTint="@color/my_color"
/>
```

Material Linear ProgressBar

Según la documentación del material :

> Un indicador de progreso lineal siempre debe llenar de 0% a 100% y nunca disminuir en valor.
> Debe representarse con barras en el borde de un encabezado o una hoja que aparecen y desaparecen.

Para usar un material Linear ProgressBar solo use en su xml:

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```



Indeterminado

Para crear ProgressBar **indeterminado,** establezca el atributo `android:indeterminate` en `true` .

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="true"/>
```

Determinado

Para crear una **barra de** progreso **determinada,** establezca el atributo `android:indeterminate` en `false` y use los atributos `android:max` y `android:progress` :

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"/>
```

Solo usa este código para actualizar el valor:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
```

Buffer

Para crear un efecto de **búfer** con la Barra de progreso, establezca el atributo `android:indeterminate` en `false` y use los atributos de `android:max` , `android:progress` y `android:secondaryProgress` :

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"
    android:secondaryProgress="25"/>
```

El valor del búfer está definido por el atributo `android:secondaryProgress` .
Solo usa este código para actualizar los valores:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
progressBar.setSecondaryProgress(50);
```

Indeterminado y determinado

Para obtener este tipo de ProgressBar solo usa una ProgressBar indeterminada usando el atributo `android:indeterminate` como verdadero.

```
<ProgressBar
    android:id="@+id/progressBar"
```

```
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="true"/>
```

Luego, cuando necesite cambiar del progreso indeterminado al progreso determinado, use el método `setIndeterminate()` .

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setIndeterminate(false);
```

Creación de un diálogo de progreso personalizado

Al crear una clase de diálogo de progreso personalizado, el diálogo se puede usar para mostrar en la instancia de la interfaz de usuario, sin volver a crear el diálogo.

Primero crea una clase personalizada de diálogo de progreso.

CustomProgress.java

```
public class CustomProgress {

   public static CustomProgress customProgress = null;
   private Dialog mDialog;

   public static CustomProgress getInstance() {
       if (customProgress == null) {
           customProgress = new CustomProgress();
       }
       return customProgress;
   }

   public void showProgress(Context context, String message, boolean cancelable) {
       mDialog = new Dialog(context);
    // no tile for the dialog
       mDialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
       mDialog.setContentView(R.layout.prograss_bar_dialog);
       mProgressBar = (ProgressBar) mDialog.findViewById(R.id.progress_bar);
    //  mProgressBar.getIndeterminateDrawable().setColorFilter(context.getResources()
    // .getColor(R.color.material_blue_gray_500), PorterDuff.Mode.SRC_IN);
       TextView progressText = (TextView) mDialog.findViewById(R.id.progress_text);
       progressText.setText("" + message);
       progressText.setVisibility(View.VISIBLE);
       mProgressBar.setVisibility(View.VISIBLE);
    // you can change or add this line according to your need
       mProgressBar.setIndeterminate(true);
       mDialog.setCancelable(cancelable);
       mDialog.setCanceledOnTouchOutside(cancelable);
       mDialog.show();
   }

   public void hideProgress() {
       if (mDialog != null) {
           mDialog.dismiss();
           mDialog = null;
       }
   }
}
```

Ahora creando el diseño de progreso personalizado

prograss_bar_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="65dp"
    android:background="@android:color/background_dark"
    android:orientation="vertical">

    <TextView
        android:id="@+id/progress_text"
        android:layout_width="wrap_content"
        android:layout_height="40dp"
        android:layout_above="@+id/progress_bar"
        android:layout_marginLeft="10dp"
        android:layout_marginStart="10dp"
        android:background="@android:color/transparent"
        android:gravity="center_vertical"
        android:text=""
        android:textColor="@android:color/white"
        android:textSize="16sp"
        android:visibility="gone" />

    <-- Where the style can be changed to any kind of ProgressBar -->

    <ProgressBar
        android:id="@+id/progress_bar"
        style="@android:style/Widget.DeviceDefault.ProgressBar.Horizontal"
        android:layout_width="match_parent"
        android:layout_height="30dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_gravity="center"
        android:background="@color/cardview_dark_background"
        android:maxHeight="20dp"
        android:minHeight="20dp" />

</RelativeLayout>
```

Eso es todo. Ahora para llamar al Dialog in Code

```
CustomProgress customProgress = CustomProgress.getInstance();

// now you have the instance of CustomProgres
// for showing the ProgressBar

customProgress.showProgress(#Context, getString(#StringId), #boolean);

// for hiding the ProgressBar

customProgress.hideProgress();
```

Lea Barra de progreso en línea: https://riptutorial.com/es/android/topic/3353/barra-de-progreso

Capítulo 35: Base de datos en tiempo real de Firebase

Observaciones

Otros temas relacionados:

- Base de fuego

Examples

Controlador de eventos Firebase Realtime DataBase

Primero inicialice FirebaseDatabase:

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

Escribe en tu base de datos:

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Lee de tu base de datos:

```
// Read from the database
myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // This method is called once with the initial value and again
        // whenever data at this location is updated.
        String value = dataSnapshot.getValue(String.class);
        Log.d(TAG, "Value is: " + value);
    }

    @Override
    public void onCancelled(DatabaseError error) {
        // Failed to read value
        Log.w(TAG, "Failed to read value.", error.toException());
    }
});
```

Recuperar datos en eventos de Android:

```
ChildEventListener childEventListener = new ChildEventListener() {
    @Override
```

```
    public void onChildAdded(DataSnapshot dataSnapshot, String previousChildName) {
        Log.d(TAG, "onChildAdded:" + dataSnapshot.getKey());
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String previousChildName) {
        Log.d(TAG, "onChildChanged:" + dataSnapshot.getKey());
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
        Log.d(TAG, "onChildRemoved:" + dataSnapshot.getKey());

    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String previousChildName) {
        Log.d(TAG, "onChildMoved:" + dataSnapshot.getKey());

    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        Log.w(TAG, "postComments:onCancelled", databaseError.toException());
        Toast.makeText(mContext, "Failed to load comments.",
                Toast.LENGTH_SHORT).show();
    }
};
ref.addChildEventListener(childEventListener);
```

Configuración rápida

1. Complete la parte de Instalación y configuración para conectar su aplicación a Firebase. Esto creará el proyecto en Firebase.

2. Agregue la dependencia de Firebase Realtime Database a su archivo `build.gradle` nivel de `build.gradle` :

```
compile 'com.google.firebase:firebase-database:10.2.1'
```

3. Configurar las reglas de la base de datos de Firebase

Ahora está listo para trabajar con la base de datos en tiempo real en Android.

Por ejemplo, escribe un mensaje de `Hello World` en la base de datos debajo de la clave de `message` .

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Diseño y comprensión de cómo recuperar datos en tiempo real de la base de

datos de Firebase

Este ejemplo asume que ya ha configurado una base de datos en tiempo real de Firebase. Si eres un iniciador, infórmate aquí sobre cómo agregar Firebase a tu proyecto de Android.

Primero, agregue la dependencia de la base de datos Firebase al archivo *build.gradle de* nivel de aplicación:

```
compile 'com.google.firebase:firebase-database:9.4.0'
```

Ahora, creemos una aplicación de chat que almacene datos en la base de datos de Firebase.

Paso 1: Crea una clase llamada Chat

Solo crea una clase con algunas variables básicas requeridas para el chat:

```
public class Chat{
    public String name, message;
}
```

Paso 2: Crea algunos datos JSON

Para enviar / recuperar datos a / desde la base de datos de Firebase, debe usar JSON. Supongamos que algunos chats ya están almacenados en el nivel raíz en la base de datos. Los datos de estos chats podrían verse como sigue:

```
[
    {
        "name":"John Doe",
        "message":"My first Message"
    },
    {
        "name":"John Doe",
        "message":"Second Message"
    },
    {
        "name":"John Doe",
        "message":"Third Message"
    }
]
```

Paso 3: Añadiendo los oyentes

Hay tres tipos de oyentes. En el siguiente ejemplo usaremos `childEventListener` :

```
DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference() // Referencing the
```

```
root of the database.
        .child("chats"); // Referencing the "chats" node under the root.

chatDb.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        // This function is called for every child id chat in this case, so using the above
        // example, this function is going to be called 3 times.

        // Retrieving the Chat object from this function is simple.
        Chat chat; // Create a null chat object.

        // Use the getValue function in the dataSnapshot and pass the object's class name to
        // which you want to convert and get data. In this case it is Chat.class.
        chat = dataSnapshot.getValue(Chat.class);

        // Now you can use this chat object and add it into an ArrayList or something like
        // that and show it in the recycler view.
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {
        // This function is called when any of the node value is changed, dataSnapshot will
        // get the data with the key of the child, so you can swap the new value with the
        // old one in the ArrayList or something like that.

        // To get the key, use the .getKey() function.
        // To get the value, use code similar to the above one.
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
        // This function is called when any of the child node is removed. dataSnapshot will
        // get the data with the key of the child.

        // To get the key, use the s String parameter .
    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {
        // This function is called when any of the child nodes is moved to a different
position.

        // To get the key, use the s String parameter.
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // If anything goes wrong, this function is going to be called.

        // You can get the exception by using databaseError.toException();
    }
});
```

Paso 4: Agregar datos a la base de datos

Simplemente cree un objeto de clase de chat y agregue los valores de la siguiente manera:

```
Chat chat=new Chat();
chat.name="John Doe";
chat.message="First message from android";
```

Ahora obtenga una referencia al nodo de chats como se hizo en la sesión de recuperación:

```
DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference().child("chats");
```

Antes de comenzar a agregar datos, tenga en cuenta que necesita una referencia más profunda ya que un nodo de chat tiene varios nodos más y agregar un nuevo chat significa agregar un nuevo nodo que contenga los detalles del chat. Podemos generar un nombre nuevo y único del nodo mediante la función `push()` en el objeto `DatabaseReference` , que devolverá otra `DatabaseReference` , que a su vez apunta a un nodo recién formado para insertar los datos de chat.

Ejemplo

```
// The parameter is the chat object that was newly created a few lines above.
chatDb.push().setValue(chat);
```

La función `setValue()` se asegurará de que se `onDataChanged` funciones `onDataChanged` de la aplicación (incluido el mismo dispositivo), que es el oyente adjunto del nodo "chats".

Desnormalización: Estructura de base de datos plana

La desnormalización y una estructura de base de datos plana son necesarias para descargar de manera eficiente llamadas separadas. Con la siguiente estructura, también es posible mantener relaciones bidireccionales. La desventaja de este enfoque es que siempre debe actualizar los datos en varios lugares.

Por ejemplo, imagine una aplicación que le permita al usuario almacenar mensajes para sí mismo (memos).

Estructura de base de datos plana deseada:

```
|--database
  |-- memos
     |-- memokey1
       |-- title: "Title"
       |-- content: "Message"
     |-- memokey2
       |-- title: "Important Title"
       |-- content: "Important Message"
  |-- users
     |-- userKey1
       |-- name: "John Doe"
       |-- memos
         |-- memokey1 : true //The values here don't matter, we only need the keys.
         |-- memokey2 : true
     |-- userKey2
       |-- name: "Max Doe"
```

La clase memo usada.

```
public class Memo {
    private String title, content;
    //getters and setters ...

    //toMap() is necessary for the push process
    private Map<String, Object> toMap() {
        HashMap<String, Object> result = new HashMap<>();
        result.put("title", title);
        result.put("content", content);
        return result;
    }
}
```

Recuperando los memos de un usuario

```
//We need to store the keys and the memos seperately
private ArrayList<String> mKeys = new ArrayList<>();
private ArrayList<Memo> mMemos = new ArrayList<>();

//The user needs to be logged in to retrieve the uid
String currentUserId = FirebaseAuth.getInstance().getCurrentUser().getUid();

//This is the reference to the list of memos a user has
DatabaseReference currentUserMemoReference = FirebaseDatabase.getInstance().getReference()
    .child("users").child(currentUserId).child("memos");

//This is a reference to the list of all memos
DatabaseReference memoReference = FirebaseDatabase.getInstance().getReference()
    .child("memos");

//We start to listen to the users memos,
//this will also retrieve the memos initially
currentUserMemoReference.addChildEventListener(new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot dataSnapshot, String s) {
            //Here we retrieve the key of the memo the user has.
            String key = dataSnapshot.getKey(); //for example memokey1
            //For later manipulations of the lists, we need to store the key in a list
            mKeys.add(key);
            //Now that we know which message belongs to the user,
            //we request it from our memos:
            memoReference.child(key).addValueEventListener(new ValueEventListener() {
                @Override
                    public void onDataChange(DataSnapshot dataSnapshot) {
                         //Here we retrieve our memo:
                         Memo memo = dataSnapshot.getValue(Memo.class);
                         mMemos.add(memo);
                    }

                @Override
                public void onCancelled(DatabaseError databaseError) { }
            });
        }

        @Override
        public void onChildChanged(DataSnapshot dataSnapshot, String s) { }
```

```
        @Override
        public void onChildRemoved(DataSnapshot dataSnapshot) { }

        @Override
        public void onChildMoved(DataSnapshot dataSnapshot, String s) { }

        @Override
        public void onCancelled(DatabaseError databaseError) { }
    }
```

Creando un memo

```
//The user needs to be logged in to retrieve the uid
String currentUserUid = FirebaseAuth.getInstance().getCurrentUser().getUid();

//This is the path to the list of memos a user has
String userMemoPath = "users/" + currentUserUid + "/memos/";

//This is the path to the list of all memos
String memoPath = "memos/";

//We need to retrieve an unused key from the memos reference
DatabaseReference memoReference =
FirebaseDatabase.getInstance().getReference().child("memos");
String key = memoReference.push().getKey();
Memo newMemo = new Memo("Important numbers", "1337, 42, 3.14159265359");

Map<String, Object> childUpdates = new HashMap<>();
//The second parameter **here** (the value) does not matter, it's just that the key exists
childUpdates.put(userMemoPath + key, true);
childUpdates.put(memoPath + key, newMemo.toMap());

FirebaseDatabase.getInstance().getReference().updateChildren(childUpdates);
```

Después de la inserción, o la base de datos se ve así:

```
|--database
  |-- memos
     |-- memokey1
       |-- title: "Title"
       |-- content: "Message"
     |-- memokey2
       |-- title: "Important Title"
       |-- content: "Important Message"
     |-- generatedMemokey3
       |-- title: "Important numbers"
       |-- content: "1337, 42, 3.14159265359"
  |-- users
     |-- userKey1
       |-- name: "John Doe"
       |-- memos
         |-- memokey1 : true //The values here don't matter, we only need the keys.
         |-- memokey2 : true
         |-- generatedMemokey3 : true
     |-- userKey2
       |-- name: "Max Doe"
```

Entendiendo la base de datos JSON de base de fuego

Antes de ensuciarnos las manos con el código, creo que es necesario comprender cómo se almacenan los datos en la base de fuego. A diferencia de las bases de datos relacionales, firebase almacena datos en formato JSON. Piense en cada fila de una base de datos relacional como un objeto JSON (que básicamente es un par de clave-valor desordenado). Por lo tanto, el nombre de la columna se convierte en clave y el valor almacenado en esa columna para una fila en particular es el valor. De esta manera, toda la fila se representa como un objeto JSON y una lista de estos representa una tabla de base de datos completa. El beneficio inmediato que veo para esto es que la modificación del esquema se convierte en una operación mucho más barata en comparación con los RDBMS antiguos. Es más fácil agregar un par de atributos más a un JSON que alterar una estructura de tabla.

Aquí hay un JSON de muestra para mostrar cómo se almacenan los datos en firebase:

```
 {
  "user_base" : {
    "342343" : {
      "email" : "kaushal.xxxxx@gmail.com",
      "authToken" : "some string",
      "name" : "Kaushal",
      "phone" : "+919916xxxxxx",
      "serviceProviderId" : "firebase",
      "signInServiceType" : "google",
    },
    "354895" : {
      "email" : "xxxxx.devil@gmail.com",
      "authToken" : "some string",
      "name" : "devil",
      "phone" : "+919685xxxxxx",
      "serviceProviderId" : "firebase",
      "signInServiceType" : "github"
    },
    "371298" : {
      "email" : "bruce.wayne@wayneinc.com",
      "authToken" : "I am batman",
      "name" : "Bruce Wayne",
      "phone" : "+14085xxxxxx",
      "serviceProviderId" : "firebase",
      "signInServiceType" : "shield"
    }
  },
  "user_prefs": {
    "key1":{
      "data": "for key one"
    },
    "key2":{
      "data": "for key two"
    },
    "key3":{
      "data": "for key three"
    }
  },
  //other structures
}
```

Esto muestra claramente cómo los datos que utilizamos para almacenar en bases de datos relacionales se pueden almacenar en formato JSON. A continuación veamos cómo leer estos datos en dispositivos Android.

Recuperando datos de base de fuego

Voy a asumir que ya sabes acerca de la adición de dependencias de gradle base de fuego en Android Studio. Si no sigues la guía desde aquí . Agrega tu aplicación en la consola firebase, gradle sync android studio después de agregar dependencias. No se necesitan todas las dependencias, solo base de datos firebase y autenticación firebase.

Ahora que sabemos cómo se almacenan los datos y cómo agregar dependencias de Gradle, veamos cómo usar el SDK de Android de base de fuego importado para recuperar datos.

crear una referencia de base de datos de base de fuego

```
DatabaseReference userDBRef = FirebaseDatabase.getInstance().getReference();
// above statement point to base tree
userDBRef = DatabaseReference.getInstance().getReference().child("user_base")
// points to user_base table JSON (see previous section)
```

desde aquí puede encadenar varias llamadas de método child () para señalar los datos que le interesan. Por ejemplo, si los datos se almacenan como se muestra en la sección anterior y desea señalar al usuario de Bruce Wayne, puede usar:

```
DatabaseReference bruceWayneRef = userDBRef.child("371298");
// 371298 is key of bruce wayne user in JSON structure (previous section)
```

O simplemente pase la referencia completa al objeto JSON:

```
DatabaseReference bruceWayneRef = DatabaseReference.getInstance().getReference()
     .child("user_base/371298");
// deeply nested data can also be referenced this way, just put the fully
// qualified path in pattern shown in above code "blah/blah1/blah1-2/blah1-2-3..."
```

Ahora que tenemos la referencia de los datos que queremos obtener, podemos usar oyentes para obtener datos en aplicaciones de Android. A diferencia de las llamadas tradicionales en las que se activan las llamadas de la API REST mediante retrofit o volley, aquí se requiere un simple detector de devolución de llamada para obtener los datos. Firebase SDK llama a los métodos de devolución de llamada y ya está.

Básicamente, puede adjuntar dos tipos de oyentes, uno es ValueEventListener y el otro es ChildEventListener (descrito en la siguiente sección). Para cualquier cambio en los datos bajo el nodo al que tenemos referencias y escuchas agregadas, los escuchas de eventos de valor devuelven la estructura JSON completa y el oyente de eventos hijo devuelve hijos específicos donde ocurrió el cambio. Ambos son útiles a su manera. Para obtener los datos de la base de fuego, podemos agregar uno o más escuchas a una referencia de base de datos de la base de fuego (lista de usuarios DBRef que creamos anteriormente).

Aquí hay un código de ejemplo (explicación del código después del código):

```
userDBRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        User bruceWayne = dataSnapshot.child("371298").getValue(User.class);
        // Do something with the retrieved data or Bruce Wayne
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        Log.e("UserListActivity", "Error occured");
        // Do something about the error
    });
```

¿Notaste que el tipo de clase pasó? DataSnapshot puede convertir datos JSON en nuestros POJO definidos, simplemente pase el tipo de clase correcto.

Si su caso de uso no requiere todos los datos (en nuestra tabla user_base) cada vez que ocurre un pequeño cambio o dice que desea ***obtener los datos solo una vez*** , puede usar el ***método addListenerForSingleValueEvent ()*** de referencia de la base de datos. Esto dispara la devolución de llamada sólo una vez.

```
userDBRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // Do something
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Do something about the error
    });
```

Las muestras anteriores le darán el valor del nodo JSON. Para obtener la clave simplemente llame:

```
String myKey = dataSnapshot.getKey();
```

Escuchando actualizaciones de niños

Tome un caso de uso, como una aplicación de chat o una aplicación de lista de compras colaborativa (que básicamente requiere una lista de objetos para sincronizar a los usuarios). Si usa la base de datos de base de fuego y agrega un detector de eventos de valor al nodo primario del chat o al nodo primario de la lista de la compra, terminará con la estructura completa del chat desde el principio del tiempo (me refiero al comienzo del chat) cada vez que se agregue un nodo del chat (es decir, cualquiera dice hola). Si no queremos hacerlo, lo que nos interesa es solo el nuevo nodo o solo el nodo anterior que se eliminó o modificó, los que no se han modificado no deben devolverse.

En este caso podemos usar ChildEvenListener . Sin más adiós, aquí hay un ejemplo de código (ver las secciones previas para datos de muestra JSON):

```
userDBRef.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {
        //If not dealing with ordered data forget about this
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
    });
```

Los nombres de los métodos son auto explicativos. Como puede ver, cada vez que se agrega un nuevo usuario o se modifica alguna propiedad del usuario existente o se elimina o elimina el usuario, se llama al método de devolución de llamada apropiado del oyente de eventos secundarios con datos relevantes. Por lo tanto, si mantiene la interfaz de usuario actualizada para, por ejemplo, la aplicación de chat, obtenga el JSON de onChildAdded () parse en POJO y colóquelo en su interfaz de usuario. Solo recuerda eliminar a tu oyente cuando el usuario salga de la pantalla.

onChildChanged () proporciona todo el valor secundario con propiedades modificadas (nuevas).

onChiledRemoved () devuelve el nodo secundario eliminado.

Recuperando datos con paginación

Cuando tiene una gran base de datos JSON, agregar un valor de escucha de eventos no tiene sentido. Devolverá el enorme JSON y analizarlo llevaría mucho tiempo. En tales casos, podemos usar la paginación y obtener parte de los datos y mostrarlos o procesarlos. Algo así como la carga perezosa o como buscar chats antiguos cuando el usuario hace clic en mostrar chat anterior. En este caso se puede utilizar la consulta .

Tomemos nuestro ejemplo anterior en secciones anteriores. La base de usuarios contiene 3 usuarios, si crece hasta decir 3 cientos mil usuarios y desea obtener la lista de usuarios en lotes de 50:

```
// class level
final int limit = 50;
int start = 0;

// event level
Query userListQuery = userDBRef.orderByChild("email").limitToFirst(limit)
        .startAt(start)
userListQuery.addValueEventListener(new ValueEventListener() {
```

```
    @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
        // Do something
        start += (limit+1);
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Do something about the error
    });
```

Aquí se pueden agregar y escuchar eventos de valor o secundarios. Vuelva a llamar a la consulta para obtener los próximos 50. **Asegúrese de agregar el método orderByChild ()** , esto no funcionará sin eso. Firebase necesita saber el orden por el cual está paginando.

Lea Base de datos en tiempo real de Firebase en línea:
https://riptutorial.com/es/android/topic/5511/base-de-datos-en-tiempo-real-de-firebase

Capítulo 36: Base de fuego

Introducción

Firebase es una plataforma de aplicaciones web y móviles con herramientas e infraestructura diseñadas para ayudar a los desarrolladores a crear aplicaciones de alta calidad.

Caracteristicas

Firebase Cloud Messaging, Firebase Auth, Base de datos en tiempo real, Firebase Storage, Firebase Hosting, Firebase Test Lab para Android, Firebase Crash Reporting.

Observaciones

Firebase - Documentación extendida:

Hay otra etiqueta donde puede encontrar más temas y ejemplos sobre el uso de Firebase.

Otros temas relacionados:

- Base de datos en tiempo real de Firebase
- Indexación de la aplicación Firebase
- Firebase Crash Reporting
- Firebase Cloud Messaging

Examples

Crear un usuario de Firebase

```
public class SignUpActivity extends BaseAppCompatActivity {

    @BindView(R.id.tIETSignUpEmail)
    EditText mEditEmail;
    @BindView(R.id.tIETSignUpPassword)
    EditText mEditPassword;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    }

    @OnClick(R.id.btnSignUpSignUp)
    void signUp() {

        FormValidationUtils.clearErrors(mEditEmail, mEditPassword);

        if (FormValidationUtils.isBlank(mEditEmail)) {
```

```
            mEditEmail.setError("Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditEmail)) {
            mEditEmail.setError("Please enter valid email");
            return;
        }

        if (TextUtils.isEmpty(mEditPassword.getText())) {
            mEditPassword.setError("Please enter password");
            return;
        }

        createUserWithEmailAndPassword(mEditEmail.getText().toString(),
mEditPassword.getText().toString());
    }

    private void createUserWithEmailAndPassword(String email, String password) {
        DialogUtils.showProgressDialog(this, "", getString(R.string.str_creating_account),
false);
        mFirebaseAuth
                .createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (!task.isSuccessful()) {
                            Toast.makeText(SignUpActivity.this,
task.getException().getMessage(),
                                    Toast.LENGTH_SHORT).show();
                            DialogUtils.dismissProgressDialog();
                        } else {
                            Toast.makeText(SignUpActivity.this,
R.string.str_registration_successful, Toast.LENGTH_SHORT).show();
                            DialogUtils.dismissProgressDialog();
                            startActivity(new Intent(SignUpActivity.this,
HomeActivity.class));
                        }
                    }
                });
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_sign_up;
    }
}
```

Iniciar sesión en Firebase usuario con correo electrónico y contraseña

```
public class LoginActivity extends BaseAppCompatActivity {

    @BindView(R.id.tIETLoginEmail)
    EditText mEditEmail;
    @BindView(R.id.tIETLoginPassword)
    EditText mEditPassword;

    @Override
    protected void onResume() {
```

```
        super.onResume();
        FirebaseUser firebaseUser = mFirebaseAuth.getCurrentUser();
        if (firebaseUser != null)
            startActivity(new Intent(this, HomeActivity.class));
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_login;
    }

    @OnClick(R.id.btnLoginLogin)
    void onSignInClick() {

        FormValidationUtils.clearErrors(mEditEmail, mEditPassword);

        if (FormValidationUtils.isBlank(mEditEmail)) {
            FormValidationUtils.setError(null, mEditEmail, "Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditEmail)) {
            FormValidationUtils.setError(null, mEditEmail, "Please enter valid email");
            return;
        }

        if (TextUtils.isEmpty(mEditPassword.getText())) {
            FormValidationUtils.setError(null, mEditPassword, "Please enter password");
            return;
        }

        signInWithEmailAndPassword(mEditEmail.getText().toString(),
mEditPassword.getText().toString());
    }

    private void signInWithEmailAndPassword(String email, String password) {
        DialogUtils.showProgressDialog(this, "", getString(R.string.sign_in), false);
        mFirebaseAuth
                .signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {

                        DialogUtils.dismissProgressDialog();

                        if (task.isSuccessful()) {
                            Toast.makeText(LoginActivity.this, "Login Successful",
Toast.LENGTH_SHORT).show();
                            startActivity(new Intent(LoginActivity.this, HomeActivity.class));
                            finish();
                        } else {
                            Toast.makeText(LoginActivity.this,
task.getException().getMessage(),
                                    Toast.LENGTH_SHORT).show();
                        }
                    }
                });
    }

    @OnClick(R.id.btnLoginSignUp)
    void onSignUpClick() {
```

```
        startActivity(new Intent(this, SignUpActivity.class));
    }


    @OnClick(R.id.btnLoginForgotPassword)
    void forgotPassword() {
        startActivity(new Intent(this, ForgotPasswordActivity.class));
    }
}
```

Enviar correo electrónico de restablecimiento de contraseña de Firebase

```
public class ForgotPasswordActivity extends AppCompatActivity {

    @BindView(R.id.tIETForgotPasswordEmail)
    EditText mEditEmail;
    private FirebaseAuth mFirebaseAuth;
    private FirebaseAuth.AuthStateListener mAuthStateListener;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_forgot_password);
        ButterKnife.bind(this);

        mFirebaseAuth = FirebaseAuth.getInstance();

        mAuthStateListener = new FirebaseAuth.AuthStateListener() {
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
                FirebaseUser firebaseUser = firebaseAuth.getCurrentUser();
                if (firebaseUser != null) {
                    // Do whatever you want with the UserId by firebaseUser.getUid()
                } else {

                }
            }
        };
    }

    @Override
    protected void onStart() {
        super.onStart();
        mFirebaseAuth.addAuthStateListener(mAuthStateListener);
    }

    @Override
    protected void onStop() {
        super.onStop();
        if (mAuthStateListener != null) {
            mFirebaseAuth.removeAuthStateListener(mAuthStateListener);
        }
    }

    @OnClick(R.id.btnForgotPasswordSubmit)
    void onSubmitClick() {

        if (FormValidationUtils.isBlank(mEditEmail)) {
            FormValidationUtils.setError(null, mEditEmail, "Please enter email");
```

```
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditEmail)) {
            FormValidationUtils.setError(null, mEditEmail, "Please enter valid email");
            return;
        }

        DialogUtils.showProgressDialog(this, "", "Please wait...", false);
        mFirebaseAuth.sendPasswordResetEmail(mEditEmail.getText().toString())
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            Toast.makeText(ForgotPasswordActivity.this, "An email has been
sent to you.", Toast.LENGTH_SHORT).show();
                            finish();
                        } else {
                            Toast.makeText(ForgotPasswordActivity.this,
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                        }
                    }
                });
    }
}
```

Actualización del correo electrónico de un usuario de Firebase

```
public class ChangeEmailActivity extends BaseAppCompatActivity implements
ReAuthenticateDialogFragment.OnReauthenticateSuccessListener {

    @BindView(R.id.et_change_email)
    EditText mEditText;
    private FirebaseUser mFirebaseUser;

    @OnClick(R.id.btn_change_email)
    void onChangeEmailClick() {

        FormValidationUtils.clearErrors(mEditText);

        if (FormValidationUtils.isBlank(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter valid email");
            return;
        }

        changeEmail(mEditText.getText().toString());
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mFirebaseUser = mFirebaseAuth.getCurrentUser();
```

```
    }

    private void changeEmail(String email) {
        DialogUtils.showProgressDialog(this, "Changing Email", "Please wait...", false);
        mFirebaseUser.updateEmail(email)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            showToast("Email updated successfully.");
                            return;
                        }

                        if (task.getException() instanceof
FirebaseAuthRecentLoginRequiredException) {
                            FragmentManager fm = getSupportFragmentManager();
                            ReAuthenticateDialogFragment reAuthenticateDialogFragment = new
ReAuthenticateDialogFragment();
                            reAuthenticateDialogFragment.show(fm,
reAuthenticateDialogFragment.getClass().getSimpleName());
                        }
                    }
                });
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_change_email;
    }

    @Override
    public void onReauthenticateSuccess() {
        changeEmail(mEditText.getText().toString());
    }
}
```

Cambia la contraseña

```
public class ChangePasswordActivity extends BaseAppCompatActivity implements
ReAuthenticateDialogFragment.OnReauthenticateSuccessListener {
    @BindView(R.id.et_change_password)
    EditText mEditText;
    private FirebaseUser mFirebaseUser;

    @OnClick(R.id.btn_change_password)
    void onChangePasswordClick() {

        FormValidationUtils.clearErrors(mEditText);

        if (FormValidationUtils.isBlank(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter password");
            return;
        }

        changePassword(mEditText.getText().toString());
    }

    private void changePassword(String password) {
```

```
        DialogUtils.showProgressDialog(this, "Changing Password", "Please wait...", false);
        mFirebaseUser.updatePassword(password)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            showToast("Password updated successfully.");
                            return;
                        }

                        if (task.getException() instanceof
FirebaseAuthRecentLoginRequiredException) {
                            FragmentManager fm = getSupportFragmentManager();
                            ReAuthenticateDialogFragment reAuthenticateDialogFragment = new
ReAuthenticateDialogFragment();
                            reAuthenticateDialogFragment.show(fm,
reAuthenticateDialogFragment.getClass().getSimpleName());
                        }
                    }
                });
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mFirebaseUser = mFirebaseAuth.getCurrentUser();
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_change_password;
    }

    @Override
    public void onReauthenticateSuccess() {
        changePassword(mEditText.getText().toString());
    }
}
```

Volver a autenticar al usuario de Firebase

```
public class ReAuthenticateDialogFragment extends DialogFragment {

    @BindView(R.id.et_dialog_reauthenticate_email)
    EditText mEditTextEmail;
    @BindView(R.id.et_dialog_reauthenticate_password)
    EditText mEditTextPassword;
    private OnReauthenticateSuccessListener mOnReauthenticateSuccessListener;

    @OnClick(R.id.btn_dialog_reauthenticate)
    void onReauthenticateClick() {

        FormValidationUtils.clearErrors(mEditTextEmail, mEditTextPassword);

        if (FormValidationUtils.isBlank(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "Please enter email");
            return;
```

```
        }

        if (!FormValidationUtils.isEmailValid(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "Please enter valid email");
            return;
        }

        if (TextUtils.isEmpty(mEditTextPassword.getText())) {
            FormValidationUtils.setError(null, mEditTextPassword, "Please enter password");
            return;
        }

        reauthenticateUser(mEditTextEmail.getText().toString(),
mEditTextPassword.getText().toString());
    }

    private void reauthenticateUser(String email, String password) {
        DialogUtils.showProgressDialog(getActivity(), "Re-Authenticating", "Please wait...",
false);
        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        AuthCredential authCredential = EmailAuthProvider.getCredential(email, password);
        firebaseUser.reauthenticate(authCredential)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            mOnReauthenticateSuccessListener.onReauthenticateSuccess();
                            dismiss();
                        } else {
                            ((BaseAppCompatActivity)
getActivity()).showToast(task.getException().getMessage());
                        }
                    }
                });
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        mOnReauthenticateSuccessListener = (OnReauthenticateSuccessListener) context;
    }

    @OnClick(R.id.btn_dialog_reauthenticate_cancel)
    void onCancelClick() {
        dismiss();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.dialog_reauthenticate, container);
        ButterKnife.bind(this, view);
        return view;
    }

    @Override
    public void onResume() {
        super.onResume();
        Window window = getDialog().getWindow();
        window.setLayout(WindowManager.LayoutParams.MATCH_PARENT,
```

```
WindowManager.LayoutParams.WRAP_CONTENT);
    }

    interface OnReauthenticateSuccessListener {
        void onReauthenticateSuccess();
    }
}
```

Operaciones de almacenamiento de Firebase

Con este ejemplo, podrás realizar las siguientes operaciones:

1. Conectar a Firebase Storage
2. Crear un directorio llamado "imágenes"
3. Subir un archivo en el directorio de imágenes
4. Descarga un archivo del directorio de imágenes
5. Eliminar un archivo del directorio de imágenes

```
public class MainActivity extends AppCompatActivity {

    private static final int REQUEST_CODE_PICK_IMAGE = 1;
    private static final int PERMISSION_READ_WRITE_EXTERNAL_STORAGE = 2;

    private FirebaseStorage mFirebaseStorage;
    private StorageReference mStorageReference;
    private StorageReference mStorageReferenceImages;
    private Uri mUri;
    private ImageView mImageView;
    private ProgressDialog mProgressDialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        mImageView = (ImageView) findViewById(R.id.imageView);
        setSupportActionBar(toolbar);

        // Create an instance of Firebase Storage
        mFirebaseStorage = FirebaseStorage.getInstance();
    }

    private void pickImage() {
        Intent intent = new Intent(Intent.ACTION_PICK,
android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
        intent.addFlags(Intent.FLAG_GRANT_WRITE_URI_PERMISSION);
        startActivityForResult(intent, REQUEST_CODE_PICK_IMAGE);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == RESULT_OK) {
            if (requestCode == REQUEST_CODE_PICK_IMAGE) {
                String filePath = FileUtil.getPath(this, data.getData());
                mUri = Uri.fromFile(new File(filePath));
```

```
                uploadFile(mUri);
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == PERMISSION_READ_WRITE_EXTERNAL_STORAGE) {
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                pickImage();
            }
        }
    }

    private void showProgressDialog(String title, String message) {
        if (mProgressDialog != null && mProgressDialog.isShowing())
            mProgressDialog.setMessage(message);
        else
            mProgressDialog = ProgressDialog.show(this, title, message, true, false);
    }

    private void hideProgressDialog() {
        if (mProgressDialog != null && mProgressDialog.isShowing()) {
            mProgressDialog.dismiss();
        }
    }

    private void showToast(String message) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    }

    public void showHorizontalProgressDialog(String title, String body) {

        if (mProgressDialog != null && mProgressDialog.isShowing()) {
            mProgressDialog.setTitle(title);
            mProgressDialog.setMessage(body);
        } else {
            mProgressDialog = new ProgressDialog(this);
            mProgressDialog.setTitle(title);
            mProgressDialog.setMessage(body);
            mProgressDialog.setIndeterminate(false);
            mProgressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
            mProgressDialog.setProgress(0);
            mProgressDialog.setMax(100);
            mProgressDialog.setCancelable(false);
            mProgressDialog.show();
        }
    }

    public void updateProgress(int progress) {
        if (mProgressDialog != null && mProgressDialog.isShowing()) {
            mProgressDialog.setProgress(progress);
        }
    }

    /**
     * Step 1: Create a Storage
     *
     * @param view
```

```
     */
    public void onCreateReferenceClick(View view) {
        mStorageReference =
mFirebaseStorage.getReferenceFromUrl("gs://**something**.appspot.com");
        showToast("Reference Created Successfully.");
        findViewById(R.id.button_step_2).setEnabled(true);
    }

    /**
     * Step 2: Create a directory named "Images"
     *
     * @param view
     */
    public void onCreateDirectoryClick(View view) {
        mStorageReferenceImages = mStorageReference.child("images");
        showToast("Directory 'images' created Successfully.");
        findViewById(R.id.button_step_3).setEnabled(true);
    }

    /**
     * Step 3: Upload an Image File and display it on ImageView
     *
     * @param view
     */
    public void onUploadFileClick(View view) {
        if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED ||
ActivityCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED)
            ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE,
Manifest.permission.WRITE_EXTERNAL_STORAGE}, PERMISSION_READ_WRITE_EXTERNAL_STORAGE);
        else {
            pickImage();
        }
    }

    /**
     * Step 4: Download an Image File and display it on ImageView
     *
     * @param view
     */
    public void onDownloadFileClick(View view) {
        downloadFile(mUri);
    }

    /**
     * Step 5: Delete am Image File and remove Image from ImageView
     *
     * @param view
     */
    public void onDeleteFileClick(View view) {
        deleteFile(mUri);
    }

    private void showAlertDialog(Context ctx, String title, String body,
DialogInterface.OnClickListener okListener) {

        if (okListener == null) {
            okListener = new DialogInterface.OnClickListener() {
```

```
                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }
            };
        }

        AlertDialog.Builder builder = new
AlertDialog.Builder(ctx).setMessage(body).setPositiveButton("OK",
okListener).setCancelable(false);

        if (!TextUtils.isEmpty(title)) {
            builder.setTitle(title);
        }

        builder.show();
    }

    private void uploadFile(Uri uri) {
        mImageView.setImageResource(R.drawable.placeholder_image);

        StorageReference uploadStorageReference =
mStorageReferenceImages.child(uri.getLastPathSegment());
        final UploadTask uploadTask = uploadStorageReference.putFile(uri);
        showHorizontalProgressDialog("Uploading", "Please wait...");
        uploadTask
                .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                        hideProgressDialog();
                        Uri downloadUrl = taskSnapshot.getDownloadUrl();
                        Log.d("MainActivity", downloadUrl.toString());
                        showAlertDialog(MainActivity.this, "Upload Complete",
downloadUrl.toString(), new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialogInterface, int i) {
                                findViewById(R.id.button_step_3).setEnabled(false);
                                findViewById(R.id.button_step_4).setEnabled(true);
                            }
                        });

                        Glide.with(MainActivity.this)
                                .load(downloadUrl)
                                .into(mImageView);
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception exception) {
                        exception.printStackTrace();
                        // Handle unsuccessful uploads
                        hideProgressDialog();
                    }
                })
                .addOnProgressListener(MainActivity.this, new
OnProgressListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onProgress(UploadTask.TaskSnapshot taskSnapshot) {
                        int progress = (int) (100 * (float) taskSnapshot.getBytesTransferred()
/ taskSnapshot.getTotalByteCount());
                        Log.i("Progress", progress + "");
                        updateProgress(progress);
```

```
                    }
                });
    }

    private void downloadFile(Uri uri) {
        mImageView.setImageResource(R.drawable.placeholder_image);
        final StorageReference storageReferenceImage =
mStorageReferenceImages.child(uri.getLastPathSegment());
        File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
                Environment.DIRECTORY_PICTURES), "Firebase Storage");
        if (!mediaStorageDir.exists()) {
            if (!mediaStorageDir.mkdirs()) {
                Log.d("MainActivity", "failed to create Firebase Storage directory");
            }
        }

        final File localFile = new File(mediaStorageDir, uri.getLastPathSegment());
        try {
            localFile.createNewFile();
        } catch (IOException e) {
            e.printStackTrace();
        }

        showHorizontalProgressDialog("Downloading", "Please wait...");
        storageReferenceImage.getFile(localFile).addOnSuccessListener(new
OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {
                hideProgressDialog();
                showAlertDialog(MainActivity.this, "Download Complete",
localFile.getAbsolutePath(), new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        findViewById(R.id.button_step_4).setEnabled(false);
                        findViewById(R.id.button_step_5).setEnabled(true);
                    }
                });

                Glide.with(MainActivity.this)
                        .load(localFile)
                        .into(mImageView);
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception exception) {
                // Handle any errors
                hideProgressDialog();
                exception.printStackTrace();
            }
        }).addOnProgressListener(new OnProgressListener<FileDownloadTask.TaskSnapshot>() {
            @Override
            public void onProgress(FileDownloadTask.TaskSnapshot taskSnapshot) {
                int progress = (int) (100 * (float) taskSnapshot.getBytesTransferred() /
taskSnapshot.getTotalByteCount());
                Log.i("Progress", progress + "");
                updateProgress(progress);
            }
        });
    }

    private void deleteFile(Uri uri) {
```

```
        showProgressDialog("Deleting", "Please wait...");
        StorageReference storageReferenceImage =
mStorageReferenceImages.child(uri.getLastPathSegment());
        storageReferenceImage.delete().addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                hideProgressDialog();
                showAlertDialog(MainActivity.this, "Success", "File deleted successfully.",
new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        mImageView.setImageResource(R.drawable.placeholder_image);
                        findViewById(R.id.button_step_3).setEnabled(true);
                        findViewById(R.id.button_step_4).setEnabled(false);
                        findViewById(R.id.button_step_5).setEnabled(false);
                    }
                });
                File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
                        Environment.DIRECTORY_PICTURES), "Firebase Storage");
                if (!mediaStorageDir.exists()) {
                    if (!mediaStorageDir.mkdirs()) {
                        Log.d("MainActivity", "failed to create Firebase Storage directory");
                    }
                }
                deleteFiles(mediaStorageDir);
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception exception) {
                hideProgressDialog();
                exception.printStackTrace();
            }
        });
    }

    private void deleteFiles(File directory) {
        if (directory.isDirectory())
            for (File child : directory.listFiles())
                child.delete();
    }
}
```

De forma predeterminada, las reglas de almacenamiento de Firebase aplican la restricción de autenticación. Si el usuario está autenticado, solo entonces, puede realizar operaciones en Firebase Storage, de lo contrario no podrá hacerlo. He deshabilitado la parte de autenticación en esta demostración actualizando las reglas de almacenamiento. Anteriormente, las reglas parecían:

```
 service firebase.storage {
   match /b/**something**.appspot.com/o {
     match /{allPaths=**} {
       allow read, write: if request.auth != null;
     }
   }
 }
```

Pero he cambiado para saltar la autenticación:

```
service firebase.storage {
  match /b/**something**.appspot.com/o {
    match /{allPaths=**} {
      allow read, write;
    }
  }
}
```

Firebase Cloud Messaging

En primer lugar, debe configurar su proyecto agregando Firebase a su proyecto de Android siguiendo los pasos descritos en este tema .

Configurar Firebase y el FCM SDK

Agregue la dependencia FCM a su archivo `build.gradle` nivel de `build.gradle`

```
dependencies {
 compile 'com.google.firebase:firebase-messaging:11.0.4'
}
```

Y en la parte inferior (esto es importante) agregue:

```
// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

Edita tu manifiesto de aplicación

Agregue lo siguiente al manifiesto de su aplicación:

- Un servicio que amplía `FirebaseMessagingService` . Esto es necesario si desea realizar cualquier manejo de mensajes más allá de recibir notificaciones en aplicaciones en segundo plano.
- Un servicio que extiende `FirebaseInstanceIdService` para manejar la creación, rotación y actualización de tokens de registro.

Por ejemplo:

```
    <service
        android:name=".MyInstanceIdListenerService">
        <intent-filter>
            <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
        </intent-filter>
    </service>
    <service
        android:name=".MyFcmListenerService">
        <intent-filter>
            <action android:name="com.google.firebase.MESSAGING_EVENT" />
```

```
        </intent-filter>
    </service>
```

Aquí están las implementaciones simples de los 2 servicios.

Para recuperar el token de registro actual, extienda la clase `FirebaseInstanceIdService` y anule el método `onTokenRefresh()` :

```
public class MyInstanceIdListenerService extends FirebaseInstanceIdService {

    // Called if InstanceID token is updated. Occurs if the security of the previous token had
been
    // compromised. This call is initiated by the InstanceID provider.
    @Override
    public void onTokenRefresh() {
        // Get updated InstanceID token.
        String refreshedToken = FirebaseInstanceId.getInstance().getToken();

        // Send this token to your server or store it locally
    }
}
```

Para recibir mensajes, use un servicio que extienda `FirebaseMessagingService` y anule el método `onMessageReceived` .

```
public class MyFcmListenerService extends FirebaseMessagingService {

    /**
     * Called when message is received.
     *
     * @param remoteMessage Object representing the message received from Firebase Cloud
Messaging.
     */
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        String from = remoteMessage.getFrom();

        // Check if message contains a data payload.
        if (remoteMessage.getData().size() > 0) {
            Log.d(TAG, "Message data payload: " + remoteMessage.getData());
            Map<String, String> data = remoteMessage.getData();
        }

        // Check if message contains a notification payload.
        if (remoteMessage.getNotification() != null) {
            Log.d(TAG, "Message Notification Body: " +
remoteMessage.getNotification().getBody());
        }

        // do whatever you want with this, post your own notification, or update local state
    }
```

en **Firebase** , los usuarios pueden agruparse por su comportamiento como "AppVersion, usuario libre, usuario de compra o cualquier regla específica" y luego enviar una notificación a un grupo específico enviando la Función de **tema** en fireBase.
para registrar al usuario en el uso del tema

```
FirebaseMessaging.getInstance().subscribeToTopic("Free");
```

Luego, en la consola de FireBase, envíe una notificación por nombre de tema

Más información en el tema dedicado Firebase Cloud Messaging .

Agrega Firebase a tu proyecto de Android

Aquí hay pasos simplificados (basados en la documentación oficial) necesarios para crear un proyecto Firebase y conectarlo con una aplicación de Android.

Agrega Firebase a tu aplicación

1. Cree un proyecto de Firebase en la consola de Firebase y haga clic en **Crear nuevo proyecto** .
2. Haga clic en **Agregar Firebase a su aplicación de Android** y siga los pasos de configuración.
3. Cuando se le solicite, ingrese el **nombre del paquete de** su **aplicación** .
   Es importante ingresar el nombre del paquete completo que su aplicación está usando; esto solo se puede configurar cuando agrega una aplicación a su proyecto Firebase.
4. Al final, descargará un archivo `google-services.json` . Puedes descargar este archivo de nuevo en cualquier momento.
5. Si aún no lo ha hecho, copie el archivo `google-services.json` en la carpeta del módulo de su proyecto, normalmente `app/` .

El siguiente paso es agregar el SDK para integrar las bibliotecas Firebase en el proyecto.

Agrega el SDK

Para integrar las bibliotecas de Firebase en uno de sus propios proyectos, debe realizar algunas tareas básicas para preparar su proyecto de Android Studio. Es posible que ya hayas hecho esto como parte de agregar Firebase a tu aplicación.

1. Agregue reglas a su archivo `build.gradle` nivel `build.gradle` , para incluir el **complemento de servicios de google** :

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.1.0'
    }
}
```

Luego, en su módulo de archivo Gradle (generalmente `app/build.gradle`), agregue la línea de aplicación del complemento en la parte inferior del archivo para habilitar el complemento de Gradle:

```
apply plugin: 'com.android.application'

android {
  // ...
}

dependencies {
  // ...
  compile 'com.google.firebase:firebase-core:11.0.4'
}

// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

El último paso es agregar las dependencias para el SDK de Firebase usando una o más **bibliotecas disponibles** para las diferentes características de Firebase.

Línea de dependencia de Gradle	Servicio
com.google.firebase: firebase-core: 11.0.4	Analítica
com.google.firebase: firebase-database: 11.0.4	Base de datos en tiempo real
com.google.firebase: firebase-storage: 11.0.4	Almacenamiento
com.google.firebase: firebase-crash: 11.0.4	Reporte de Accidentes
com.google.firebase: firebase-auth: 11.0.4	Autenticación
com.google.firebase: firebase-messaging: 11.0.4	Mensajería en la nube / Notificaciones
com.google.firebase: firebase-config: 11.0.4	Configuración remota
com.google.firebase: firebase-invite: 11.0.4	Invitaciones / Enlaces Dinámicos
com.google.firebase: firebase-ads: 11.0.4	AdMob
com.google.android.gms: play-services-appindexing: 11.0.4	Indexación de aplicaciones

Firebase Realtime Database: cómo configurar / obtener datos

Nota: Configuremos alguna autenticación anónima para el ejemplo.

```
{
  "rules": {
```

```
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

Una vez hecho esto, crea un hijo editando la dirección de tu base de datos. Por ejemplo:

https://your-project.firebaseio.com/ to **https://your-project.firebaseio.com/chat**

Pondremos datos a esta ubicación desde nuestro dispositivo Android. No **tiene que** crear la estructura de la base de datos (pestañas, campos, etc.), se creará automáticamente cuando envíe el objeto Java a Firebase.

Cree un objeto Java que contenga todos los atributos que desea enviar a la base de datos:

```
public class ChatMessage {
    private String username;
    private String message;

    public ChatMessage(String username, String message) {
        this.username = username;
        this.message = message;
    }

    public ChatMessage() {} // you MUST have an empty constructor

    public String getUsername() {
        return username;
    }

    public String getMessage() {
        return message;
    }
}
```

Luego en tu actividad:

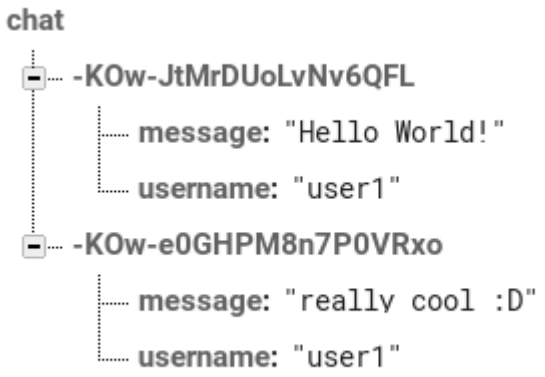
```
if (FirebaseAuth.getInstance().getCurrentUser() == null) {
        FirebaseAuth.getInstance().signInAnonymously().addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isComplete() && task.isSuccessful()){
                    FirebaseDatabase database = FirebaseDatabase.getInstance();
                    DatabaseReference reference = database.getReference("chat"); // reference
is 'chat' because we created the database at /chat
                }
            }
        });
}
```

Para enviar un valor:

```
ChatMessage msg = new ChatMessage("user1", "Hello World!");
reference.push().setValue(msg);
```

Para recibir los cambios que se producen en la base de datos:

```
reference.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        ChatMessage msg = dataSnapshot.getValue(ChatMessage.class);
        Log.d(TAG, msg.getUsername()+" "+msg.getMessage());
    }

    public void onChildChanged(DataSnapshot dataSnapshot, String s) {}
    public void onChildRemoved(DataSnapshot dataSnapshot) {}
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {}
    public void onCancelled(DatabaseError databaseError) {}
});
```

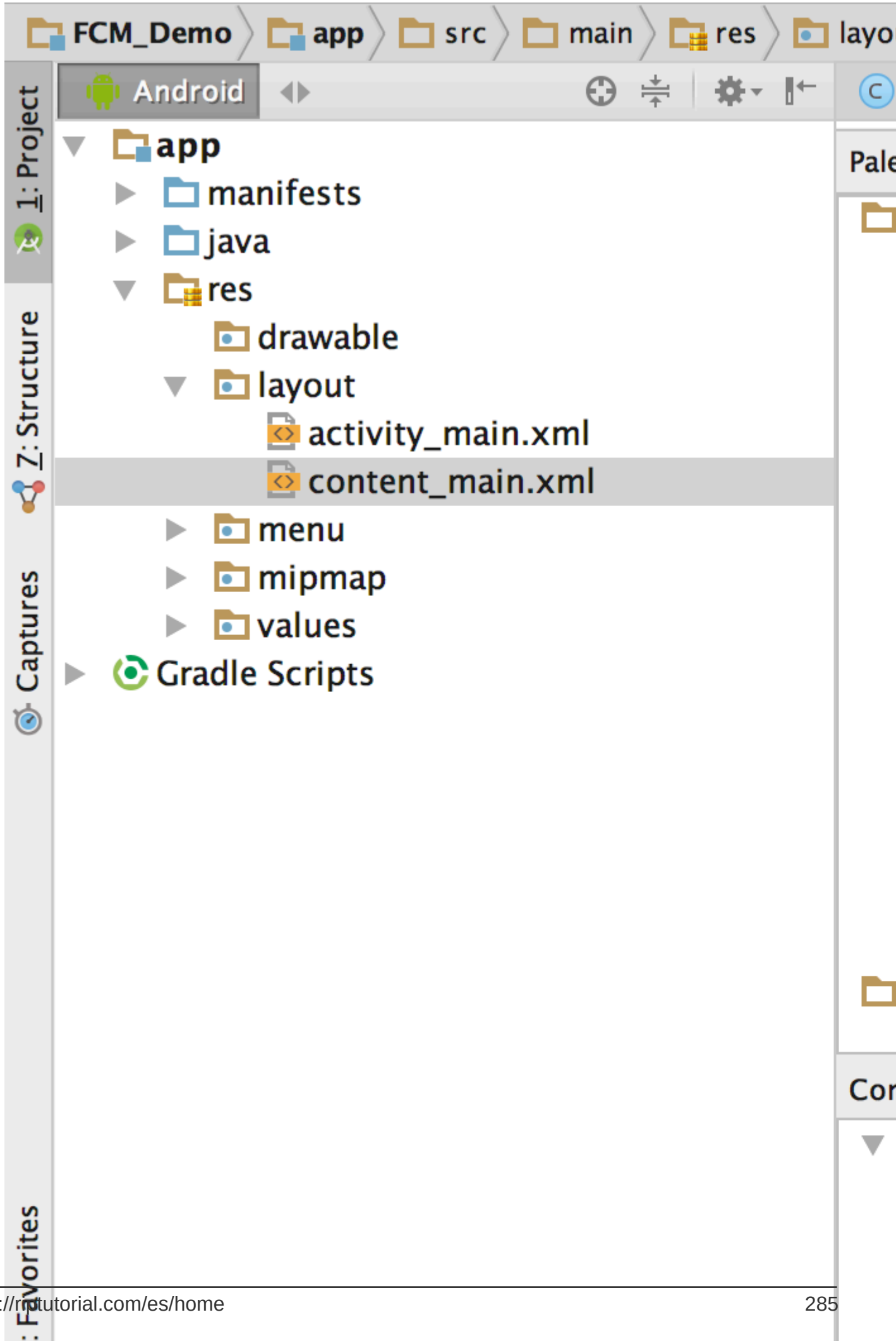


Demostración de notificaciones basadas en FCM

Este ejemplo muestra cómo usar la plataforma Firebase Cloud Messaging (FCM). FCM es un sucesor de Google Cloud Messaging (GCM). No requiere permisos C2D_MESSAGE de los usuarios de la aplicación.

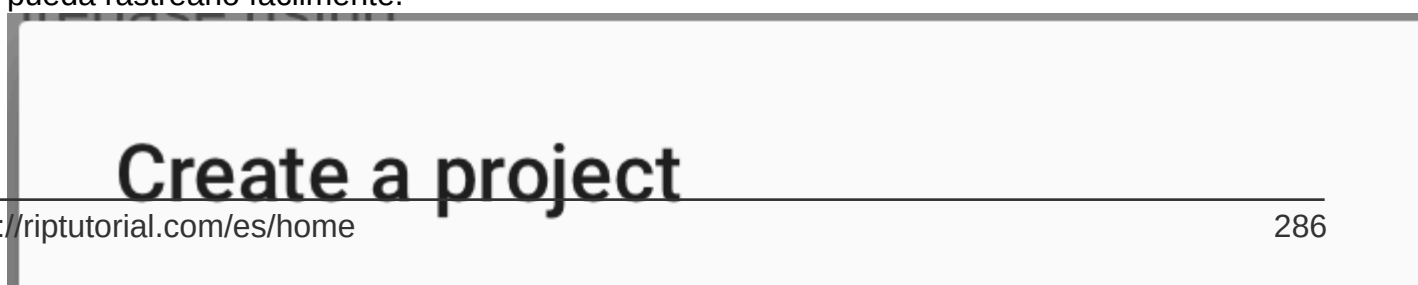
Los pasos para integrar FCM son los siguientes.

1. Cree un proyecto de ejemplo de hello world en Android Studio.


FCM_Demo
app
src
main
res
layo
Android
1: Project
7: Structure
Captures
: Favorites
app
manifests
java
res
drawable
layout
activity_main.xml
content_main.xml
menu
mipmap
values
Gradle Scripts
Pale
Cor

2. El siguiente paso es configurar el proyecto de base de fuego. Visite https://console.firebase.google.com y cree un proyecto con un nombre idéntico, para que pueda rastrearlo fácilmente.

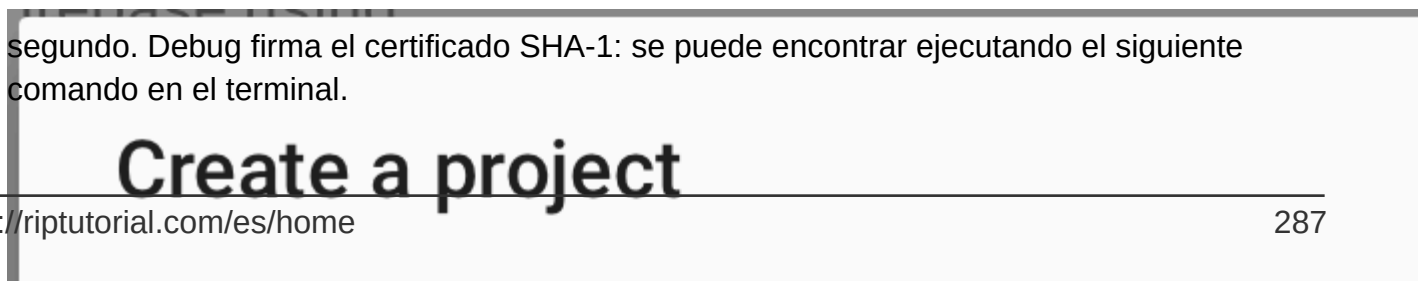
https://console.firebase.google.com y cree un proyecto con un nombre idéntico, para que pueda rastrearlo fácilmente.



3. Ahora es el momento de agregar base de fuego a su proyecto de muestra de Android que acaba de crear. Necesitará el nombre del paquete de su proyecto y el certificado de firma de depuración SHA-1 (opcional).

   a. Nombre del paquete: se puede encontrar en el archivo XML de manifiesto de Android.

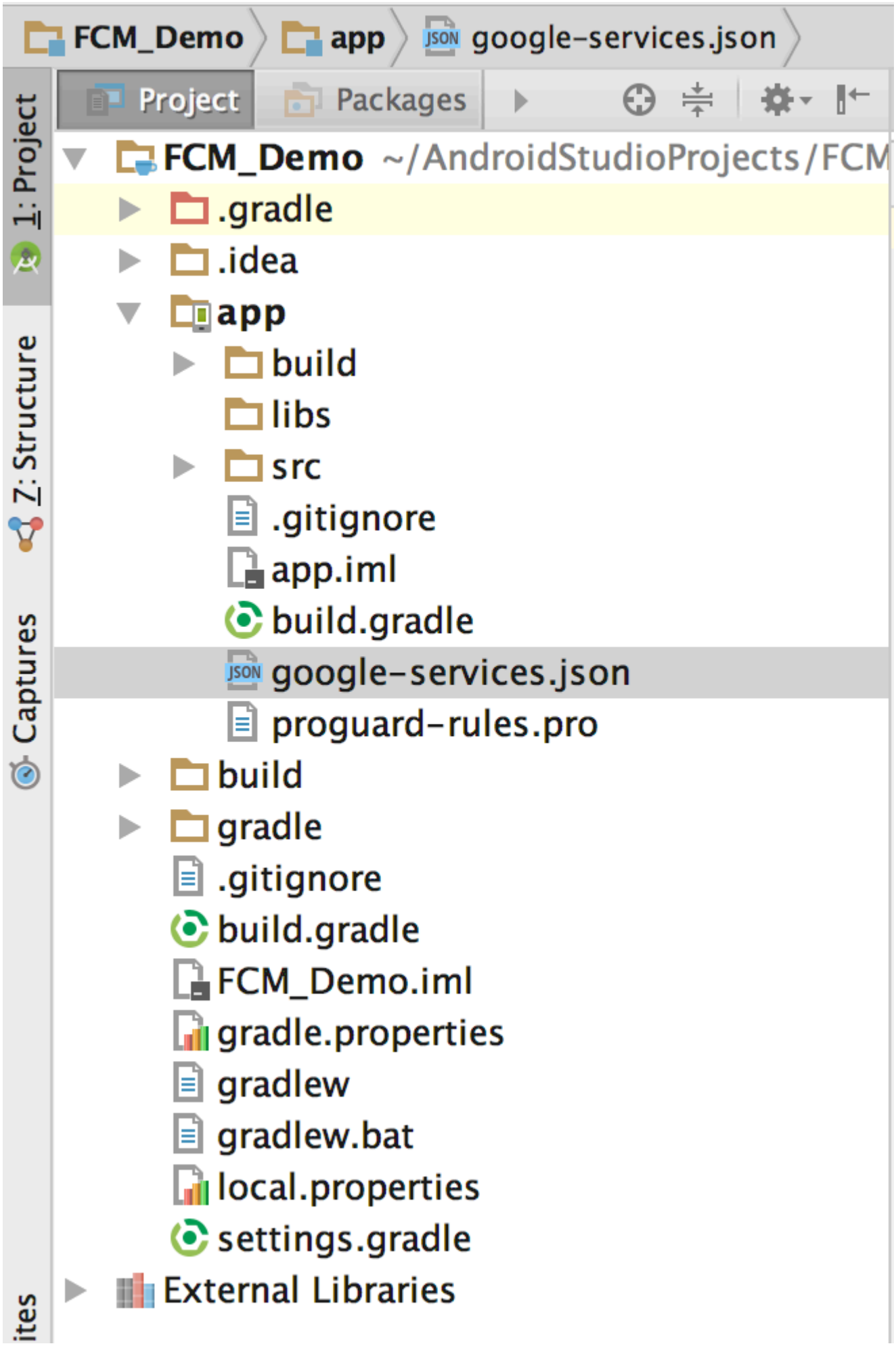
segundo. Debug firma el certificado SHA-1: se puede encontrar ejecutando el siguiente comando en el terminal.



```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
```

Ingrese esta información en la consola firebase y agregue la aplicación al proyecto firebase. Una vez que haga clic en el botón Agregar aplicación, su navegador descargará automáticamente un archivo JSON llamado "google-services.json".

4. Ahora copie el archivo google-services.json que acaba de descargar en el directorio raíz del módulo de su aplicación Android.


FCM_Demo
app
google-services.json
Project
Packages
1: Project
Z: Structure
Captures
FCM_Demo ~/AndroidStudioProjects/FCM
.gradle
.idea
app
build
libs
src
.gitignore
app.iml
build.gradle
google-services.json
proguard-rules.pro
build
gradle
.gitignore
build.gradle
FCM_Demo.iml
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
External Libraries

5. Siga las instrucciones proporcionadas en la consola firebase a medida que avanza. a. Agregue la siguiente línea de código a su nivel de proyecto build.gradle

```
dependencies{ classpath 'com.google.gms:google-services:3.1.0' .....
```

segundo. Agregue la siguiente línea de código al final de su nivel de aplicación build.gradle.

```
    //following are the dependencies to be added
    compile 'com.google.firebase:firebase-messaging:11.0.4'
    compile 'com.android.support:multidex:1.0.1'
}
// this line goes to the end of the file
apply plugin: 'com.google.gms.google-services'
```

do. Android studio te pedirá que sincronices el proyecto. Haga clic en sincronizar ahora.

6. La siguiente tarea es agregar dos servicios. a. Un servicio FirebaseMessagingService con filtro de intento como el siguiente

```
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
```

segundo. Una extensión de FirebaseInstanceIDService.

```
<intent-filter>
    <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
</intent-filter>
```

7. El código de FirebaseMessagingService debería tener este aspecto.

```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

import com.google.firebase.messaging.FirebaseMessagingService;

public class MyFirebaseMessagingService extends FirebaseMessagingService {
    public MyFirebaseMessagingService() {
    }
}
```

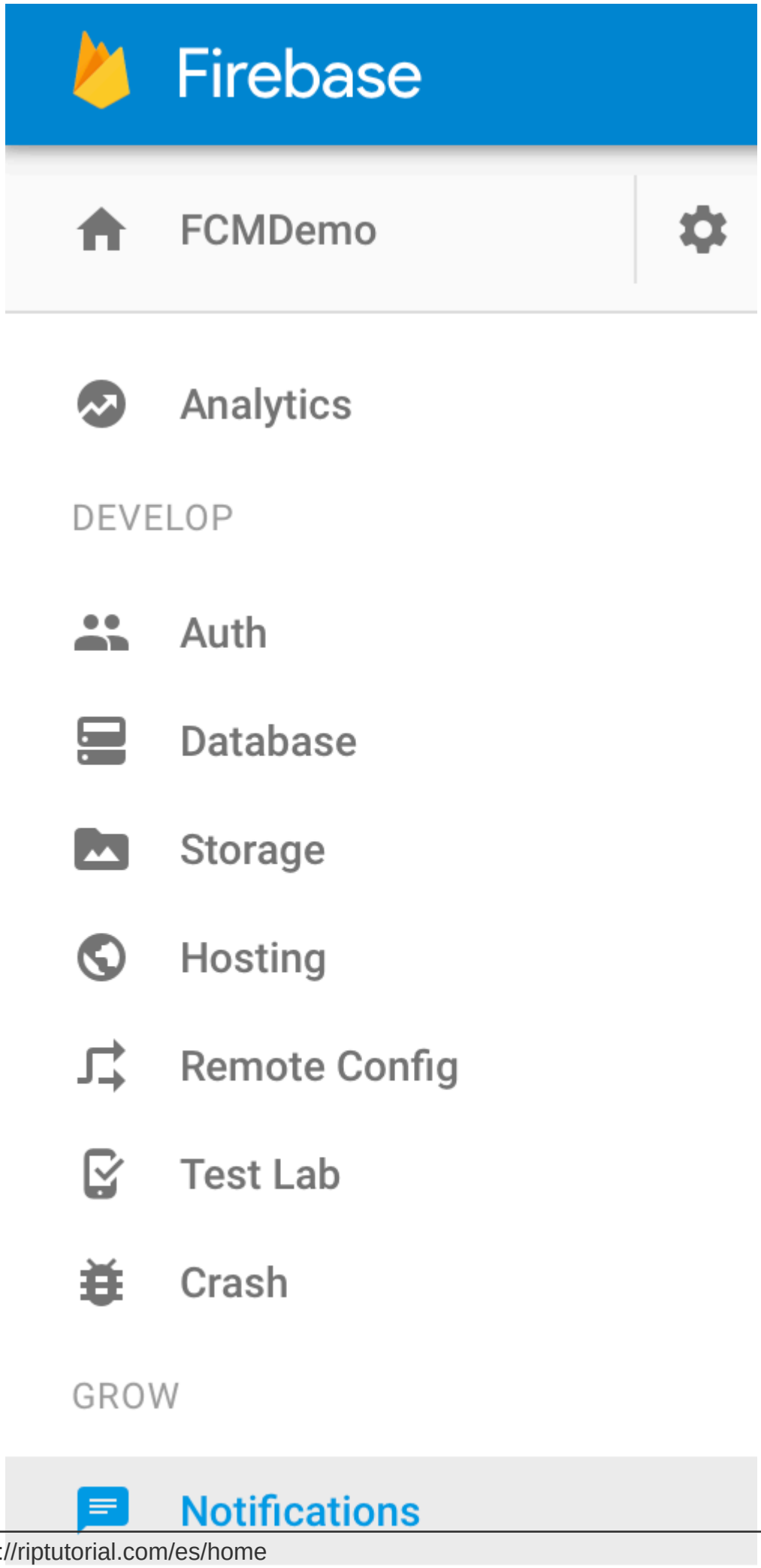
8. FirebaseInstanceIdService debería tener este aspecto.

```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

import com.google.firebase.iid.FirebaseInstanceIdService;

public class MyFirebaseInstanceIDService extends FirebaseInstanceIdService {
    public MyFirebaseInstanceIDService() {
    }
}
```

9. Ahora es el momento de capturar el token de registro del dispositivo. Agregue la siguiente línea de código al método onCreate de MainActivity.

```
String token = FirebaseInstanceId.getInstance().getToken();
Log.d("FCMAPP", "Token is "+token);
```

10. Una vez que tengamos el token de acceso, podemos usar la consola firebase para enviar la notificación. Ejecute la aplicación en su teléfono Android.


Firebase
FCMDemo
Analytics
DEVELOP
Auth
Database
Storage
Hosting
Remote Config
Test Lab
Crash
GROW
Notifications

Capítulo 37: Biblioteca de enlace de datos

Observaciones

Preparar

Antes de utilizar el enlace de datos, debe habilitar el complemento en su `build.gradle` .

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

Nota: el enlace de datos se agregó al complemento Gradle de Android en la versión 1.5.0

Encuadernación de nombres de clase

El complemento de enlace de datos genera un nombre de clase de enlace al convertir el nombre de archivo de su diseño al caso de Pascal y agregar "Enlace" al final. Por `item_detail_activity.xml` tanto, `item_detail_activity.xml` generará una clase llamada `ItemDetailActivityBinding` .

Recursos

- Documentacion oficial

Examples

Enlace de campo de texto básico

Configuración de Gradle (Módulo: aplicación)

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

Modelo de datos

```
public class Item {
    public String name;
    public String description;

    public Item(String name, String description) {
        this.name = name;
        this.description = description;
```

```
    }
}
```

Diseño XML

El primer paso es envolver su diseño en una etiqueta `<layout>` , agregar un elemento `<data>` y agregar un elemento `<variable>` para su modelo de datos.

A continuación, puede obligar a los atributos XML a campos en el modelo de datos utilizando `@{model.fieldname}` , donde `model` es el nombre de la variable y el `fieldname` es el campo al que desea acceder.

item_detail_activity.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
       <variable name="item" type="com.example.Item"/>
   </data>

   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

       <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@{item.name}"/>

       <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@{item.description}"/>

   </LinearLayout>
</layout>
```

Para cada archivo de diseño XML correctamente configurado con enlaces, el complemento Gradle de Android genera una clase correspondiente: enlaces. Debido a que tenemos un diseño denominado *item_detail_activity* , la clase de enlace generada correspondiente se llama `ItemDetailActivityBinding` .

Este enlace puede ser utilizado en una actividad como esta:

```
public class ItemDetailActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       ItemDetailActivityBinding binding = DataBindingUtil.setContentView(this,
R.layout.item_detail_activity);
       Item item = new Item("Example item", "This is an example item.");
       binding.setItem(item);
    }
}
```

Encuadernación con un método de acceso.

Si su modelo tiene métodos privados, la biblioteca de enlace de datos todavía le permite acceder a ellos en su vista sin usar el nombre completo del método.

Modelo de datos

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Diseño XML

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
       <variable name="item" type="com.example.Item"/>
   </data>

   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

       <!-- Since the "name" field is private on our data model,
            this binding will utilize the public getName() method instead. -->
       <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@{item.name}"/>

   </LinearLayout>
</layout>
```

Clases de referencia

Modelo de datos

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Diseño XML

Debe importar las clases referenciadas, tal como lo haría en Java.

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
       <import type="android.view.View"/>
       <variable name="item" type="com.example.Item"/>
   </data>

   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

       <!-- We reference the View class to set the visibility of this TextView -->
       <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@{item.name}"
           android:visibility="@{item.name == null ? View.VISIBLE : View.GONE"/>

   </LinearLayout>
</layout>
```

Nota: el paquete importa `java.lang.*` Automáticamente. (Lo mismo está hecho por `JVM` para `Java`)

Encuadernación de datos en Fragmento

Modelo de datos

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name){
        this.name = name;
    }

}
```

Diseño XML

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
       <variable name="item" type="com.example.Item"/>
   </data>

   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

       <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
```

```
           android:text="@{item.name}"/>

   </LinearLayout>
</layout>
```

Fragmento

```
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable
Bundle savedInstanceState) {
    FragmentTest binding = DataBindingUtil.inflate(inflater, R.layout.fragment_test,
container, false);
    Item item = new Item();
    item.setName("Thomas");
    binding.setItem(item);
    return binding.getRoot();
}
```

Enlace de datos bidireccional incorporado

El enlace de datos bidireccional admite los siguientes atributos:

Elemento	Propiedades
AbsListView	android:selectedItemPosition
CalendarView	android:date
CompoundButton	android:checked
DatePicker	• android:year • android:month • android:day
EditText	android:text
NumberPicker	android:value
RadioGroup	android:checkedButton
RatingBar	android:rating
SeekBar	android:progress
TabHost	android:currentTab
TextView	android:text
TimePicker	• android:hour • android:minute
ToggleButton	android:checked

Elemento	Propiedades
Switch	android:checked

Uso

```
<layout ...>
    <data>
        <variable type="com.example.myapp.User" name="user"/>
    </data>
    <RelativeLayout ...>
        <EditText android:text="@={user.firstName}" .../>
    </RelativeLayout>
</layout>
```

Observe que la expresión de enlace `@={}` **tiene un = adicional** , que es necesario para el **enlace de dos vías** . No es posible utilizar métodos en expresiones de enlace de dos vías.

Enlace de datos en el adaptador RecyclerView

También es posible utilizar el enlace de datos dentro de su adaptador `RecyclerView` .

Modelo de datos

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Diseño XML

```
<TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@{item.name}"/>
```

Clase de adaptador

```
public class ListItemAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

    private Activity host;
    private List<Item> items;

    public ListItemAdapter(Activity activity, List<Item> items) {
        this.host = activity;
        this.items = items;
    }
```

```
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        // inflate layout and retrieve binding
        ListItemBinding binding = DataBindingUtil.inflate(host.getLayoutInflater(),
                R.layout.list_item, parent, false);

        return new ItemViewHolder(binding);
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
        Item item = items.get(position);

        ItemViewHolder itemViewHolder = (ItemViewHolder)holder;
        itemViewHolder.bindItem(item);
    }

    @Override
    public int getItemCount() {
        return items.size();
    }

    private static class ItemViewHolder extends RecyclerView.ViewHolder {
        ListItemBinding binding;

        ItemViewHolder(ListItemBinding binding) {
            super(binding.getRoot());
            this.binding = binding;
        }

        void bindItem(Item item) {
            binding.setItem(item);
            binding.executePendingBindings();
        }
    }
}
```

Click listener con Binding

Crear interfaz para clickHandler

```
public interface ClickHandler {
    public void onButtonClick(View v);
}
```

Diseño XML

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>
        <variable
            name="handler"
            type="com.example.ClickHandler"/>
    </data>

    <RelativeLayout
        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="click me"
            android:onClick="@{handler.onButtonClick}"/>
    </RelativeLayout>
</layout>
```

Manejar evento en tu actividad.

```
public class MainActivity extends Activity implements ClickHandler {

    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = DataBindingUtil.setContentView(this,R.layout.activity_main);
        binding.setHandler(this);
    }

    @Override
    public void onButtonClick(View v) {
        Toast.makeText(context,"Button clicked",Toast.LENGTH_LONG).show();
    }
}
```

Evento personalizado usando la expresión lambda

Definir interfaz

```
public interface ClickHandler {
    public void onButtonClick(User user);
}
```

Crear clase de modelo

```
public class User {
    private String name;

    public User(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Diseño XML

```
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>
        <variable
            name="handler"
            type="com.example.ClickHandler"/>

        <variable
            name="user"
            type="com.example.User"/>
    </data>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{user.name}"
            android:onClick="@{() -> handler.onButtonClick(user)}"/>
    </RelativeLayout>
</layout>
```

Código de actividad:

```
public class MainActivity extends Activity implements ClickHandler {

    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = DataBindingUtil.setContentView(this,R.layout.activity_main);
        binding.setUser(new User("DataBinding User"));
        binding.setHandler(this);
    }

    @Override
    public void onButtonClick(User user) {
        Toast.makeText(MainActivity.this,"Welcome " +
user.getName(),Toast.LENGTH_LONG).show();
    }
}
```

Para algunos oyentes de vista que no están disponibles en el código xml, pero se pueden configurar en código java, se pueden enlazar con enlaces personalizados.

Clase personalizada

```
public class BindingUtil {
    @BindingAdapter({"bind:autoAdapter"})
    public static void setAdapter(AutoCompleteTextView view, ArrayAdapter<String>
pArrayAdapter) {
        view.setAdapter(pArrayAdapter);
    }
    @BindingAdapter({"bind:onKeyListener"})
    public static void setOnKeyListener(AutoCompleteTextView view , View.OnKeyListener
pOnKeyListener)
```

```
    {
        view.setOnKeyListener(pOnKeyListener);
    }
}
```

Clase de manejador

```
public class Handler extends BaseObservable {
    private ArrayAdapter<String> roleAdapter;

    public ArrayAdapter<String> getRoleAdapter() {
        return roleAdapter;
    }
    public void setRoleAdapter(ArrayAdapter<String> pRoleAdapter) {
        roleAdapter = pRoleAdapter;
    }
}
```

XML

```
<layout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:bind="http://schemas.android.com/tools" >

    <data>
        <variable
            name="handler"
            type="com.example.Handler" />
    </data>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <AutoCompleteTextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:singleLine="true"
            bind:autoAdapter="@{handler.roleAdapter}" />

    </LinearLayout>
</layout>
```

Valor por defecto en enlace de datos

> El panel de vista previa muestra los valores predeterminados para las expresiones de enlace de datos, si se proporcionan.

Por ejemplo :

```
android:layout_height="@{@dimen/main_layout_height, default=wrap_content}"
```

Tomará `wrap_content` durante el diseño y actuará como `wrap_content` en el panel de vista previa.

Otro ejemplo es

```
android:text="@{user.name, default=`Preview Text`}"
```

Mostrará `Preview Text` en el panel de vista previa, pero cuando lo ejecute en el dispositivo / emulador, se mostrará el texto real vinculado a él.

Enlace de datos con variables personalizadas (int, booleano)

A veces necesitamos realizar operaciones básicas como la vista de ocultar / mostrar basada en un solo valor, para esa única variable no podemos crear un modelo o no es una buena práctica crear un modelo para eso. DataBinding admite tipos de datos básicos para realizar esas operaciones.

```
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>

        <import type="android.view.View" />

        <variable
            name="selected"
            type="Boolean" />

    </data>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello World"
            android:visibility="@{selected ? View.VISIBLE : View.GONE}" />

    </RelativeLayout>
</layout>
```

y establecer su valor de clase java.

```
binding.setSelected(true);
```

Encuadernación de datos en diálogo

```
public void doSomething() {
    DialogTestBinding binding = DataBindingUtil
            .inflate(LayoutInflater.from(context), R.layout.dialog_test, null, false);

    Dialog dialog = new Dialog(context);
    dialog.setContentView(binding.getRoot());
    dialog.show();
}
```

Pase el widget como referencia en BindingAdapter

Diseño XML

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>

   </data>

   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

       <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

        <ImageView
        android:id="@+id/img"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        app:imageUrl="@{url}"
        app:progressbar="@{progressBar}"/>

   </LinearLayout>
</layout>
```

Método BindingAdapter

```
@BindingAdapter({"imageUrl","progressbar"})
public static void loadImage(ImageView view, String imageUrl, ProgressBar progressBar){
    Glide.with(view.getContext()).load(imageUrl)
                        .listener(new RequestListener<String, GlideDrawable>() {
                    @Override
                    public boolean onException(Exception e, String model,
Target<GlideDrawable> target, boolean isFirstResource) {
                        return false;
                    }

                    @Override
                    public boolean onResourceReady(GlideDrawable resource, String model,
Target<GlideDrawable> target, boolean isFromMemoryCache, boolean isFirstResource) {
                        progressBar.setVisibility(View.GONE);
                        return false;
                    }
                }).into(view);
}
```

Lea Biblioteca de enlace de datos en línea: https://riptutorial.com/es/android/topic/111/biblioteca-de-enlace-de-datos

Capítulo 38: Bluetooth Low Energy

Introducción

Esta documentación pretende ser una mejora con respecto a la documentación original y se centrará en la última API de Bluetooth LE introducida en Android 5.0 (API 21). Se cubrirán los roles tanto Central como Periférico, así como la forma de iniciar las operaciones de escaneo y publicidad.

Examples

Buscando dispositivos BLE

Se requieren los siguientes permisos para usar las API de Bluetooth:

```
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
```

Si está apuntando a dispositivos con Android 6.0 (**nivel de API 23**) o superior y desea realizar operaciones de escaneo / publicidad, necesitará un permiso de ubicación:

```
android.permission.ACCESS_FINE_LOCATION
```

o

```
android.permission.ACCESS_COARSE_LOCATION
```

Nota.- Los dispositivos con Android 6.0 (nivel de API 23) o superior también deben tener habilitados los Servicios de ubicación.

Se requiere un objeto BluetoothAdapter para iniciar las operaciones de escaneo / publicidad:

```
BluetoothManager bluetoothManager = (BluetoothManager)
context.getSystemService(Context.BLUETOOTH_SERVICE);
bluetoothAdapter = bluetoothManager.getAdapter();
```

El `startScan (ScanCallback callback)` de la clase BluetoothLeScanner es la forma más básica de iniciar una operación de escaneo. Se `ScanCallback` objeto `ScanCallback` para recibir resultados:

```
bluetoothAdapter.getBluetoothLeScanner().startScan(new ScanCallback() {
     @Override
     public void onScanResult(int callbackType, ScanResult result) {
     super.onScanResult(callbackType, result);
     Log.i(TAG, "Remote device name: " + result.getDevice().getName());
       }
    });
```

Conectando a un servidor GATT

Una vez que haya descubierto un objeto BluetoothDevice deseado, puede conectarse utilizando su método `connectGatt()` , que toma como parámetros un objeto Context, un booleano que indica si debe conectarse automáticamente al dispositivo BLE y una referencia BluetoothGattCallback donde los eventos de conexión y las operaciones del cliente Los resultados serán entregados:

```
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        device.connectGatt(context, false, bluetoothGattCallback,
BluetoothDevice.TRANSPORT_AUTO);
    } else {
        device.connectGatt(context, false, bluetoothGattCallback);
    }
```

Reemplace `onConnectionStateChange` en BluetoothGattCallback para recibir conexión y eventos de desconexión:

```
    BluetoothGattCallback bluetoothGattCallback =
        new BluetoothGattCallback() {
    @Override
    public void onConnectionStateChange(BluetoothGatt gatt, int status,
            int newState) {
        if (newState == BluetoothProfile.STATE_CONNECTED) {
            Log.i(TAG, "Connected to GATT server.");

        } else if (newState == BluetoothProfile.STATE_DISCONNECTED) {

            Log.i(TAG, "Disconnected from GATT server.");
        }
    }
   };
```

Escritura y lectura de características.

Una vez que estés conectado a un servidor Gatt, estarás interactuando escribiendo y leyendo las características del servidor. Para hacer esto, primero debe descubrir qué servicios están disponibles en este servidor y qué características están disponibles en cada servicio:

```
 @Override
 public void onConnectionStateChange(BluetoothGatt gatt, int status,
        int newState) {
    if (newState == BluetoothProfile.STATE_CONNECTED) {
        Log.i(TAG, "Connected to GATT server.");
        gatt.discoverServices();

    }
. . .

 @Override
    public void onServicesDiscovered(BluetoothGatt gatt, int status) {
        if (status == BluetoothGatt.GATT_SUCCESS) {
             List<BluetoothGattService> services = gatt.getServices();
                for (BluetoothGattService service : services) {
                    List<BluetoothGattCharacteristic> characteristics =
```

```
service.getCharacteristics();
                    for (BluetoothGattCharacteristic characteristic : characteristics) {
                        ///Once you have a characteristic object, you can perform read/write
                        //operations with it
                    }
                }
            }
        }
```

Una operación básica de escritura es la siguiente:

```
characteristic.setValue(newValue);
characteristic.setWriteType(BluetoothGattCharacteristic.WRITE_TYPE_DEFAULT);
gatt.writeCharacteristic(characteristic);
```

Cuando el proceso de escritura haya finalizado, se `onCharacteristicWrite` método `onCharacteristicWrite` de su `BluetoothGattCallback` :

```
    @Override
    public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
        super.onCharacteristicWrite(gatt, characteristic, status);
        Log.d(TAG, "Characteristic " + characteristic.getUuid() + " written);
    }
```

Una operación básica de escritura es la siguiente:

```
gatt.readCharacteristic(characteristic);
```

Cuando el proceso de escritura haya finalizado, se `onCharacteristicRead` método `onCharacteristicRead` de su `BluetoothGattCallback` :

```
@Override
public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
  super.onCharacteristicRead(gatt, characteristic, status);
  byte[] value = characteristic.getValue();
  }
```

Suscripción a notificaciones desde el servidor Gatt

Puede solicitar que se le notifique al Servidor Gatt cuando se haya cambiado el valor de una característica:

```
gatt.setCharacteristicNotification(characteristic, true);
BluetoothGattDescriptor descriptor = characteristic.getDescriptor(
    UUID.fromString("00002902-0000-1000-8000-00805f9b34fb");
descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
mBluetoothGatt.writeDescriptor(descriptor);
```

Todas las notificaciones del servidor se recibirán en el método `onCharacteristicChanged` de su BluetoothGattCallback:

```
    @Override
    public void onCharacteristicChanged(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic) {
        super.onCharacteristicChanged(gatt, characteristic);
        byte[] newValue = characteristic.getValue();
}
```

Publicidad de un dispositivo BLE

Puede usar Bluetooth LE Advertising para transmitir paquetes de datos a todos los dispositivos cercanos sin tener que establecer una conexión primero. Tenga en cuenta que hay un límite estricto de 31 bytes de datos de publicidad. La publicidad de su dispositivo también es el primer paso para permitir que otros usuarios se conecten con usted.

Dado que no todos los dispositivos son compatibles con Bluetooth LE Advertising, el primer paso es verificar que su dispositivo tenga todos los requisitos necesarios para admitirlo. Después, puede inicializar un objeto `BluetoothLeAdvertiser` y con él, puede iniciar operaciones de publicidad:

```
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP &&
bluetoothAdapter.isMultipleAdvertisementSupported())
        {
            BluetoothLeAdvertiser advertiser = bluetoothAdapter.getBluetoothLeAdvertiser();

            AdvertiseData.Builder dataBuilder = new AdvertiseData.Builder();
           //Define a service UUID according to your needs
           dataBuilder.addServiceUuid(SERVICE_UUID);
            dataBuilder.setIncludeDeviceName(true);


             AdvertiseSettings.Builder settingsBuilder = new AdvertiseSettings.Builder();
             settingsBuilder.setAdvertiseMode(AdvertiseSettings.ADVERTISE_MODE_LOW_POWER);
             settingsBuilder.setTimeout(0);

             //Use the connectable flag if you intend on opening a Gatt Server
             //to allow remote connections to your device.
             settingsBuilder.setConnectable(true);

            AdvertiseCallback advertiseCallback=new AdvertiseCallback() {
            @Override
            public void onStartSuccess(AdvertiseSettings settingsInEffect) {
                super.onStartSuccess(settingsInEffect);
                Log.i(TAG, "onStartSuccess: ");
            }

            @Override
            public void onStartFailure(int errorCode) {
                super.onStartFailure(errorCode);
                Log.e(TAG, "onStartFailure: "+errorCode );
            }
          };
advertising.startAdvertising(settingsBuilder.build(),dataBuilder.build(),advertiseCallback);
        }
```

Usando un servidor Gatt

Para que su dispositivo actúe como un periférico, primero debe abrir un `BluetoothGattServer`

servidor de `BluetoothGattServer` y rellenarlo con al menos un servicio de `BluetoothGattService` y una característica de caracteres de `BluetoothGattCharacteristic` :

```
BluetoothGattServer server=bluetoothManager.openGattServer(context,
bluetoothGattServerCallback);

BluetoothGattService service = new BluetoothGattService(SERVICE_UUID,
BluetoothGattService.SERVICE_TYPE_PRIMARY);
```

Este es un ejemplo de BluetoothGattCharacteristic con permisos completos de escritura, lectura y notificación. De acuerdo con sus necesidades, es posible que desee ajustar los permisos que otorga esta característica:

```
BluetoothGattCharacteristic characteristic = new
BluetoothGattCharacteristic(CHARACTERISTIC_UUID,
                    BluetoothGattCharacteristic.PROPERTY_READ |
BluetoothGattCharacteristic.PROPERTY_WRITE |
                            BluetoothGattCharacteristic.PROPERTY_NOTIFY,
                    BluetoothGattCharacteristic.PERMISSION_READ |
BluetoothGattCharacteristic.PERMISSION_WRITE);

characteristic.addDescriptor(new BluetoothGattDescriptor(UUID.fromString("00002902-0000-1000-
8000-00805f9b34fb"), BluetoothGattCharacteristic.PERMISSION_WRITE));

service.addCharacteristic(characteristic);

server.addService(service);
```

El `BluetoothGattServerCallback` es responsable de recibir todos los eventos relacionados con su `BluetoothGattServer` :

```
BluetoothGattServerCallback bluetoothGattServerCallback= new BluetoothGattServerCallback() {
                @Override
                public void onConnectionStateChange(BluetoothDevice device, int status, int
newState) {
                    super.onConnectionStateChange(device, status, newState);
                }

                @Override
                public void onCharacteristicReadRequest(BluetoothDevice device, int requestId,
int offset, BluetoothGattCharacteristic characteristic) {
                    super.onCharacteristicReadRequest(device, requestId, offset,
characteristic);
                }

                @Override
                public void onCharacteristicWriteRequest(BluetoothDevice device, int
requestId, BluetoothGattCharacteristic characteristic, boolean preparedWrite, boolean
responseNeeded, int offset, byte[] value) {
                    super.onCharacteristicWriteRequest(device, requestId, characteristic,
preparedWrite, responseNeeded, offset, value);
                }

                @Override
                public void onDescriptorReadRequest(BluetoothDevice device, int requestId, int
offset, BluetoothGattDescriptor descriptor) {
                    super.onDescriptorReadRequest(device, requestId, offset, descriptor);
```

```
                }

                @Override
                public void onDescriptorWriteRequest(BluetoothDevice device, int requestId,
BluetoothGattDescriptor descriptor, boolean preparedWrite, boolean responseNeeded, int offset,
byte[] value) {
                     super.onDescriptorWriteRequest(device, requestId, descriptor,
preparedWrite, responseNeeded, offset, value);
                }
```

Siempre que reciba una solicitud de escritura / lectura de una característica o descriptor, debe enviar una respuesta para que la solicitud se complete con éxito:

```
@Override
 public void onCharacteristicReadRequest(BluetoothDevice device, int requestId, int offset,
BluetoothGattCharacteristic characteristic) {
    super.onCharacteristicReadRequest(device, requestId, offset, characteristic);
    server.sendResponse(device, requestId, BluetoothGatt.GATT_SUCCESS, offset, YOUR_RESPONSE);
}
```

Lea Bluetooth Low Energy en línea: https://riptutorial.com/es/android/topic/10020/bluetooth-low-energy

Capítulo 39: Bluetooth y Bluetooth LE API

Observaciones

Bluetooth Classic está disponible desde Android 2.0 (nivel API 5) y superior. Bluetooth LE está disponible desde Android 4.3 (nivel de API 18) y superior.

Examples

Permisos

Agregue este permiso al archivo de manifiesto para usar las funciones de Bluetooth en su aplicación:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

Si necesita iniciar el descubrimiento del dispositivo o manipular la configuración de Bluetooth, también debe agregar este permiso:

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

El objetivo de la API de Android de nivel 23 y superior, requerirá acceso a la ubicación:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<!-- OR -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

* También consulte el tema Permisos para obtener más detalles sobre cómo usar los permisos de manera adecuada.

Compruebe si Bluetooth está habilitado

```
private static final int REQUEST_ENABLE_BT = 1; // Unique request code
BluetoothAdapter mBluetoothAdapter;

// ...

if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}

// ...

@Override
protected void onActivityResult(final int requestCode, final int resultCode, final Intent
data) {
    super.onActivityResult(requestCode, resultCode, data);
```

```
    if (requestCode == REQUEST_ENABLE_BT) {
        if (resultCode == RESULT_OK) {
            // Bluetooth was enabled
        } else if (resultCode == RESULT_CANCELED) {
            // Bluetooth was not enabled
        }
    }
}
```

Hacer que el dispositivo sea detectable

```
private static final int REQUEST_DISCOVERABLE_BT = 2; // Unique request code
private static final int DISCOVERABLE_DURATION = 120; // Discoverable duration time in seconds
                                                      // 0 means always discoverable
                                                      // maximum value is 3600

// ...

Intent discoverableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION,
DISCOVERABLE_DURATION);
startActivityForResult(discoverableIntent, REQUEST_DISCOVERABLE_BT);

// ...

@Override
protected void onActivityResult(final int requestCode, final int resultCode, final Intent
data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST_DISCOVERABLE_BT) {
        if (resultCode == RESULT_OK) {
            // Device is discoverable
        } else if (resultCode == RESULT_CANCELED) {
            // Device is not discoverable
        }
    }
}
```

Encuentra dispositivos bluetooth cercanos

Declara un adaptador `BluetoothAdapter` primero.

```
BluetoothAdapter mBluetoothAdapter;
```

Ahora crea un `BroadcastReceiver` para `ACTION_FOUND`

```
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();

    //Device found
    if (BluetoothDevice.ACTION_FOUND.equals(action))
    {
        // Get the BluetoothDevice object from the Intent
```

```
        BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
        // Add the name and address to an array adapter to show in a list
        mArrayAdapter.add(device.getName() + "\n" + device.getAddress());
    }
  }
};
```

Registrar el `BroadcastReceiver`

```
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);
```

Luego comienza a descubrir los dispositivos Bluetooth cercanos llamando a `startDiscovery`

```
mBluetoothAdapter.startDiscovery();
```

No olvide `onDestroy` el registro de `BroadcastReceiver` dentro `onDestroy`

```
unregisterReceiver(mReceiver);
```

Conectar a dispositivo Bluetooth

Después de obtener el dispositivo Bluetooth, puedes comunicarte con él. Este tipo de comunicación se realiza mediante el uso de entradas / salidas de socket:

Esos son los pasos básicos para el establecimiento de la comunicación Bluetooth:

1) Inicializar zócalo:

```
 private BluetoothSocket _socket;
 //...
 public InitializeSocket(BluetoothDevice device){
    try {
        _socket = device.createRfcommSocketToServiceRecord(<Your app UDID>);
    } catch (IOException e) {
        //Error
    }
  }
```

2) Conectar al zócalo:

```
try {
    _socket.connect();
} catch (IOException connEx) {
    try {
        _socket.close();
    } catch (IOException closeException) {
        //Error
    }
}

if (_socket != null && _socket.isConnected()) {
    //Socket is connected, now we can obtain our IO streams
```

```
}
```

3) Obtención de entrada de socket \ flujos de salida

```
private InputStream _inStream;
private OutputStream _outStream;
//....
try {
    _inStream = _socket.getInputStream();
    _outStream =  _socket.getOutputStream();
} catch (IOException e) {
   //Error
}
```

Flujo de entrada : se utiliza como canal de datos entrantes (recibe datos del dispositivo conectado)

Flujo de salida : se utiliza como canal de datos salientes (enviar datos al dispositivo conectado)

Después de finalizar el 3er paso, podemos recibir y enviar datos entre ambos dispositivos utilizando flujos previamente iniciados:

1) Recepción de datos (lectura desde el flujo de entrada de socket)

```
byte[] buffer = new byte[1024];  // buffer (our data)
int bytesCount; // amount of read bytes

while (true) {
    try {
        //reading data from input stream
        bytesCount = _inStream.read(buffer);
        if(buffer != null && bytesCount > 0)
        {
            //Parse received bytes
        }
    } catch (IOException e) {
        //Error
    }
}
```

2) Envío de datos (Escritura a flujo de salida)

```
public void write(byte[] bytes) {
    try {
        _outStream.write(bytes);
    } catch (IOException e) {
        //Error
    }
}
```

- Por supuesto, la funcionalidad de conexión, lectura y escritura se debe hacer en un hilo dedicado.
- Sockets y objetos Stream deben ser

Encuentra dispositivos Bluetooth de baja energía cercanos

La API de BluetoothLE se introdujo en la API 18. Sin embargo, la forma de escanear dispositivos ha cambiado en la API 21. La búsqueda de dispositivos debe comenzar con la definición del UUID de servicio que se va a escanear (ya sea de forma independiente o adoptada por UUID de 16 bits o de propietario) . Este ejemplo ilustra cómo hacer una forma independiente de búsqueda de dispositivos BLE para la API:

1. Crear modelo de dispositivo bluetooth:

```
public class BTDevice {
    String address;
    String name;

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

2. Definir la interfaz de escaneo de Bluetooth:

```
public interface ScanningAdapter {

    void startScanning(String name, String[] uuids);
    void stopScanning();
    List<BTDevice> getFoundDeviceList();
}
```

3. Crear clase de escaneo de fábrica:

```
public class BluetoothScanningFactory implements ScanningAdapter {

    private ScanningAdapter mScanningAdapter;

    public BluetoothScanningFactory() {
        if (isNewerAPI()) {
            mScanningAdapter = new LollipopBluetoothLEScanAdapter();
        } else {
            mScanningAdapter = new JellyBeanBluetoothLEScanAdapter();
        }
    }

    private boolean isNewerAPI() {
```

```
        return Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP;
    }

    @Override
    public void startScanning(String[] uuids) {
        mScanningAdapter.startScanning(uuids);
    }

    @Override
    public void stopScanning() {
        mScanningAdapter.stopScanning();
    }

    @Override
    public List<BTDevice> getFoundDeviceList() {
        return mScanningAdapter.getFoundDeviceList();
    }
}
```

4. Crear implementación de fábrica para cada API:

API 18:

```
import android.annotation.TargetApi;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.os.Build;
import android.os.Parcelable;
import android.util.Log;

import bluetooth.model.BTDevice;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

@TargetApi(Build.VERSION_CODES.JELLY_BEAN_MR2)
public class JellyBeanBluetoothLEScanAdapter implements ScanningAdapter{
    BluetoothAdapter bluetoothAdapter;
    ScanCallback mCallback;

    List<BTDevice> mBluetoothDeviceList;

    public JellyBeanBluetoothLEScanAdapter() {
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        mCallback = new ScanCallback();
        mBluetoothDeviceList = new ArrayList<>();
    }

    @Override
    public void startScanning(String[] uuids) {
        if (uuids == null || uuids.length == 0) {
            return;
        }
        UUID[] uuidList = createUUIDList(uuids);
        bluetoothAdapter.startLeScan(uuidList, mCallback);
    }

    private UUID[] createUUIDList(String[] uuids) {
        UUID[] uuidList = new UUID[uuids.length];
```

```
        for (int i = 0 ; i < uuids.length ; ++i) {
            String uuid = uuids[i];
            if (uuid == null) {
                continue;
            }
            uuidList[i] = UUID.fromString(uuid);
        }
        return uuidList;
    }

    @Override
    public void stopScanning() {
        bluetoothAdapter.stopLeScan(mCallback);
    }

    @Override
    public List<BTDevice> getFoundDeviceList() {
        return mBluetoothDeviceList;
    }

    private class ScanCallback implements BluetoothAdapter.LeScanCallback {

        @Override
        public void onLeScan(BluetoothDevice device, int rssi, byte[] scanRecord) {
            if (isAlreadyAdded(device)) {
                return;
            }
            BTDevice btDevice = new BTDevice();
            btDevice.setName(new String(device.getName()));
            btDevice.setAddress(device.getAddress());
            mBluetoothDeviceList.add(btDevice);
            Log.d("Bluetooth discovery", device.getName() + " " + device.getAddress());
            Parcelable[] uuids = device.getUuids();
            String uuid = "";
            if (uuids != null) {
                for (Parcelable ep : uuids) {
                    uuid += ep + " ";
                }
                Log.d("Bluetooth discovery", device.getName() + " " + device.getAddress() + "
" + uuid);
            }
        }

        private boolean isAlreadyAdded(BluetoothDevice bluetoothDevice) {
            for (BTDevice device : mBluetoothDeviceList) {
                String alreadyAddedDeviceMACAddress = device.getAddress();
                String newDeviceMACAddress = bluetoothDevice.getAddress();
                if (alreadyAddedDeviceMACAddress.equals(newDeviceMACAddress)) {
                    return true;
                }
            }
            return false;
        }
    }
}
```

API 21:

```
import android.annotation.TargetApi;
import android.bluetooth.BluetoothAdapter;
```

```
import android.bluetooth.le.BluetoothLeScanner;
import android.bluetooth.le.ScanFilter;
import android.bluetooth.le.ScanResult;
import android.bluetooth.le.ScanSettings;
import android.os.Build;
import android.os.ParcelUuid;

import bluetooth.model.BTDevice;

import java.util.ArrayList;
import java.util.List;

@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class LollipopBluetoothLEScanAdapter implements ScanningAdapter {
    BluetoothLeScanner bluetoothLeScanner;
    ScanCallback mCallback;

    List<BTDevice> mBluetoothDeviceList;

    public LollipopBluetoothLEScanAdapter() {
        bluetoothLeScanner = BluetoothAdapter.getDefaultAdapter().getBluetoothLeScanner();
        mCallback = new ScanCallback();
        mBluetoothDeviceList = new ArrayList<>();
    }

    @Override
    public void startScanning(String[] uuids) {
        if (uuids == null || uuids.length == 0) {
            return;
        }
        List<ScanFilter> filterList = createScanFilterList(uuids);
        ScanSettings scanSettings = createScanSettings();
        bluetoothLeScanner.startScan(filterList, scanSettings, mCallback);
    }

    private List<ScanFilter> createScanFilterList(String[] uuids) {
        List<ScanFilter> filterList = new ArrayList<>();
        for (String uuid : uuids) {
            ScanFilter filter = new ScanFilter.Builder()
                    .setServiceUuid(ParcelUuid.fromString(uuid))
                    .build();
            filterList.add(filter);
        };
        return filterList;
    }

    private ScanSettings createScanSettings() {
        ScanSettings settings = new ScanSettings.Builder()
                .setScanMode(ScanSettings.SCAN_MODE_BALANCED)
                .build();
        return settings;
    }

    @Override
    public void stopScanning() {
        bluetoothLeScanner.stopScan(mCallback);
    }

    @Override
    public List<BTDevice> getFoundDeviceList() {
        return mBluetoothDeviceList;
```

```
    }

    public class ScanCallback extends android.bluetooth.le.ScanCallback {

        @Override
        public void onScanResult(int callbackType, ScanResult result) {
            super.onScanResult(callbackType, result);
            if (result == null) {
                return;
            }
            BTDevice device = new BTDevice();
            device.setAddress(result.getDevice().getAddress());
            device.setName(new
StringBuffer(result.getScanRecord().getDeviceName()).toString());
            if (device == null || device.getAddress() == null) {
                return;
            }
            if (isAlreadyAdded(device)) {
                return;
            }
            mBluetoothDeviceList.add(device);
        }

        private boolean isAlreadyAdded(BTDevice bluetoothDevice) {
            for (BTDevice device : mBluetoothDeviceList) {
                String alreadyAddedDeviceMACAddress = device.getAddress();
                String newDeviceMACAddress = bluetoothDevice.getAddress();
                if (alreadyAddedDeviceMACAddress.equals(newDeviceMACAddress)) {
                    return true;
                }
            }
            return false;
        }
    }
}
```

5. Obtenga la lista de dispositivos encontrados llamando a:

```
scanningFactory.startScanning({uuidlist});

wait few seconds...

List<BTDevice> bluetoothDeviceList = scanningFactory.getFoundDeviceList();
```

Lea Bluetooth y Bluetooth LE API en línea: https://riptutorial.com/es/android/topic/2462/bluetooth-y-bluetooth-le-api

Capítulo 40: Botón

Sintaxis

- <Botón ... />
- Android: onClick = "nombre de método"
- button.setOnClickListener (nuevo OnClickListener () {...});
- clase pública nombre de clase implementa View.OnLongClickListener

Examples

en línea enClickListener

Supongamos que tenemos un botón (podemos crearlo mediante programación o vincularlo desde una vista mediante findViewbyId (), etc.)

```
Button btnOK = (...)
```

Ahora, crea una clase anónima y configúrala en línea.

```
btnOk.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Do stuff here...
    }
});
```

Usando el diseño para definir una acción de clic

Cuando creamos un botón en el diseño, podemos usar el atributo `android:onClick` para hacer referencia a un método en el código para manejar los clics.

Botón

```
<Button
    android:width="120dp"
    android:height="wrap_content"
    android:text="Click me"
    android:onClick="handleClick" />
```

Luego, en tu actividad, crea el método `handleClick` .

```
public void handleClick(View v) {
    // Do whatever.
}
```

Usando el mismo evento de clic para una o más Vistas en el XML

Cuando creamos una Vista en diseño, podemos usar el atributo android: onClick para hacer referencia a un método en la actividad asociada o fragmento para manejar los eventos de clic.

Diseño XML

```
<Button android:id="@+id/button"
    ...
    // onClick should reference the method in your activity or fragment
    android:onClick="doSomething" />

// Note that this works with any class which is a subclass of View, not just Button
<ImageView android:id="@+id/image"
    ...
    android:onClick="doSomething" />
```

Código de actividad / fragmento

En su **código** , cree el método que nombró, donde v será la vista que se tocó, y haga algo para cada vista que llame a este método.

```
public void doSomething(View v) {
    switch(v.getId()) {
        case R.id.button:
            // Button was clicked, do something.
            break;
        case R.id.image:
            // Image was clicked, do something else.
            break;
    }
}
```

Si lo desea, también puede usar un método diferente para cada Vista (en este caso, por supuesto, no tiene que verificar la ID).

Escuchando los eventos de clic largo

Para capturar un clic prolongado y usarlo, debe proporcionar el oyente adecuado al botón:

```
View.OnLongClickListener listener = new View.OnLongClickListener() {
    public boolean onLongClick(View v) {
        Button clickedButton = (Button) v;
        String buttonText = clickedButton.getText().toString();
        Log.v(TAG, "button long pressed --> " + buttonText);
        return true;
    }
};

button.setOnLongClickListener(listener);
```

Definiendo el oyente externo

¿Cuándo debo usarlo?

- Cuando el código dentro de un oyente en línea es demasiado grande y su método / clase se vuelve feo y difícil de leer
- Desea realizar la misma acción en varios elementos (vista) de su aplicación

---

Para lograr esto, necesita crear una clase que implemente uno de los escuchas en la API View .

Por ejemplo, brinde ayuda cuando haga clic en cualquier elemento:

```
public class HelpLongClickListener implements View.OnLongClickListener
{
    public HelpLongClickListener() {
    }

    @Override
    public void onLongClick(View v) {
        // show help toast or popup
    }
}
```

Entonces solo necesita tener un atributo o variable en su `Activity` para usarlo:

```
HelpLongClickListener helpListener = new HelpLongClickListener(...);

button1.setOnClickListener(helpListener);
button2.setOnClickListener(helpListener);
label.setOnClickListener(helpListener);
button1.setOnClickListener(helpListener);
```

NOTA: la definición de escuchas en una clase separada tiene una desventaja, no puede acceder a los campos de la clase directamente, por lo que debe pasar los datos (contexto, vista) a través del constructor a menos que haga públicos los atributos o defina captadores.

Personalizado Click Listener para evitar múltiples clics rápidos

Para **evitar que un botón se dispare varias veces en un corto período de tiempo** (digamos 2 clics en 1 segundo, lo que puede causar problemas graves si no se controla el flujo), se puede implementar un ***SingleClickListener*** personalizado.

Este ClickListener establece un intervalo de tiempo específico como umbral (por ejemplo, 1000ms).
Cuando se hace clic en el botón, se ejecutará una comprobación para ver si el disparador se ejecutó en el último período de tiempo que definió, y si no, se activará.

```
public class SingleClickListener implements View.OnClickListener {

    protected int defaultInterval;
    private long lastTimeClicked = 0;
```

```
    public SingleClickListener() {
        this(1000);
    }

    public SingleClickListener(int minInterval) {
        this.defaultInterval = minInterval;
    }

    @Override
    public void onClick(View v) {
        if (SystemClock.elapsedRealtime() - lastTimeClicked < defaultInterval) {
            return;
        }
        lastTimeClicked = SystemClock.elapsedRealtime();
        performClick(v);
    }

    public abstract void performClick(View v);

}
```

Y en la clase, el SingleClickListener está asociado al botón en juego

```
myButton = (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(new SingleClickListener() {
    @Override
    public void performClick(View view) {
        // do stuff
    }
});
```

Personalizar estilo de botón

Hay muchas formas posibles de personalizar el aspecto de un botón. Este ejemplo presenta varias opciones:

Opción 0: usar ThemeOverlay (actualmente la forma más fácil / rápida)

Crea un nuevo estilo en tu archivo de estilos:

styles.xml

```
<resources>
    <style name="mybutton" parent="ThemeOverlay.AppCompat.Ligth">
        <!-- customize colorButtonNormal for the disable color -->
        <item name="colorButtonNormal">@color/colorbuttonnormal</item>
        <!-- customize colorAccent for the enabled color -->
        <item name="colorButtonNormal">@color/coloraccent</item>
    </style>
</resources>
```

Luego, en el diseño donde coloca su botón (por ejemplo, MainActivity):

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

            <Button
                android:id="@+id/mybutton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello"
                android:theme="@style/mybutton"
                style="@style/Widget.AppCompat.Button.Colored"/>

</LinearLayout>
```

Opción 1: Crea tu propio estilo de botón

En values / styles.xml, cree un nuevo estilo para su botón:

styles.xml

```
<resources>
        <style name="mybuttonstyle" parent="@android:style/Widget.Button">
            <item name="android:gravity">center_vertical|center_horizontal</item>
            <item name="android:textColor">#FFFFFFFF</item>
            <item name="android:shadowColor">#FF000000</item>
            <item name="android:shadowDx">0</item>
            <item name="android:shadowDy">-1</item>
            <item name="android:shadowRadius">0.2</item>
            <item name="android:textSize">16dip</item>
            <item name="android:textStyle">bold</item>
            <item name="android:background">@drawable/button</item>
        </style>
    </resources>
```

Luego, en el diseño donde coloca su botón (por ejemplo, en MainActivity):

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
```

```
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

            <Button
                android:id="@+id/mybutton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello"
                android:theme="@style/mybuttonstyle"/>

</LinearLayout>
```

Opción 2: Asigna un dibujo para cada uno de tus estados de botón

Cree un archivo xml en la carpeta dibujable llamada 'mybuttondrawable.xml' para definir el recurso dibujable de cada uno de los estados de sus botones:

drawable / mybutton.xml

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:state_enabled="false"
        android:drawable="@drawable/mybutton_disabled" />
    <item
        android:state_pressed="true"
        android:state_enabled="true"
        android:drawable="@drawable/mybutton_pressed" />
    <item
        android:state_focused="true"
        android:state_enabled="true"
        android:drawable="@drawable/mybutton_focused" />
    <item
        android:state_enabled="true"
        android:drawable="@drawable/mybutton_enabled" />
</selector>
```

Cada uno de esos elementos dibujables puede ser imágenes (por ejemplo, mybutton_disabled.png) o archivos xml definidos por usted y almacenados en la carpeta de elementos dibujables. Por ejemplo:

drawable / mybutton_disabled.xml

```
<?xml version="1.0" encoding="utf-8"?>

    <shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle">
        <gradient
            android:startColor="#F2F2F2"
            android:centerColor="#A4A4A4"
            android:endColor="#F2F2F2"
            android:angle="90"/>
```

```
        <padding android:left="7dp"
            android:top="7dp"
            android:right="7dp"
            android:bottom="7dp" />
        <stroke
            android:width="2dip"
            android:color="#FFFFFF" />
        <corners android:radius= "8dp" />
    </shape>
```

Luego, en el diseño donde coloca su botón (por ejemplo, MainActivity):

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

            <Button
                android:id="@+id/mybutton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello"
                android:background="@drawable/mybuttondrawable"/>

</LinearLayout>
```

Opción 3: Agregue su estilo de botón a su tema de aplicación

Puede anular el estilo de botón de Android predeterminado en la definición del tema de su aplicación (en values / styles.xml).

styles.xml

```
<resources>
     <style name="AppTheme" parent="android:Theme">
            <item name="colorPrimary">@color/colorPrimary</item>
            <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
            <item name="colorAccent">@color/colorAccent</item>
              <item name="android:button">@style/mybutton</item>
     </style>

     <style name="mybutton" parent="android:style/Widget.Button">
             <item name="android:gravity">center_vertical|center_horizontal</item>
                <item name="android:textColor">#FFFFFFFF</item>
                <item name="android:shadowColor">#FF000000</item>
```

```
                <item name="android:shadowDx">0</item>
                <item name="android:shadowDy">-1</item>
                <item name="android:shadowRadius">0.2</item>
                <item name="android:textSize">16dip</item>
                <item name="android:textStyle">bold</item>
                <item name="android:background">@drawable/anydrawable</item>
     </style>
</resources>
```

Opción 4: Superponer un color en el estilo de botón predeterminado programáticamente

Simplemente encuentre el botón en su actividad y aplique un filtro de color:

```
Button mybutton = (Button) findViewById(R.id.mybutton);
mybutton.getBackground().setColorFilter(anycolor, PorterDuff.Mode.MULTIPLY)
```

Puedes consultar diferentes modos de fusión aquí y buenos ejemplos aquí .

Lea Botón en línea: https://riptutorial.com/es/android/topic/5607/boton

Capítulo 41: Botón de acción flotante

Introducción

El botón de acción flotante se usa para un tipo especial de acción promovida, se anima en la pantalla como una pieza de material en expansión, de forma predeterminada. El icono dentro de él puede estar animado, y también FAB puede moverse de manera diferente a otros elementos de la interfaz de usuario debido a su importancia relativa. Un botón de acción flotante representa la acción principal en una aplicación que simplemente puede desencadenar una acción o navegar a algún lugar.

Parámetros

Parámetro	Detalle
`android.support.design:elevation`	Valor de elevación para el FAB. Puede ser una referencia a otro recurso, en la forma "@ [+] [paquete:] tipo / nombre" o un atributo de tema en la forma "? [Paquete:] tipo / nombre".
`android.support.design:fabSize`	Tamaño para la FAB.
`android.support.design:rippleColor`	Color de la ondulación para el FAB.
`android.support.design:useCompatPadding`	Habilitar el relleno de compat.

Observaciones

Los botones de acción flotantes se utilizan para un tipo especial de acción promovida. Se distinguen por un icono en forma de círculo que flota sobre la interfaz de usuario y tienen comportamientos de movimiento especiales relacionados con la transformación, el lanzamiento y el punto de anclaje de transferencia.

Solo se recomienda un botón de acción flotante por pantalla para representar la acción más común.

Antes de usar el `FloatingActionButton` debe agregar la dependencia de la biblioteca de soporte de diseño en el archivo `build.gradle` :

```
dependencies {
    compile 'com.android.support:design:25.1.0'
}
```

Documentación oficial:

https://developer.android.com/reference/android/support/design/widget/FloatingActionButton.html

Especificaciones de materiales de diseño:

https://material.google.com/components/buttons-floating-action-button.html

Examples

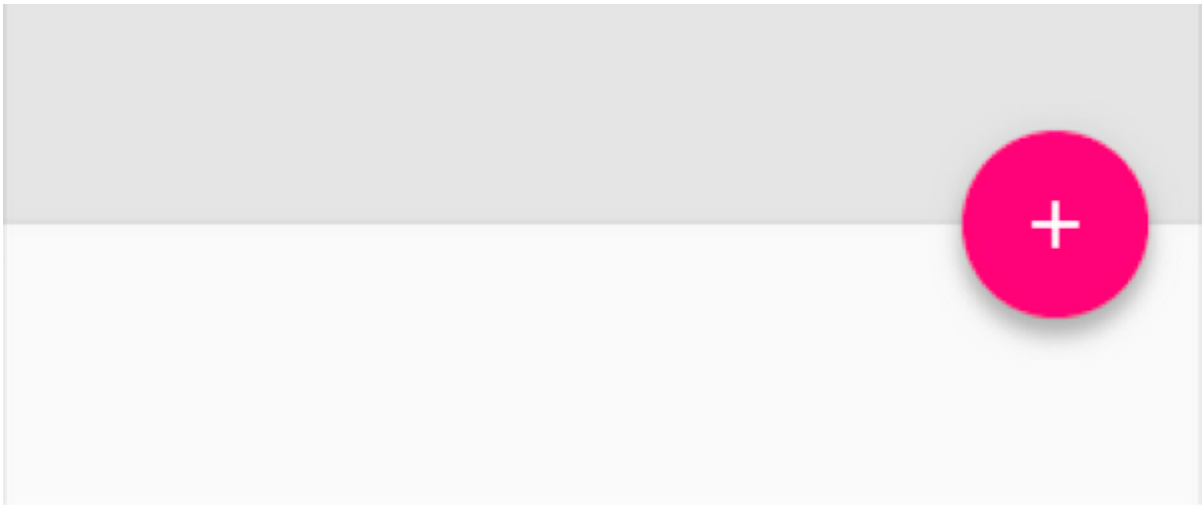
Cómo agregar el FAB al diseño

Para usar un FloatingActionButton, simplemente agregue la dependencia en el archivo `build.gradle` como se describe en la sección de comentarios.

A continuación, agregue al diseño:

```
<android.support.design.widget.FloatingActionButton
       android:id="@+id/fab"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_gravity="bottom|end"
       android:layout_margin="@dimen/fab_margin"
       android:src="@drawable/my_icon" />
```

Un ejemplo:



Color

El color de fondo de esta vista se establece de manera predeterminada en el colorAccent de su tema.

En la imagen de arriba, si el `src` solo apunta al ícono + (por defecto, 24x24 dp), para obtener el *color* de *fondo* del círculo completo puede usar la `app:backgroundTint="@color/your_colour"`

Si desea cambiar el color en el código que puede utilizar,

```
myFab.setBackgroundTintList(ColorStateList.valueOf(your color in int));
```

Si desea cambiar el color de FAB en estado presionado

```
mFab.setRippleColor(your color in int);
```

Posicionamiento

Se recomienda colocar un mínimo de 16dp desde el borde en el móvil y un mínimo de 24dp en la tableta / escritorio.

Nota : una vez que establezca un src excepto para cubrir el área completa de `FloatingActionButton` asegúrese de tener el tamaño correcto de esa imagen para obtener el mejor resultado.

El tamaño predeterminado del círculo es 56 x 56dp



Mini tamaño de círculo: 40 x 40dp

Si solo desea cambiar solo el icono Interior, use un icono de 24 x 24dp para el tamaño predeterminado

Mostrar y ocultar el botón de acción flotante al deslizar

Para mostrar y ocultar un `FloatingActionButton` con la animación predeterminada, simplemente llame a los métodos `show()` y `hide()` . Es una buena práctica mantener un `FloatingActionButton` en el diseño de la actividad en lugar de colocarlo en un fragmento, esto permite que las animaciones predeterminadas funcionen cuando se muestran y ocultan.

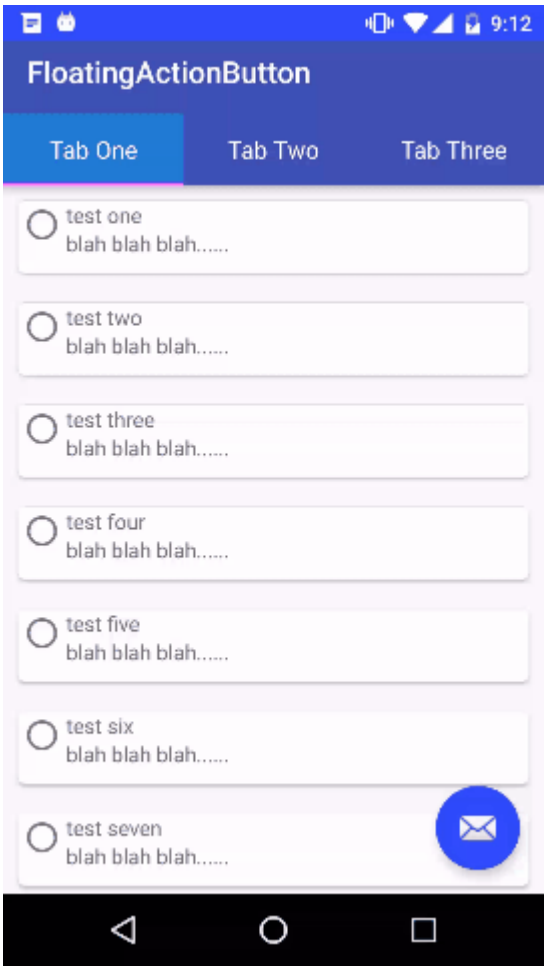
Aquí hay un ejemplo con un `ViewPager` :

- Tres pestañas
- Mostrar `FloatingActionButton` para la primera y tercera pestaña
- Ocultar el `FloatingActionButton` de `FloatingActionButton` en la pestaña central

```
public class MainActivity extends AppCompatActivity {

    FloatingActionButton fab;
    ViewPager viewPager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        fab = (FloatingActionButton) findViewById(R.id.fab);
        viewPager = (ViewPager) findViewById(R.id.viewpager);

        // ..... set up ViewPager .....
```

```
        viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {

            @Override
            public void onPageSelected(int position) {
                if (position == 0) {
                    fab.setImageResource(android.R.drawable.ic_dialog_email);
                    fab.show();
                } else if (position == 2) {
                    fab.setImageResource(android.R.drawable.ic_dialog_map);
                    fab.show();
                } else {
                    fab.hide();
                }
            }

            @Override
            public void onPageScrolled(int position, float positionOffset, int
positionOffsetPixels) {}

            @Override
            public void onPageScrollStateChanged(int state) {}
        });

        // Handle the FloatingActionButton click event:
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int position = viewPager.getCurrentItem();
                if (position == 0) {
                    openSend();
                } else if (position == 2) {
                    openMap();
                }
            }
        });

    }
}
```

Resultado:



Mostrar y ocultar el botón de acción flotante en el desplazamiento

A partir de la biblioteca de soporte, versión 22.2.1, es posible mostrar y ocultar un FloatingActionButton del comportamiento de desplazamiento utilizando una clase secundaria `FloatingActionButton.Behavior` que aprovecha los métodos `show()` y `hide()` .

Tenga en cuenta que esto solo funciona con un CoordinatorLayout junto con las vistas internas que admiten el desplazamiento anidado, como RecyclerView y NestedScrollView .

Esta clase `ScrollAwareFABBehavior` proviene de las Guías de Android en Codepath (se requiere cc-wiki con atribución)

```
public class ScrollAwareFABBehavior extends FloatingActionButton.Behavior {
    public ScrollAwareFABBehavior(Context context, AttributeSet attrs) {
        super();
    }

    @Override
    public boolean onStartNestedScroll(final CoordinatorLayout coordinatorLayout, final
FloatingActionButton child,
                                       final View directTargetChild, final View target, final
int nestedScrollAxes) {
        // Ensure we react to vertical scrolling
        return nestedScrollAxes == ViewCompat.SCROLL_AXIS_VERTICAL
                || super.onStartNestedScroll(coordinatorLayout, child, directTargetChild,
target, nestedScrollAxes);
    }
```

```
    @Override
    public void onNestedScroll(final CoordinatorLayout coordinatorLayout, final
FloatingActionButton child,
                               final View target, final int dxConsumed, final int dyConsumed,
                               final int dxUnconsumed, final int dyUnconsumed) {
        super.onNestedScroll(coordinatorLayout, child, target, dxConsumed, dyConsumed,
dxUnconsumed, dyUnconsumed);
        if (dyConsumed > 0 && child.getVisibility() == View.VISIBLE) {
            // User scrolled down and the FAB is currently visible -> hide the FAB
            child.hide();
        } else if (dyConsumed < 0 && child.getVisibility() != View.VISIBLE) {
            // User scrolled up and the FAB is currently not visible -> show the FAB
            child.show();
        }
    }
}
```

En el xml de diseño de FloatingActionButton, especifique la `app:layout_behavior` con el nombre de clase completamente calificado de `ScrollAwareFABBehavior` :

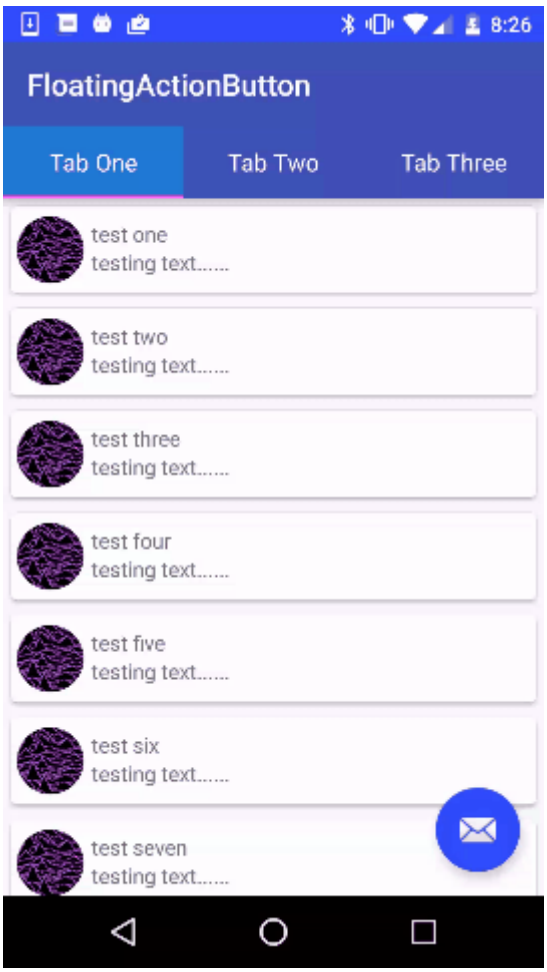
```
app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
```

Por ejemplo, con este diseño:

```
<android.support.design.widget.CoordinatorLayout
    android:id="@+id/main_layout"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <android.support.design.widget.AppBarLayout
        android:id="@+id/appBarLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:elevation="6dp">
        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:background="?attr/colorPrimary"
            android:minHeight="?attr/actionBarSize"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
            app:elevation="0dp"
            app:layout_scrollFlags="scroll|enterAlways"
            />

        <android.support.design.widget.TabLayout
            android:id="@+id/tab_layout"
            app:tabMode="fixed"
            android:layout_below="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
            android:background="?attr/colorPrimary"
            app:elevation="0dp"
            app:tabTextColor="#d3d3d3"
            android:minHeight="?attr/actionBarSize"
            />

    </android.support.design.widget.AppBarLayout>

    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager"
        android:layout_below="@+id/tab_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
        android:layout_margin="@dimen/fab_margin"
        android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

Aquí está el resultado:



Ajuste del comportamiento de FloatingActionButton

Puede establecer el comportamiento de la FAB en XML.

Por ejemplo:

```
<android.support.design.widget.FloatingActionButton
   app:layout_behavior=".MyBehavior" />
```

O puedes configurar programáticamente usando:

```
CoordinatorLayout.LayoutParams p = (CoordinatorLayout.LayoutParams) fab.getLayoutParams();
p.setBehavior(xxxx);
fab.setLayoutParams(p);
```

Lea Botón de acción flotante en línea: https://riptutorial.com/es/android/topic/2979/boton-de-accion-flotante

Capítulo 42: Caché de mapa de bits

Introducción

Caché de mapa de bits eficiente en memoria: esto es particularmente importante si su aplicación usa animaciones, ya que se detendrán durante la limpieza del GC y harán que su aplicación parezca lenta para el usuario. Un caché permite reutilizar objetos que son caros de crear. Si carga un objeto en la memoria, puede pensar en esto como un caché para el objeto. Trabajar con un mapa de bits en Android es complicado. Es más importante almacenar el bimap en caché si lo va a usar repetidamente.

Sintaxis

- `LruCache<String, Bitmap> mMemoryCache;//declaration of LruCache object.`
- void addBitmapToMemoryCache (clave de cadena, mapa de bits de mapa de bits) {} // declaración del método genérico que agrega un mapa de bits en la memoria caché
- Bitmap getBitmapFromMemCache (String key) {} // declaración del método genérico para obtener bimap desde el caché.

Parámetros

Parámetro	Detalles
llave	clave para almacenar bitmap en memoria caché
mapa de bits	valor de mapa de bits que se almacenará en la memoria caché

Examples

Caché de mapa de bits utilizando caché LRU

Caché LRU

El siguiente código de ejemplo muestra una posible implementación de la clase LruCache para almacenar imágenes en caché.

```
private LruCache<String, Bitmap> mMemoryCache;
```

Aquí el valor de cadena es clave para el valor de mapa de bits.

```
// Get max available VM memory, exceeding this amount will throw an
// OutOfMemory exception. Stored in kilobytes as LruCache takes an
// int in its constructor.
final int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);
```

```
// Use 1/8th of the available memory for this memory cache.
final int cacheSize = maxMemory / 8;

mMemoryCache = new LruCache<String, Bitmap>(cacheSize) {
    @Override
    protected int sizeOf(String key, Bitmap bitmap) {
        // The cache size will be measured in kilobytes rather than
        // number of items.
        return bitmap.getByteCount() / 1024;
    }
};
```

Para agregar bitmap a la memoria caché

```
public void addBitmapToMemoryCache(String key, Bitmap bitmap) {
if (getBitmapFromMemCache(key) == null) {
        mMemoryCache.put(key, bitmap);
    }
}
```

Para obtener bitmap desde la memoria caché

```
public Bitmap getBitmapFromMemCache(String key) {
    return mMemoryCache.get(key);
}
```

Para cargar el mapa de bits en la vista de imagen, use **getBitmapFromMemCache ("Clave de acceso").**

Lea Caché de mapa de bits en línea: https://riptutorial.com/es/android/topic/9901/cache-de-mapa-de-bits

Capítulo 43: Camara y galeria

Examples

Tomando fotos de tamaño completo de la cámara

Para tomar una foto, primero debemos declarar los permisos necesarios en `AndroidManifest.xml` . Necesitamos dos permisos:

- `Camera` - para abrir la aplicación de la cámara. Si el atributo `required` se establece en `true` , no podrá instalar esta aplicación si no tiene una cámara de hardware.
- `WRITE_EXTERNAL_STORAGE` : este permiso es necesario para crear un nuevo archivo, en el que se guardará la foto capturada.

AndroidManifest.xml

```
<uses-feature android:name="android.hardware.camera"
          android:required="true" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

La idea principal al tomar una foto de tamaño completo de la cámara es que necesitamos crear un nuevo archivo para la foto, antes de abrir la aplicación de la cámara y capturar la foto.

```
private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    // Ensure that there's a camera activity to handle the intent
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        // Create the File where the photo should go
        File photoFile = null;
        try {
            photoFile = createImageFile();
        } catch (IOException ex) {
            Log.e("DEBUG_TAG", "createFile", ex);
        }
        // Continue only if the File was successfully created
        if (photoFile != null) {
            takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(photoFile));
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
        }
    }
}

private File createImageFile() throws IOException {
    // Create an image file name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.getDefault()).format(new
Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = getAlbumDir();
    File image = File.createTempFile(
            imageFileName,  /* prefix */
            ".jpg",         /* suffix */
```

```
            storageDir      /* directory */
    );

    // Save a file: path for use with ACTION_VIEW intents
    mCurrentPhotoPath = image.getAbsolutePath();
    return image;
}

private File getAlbumDir() {
    File storageDir = null;

    if (Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())) {

        storageDir = new File(Environment.getExternalStorageDirectory()
                + "/dcim/"
                + "MyRecipes");

        if (!storageDir.mkdirs()) {
            if (!storageDir.exists()) {
                Log.d("CameraSample", "failed to create directory");
                return null;
            }
        }

    } else {
        Log.v(getString(R.string.app_name), "External storage is not mounted READ/WRITE.");
    }

    return storageDir;
}

private void setPic() {

    /* There isn't enough memory to open up more than a couple camera photos */
    /* So pre-scale the target bitmap into which the file is decoded */

    /* Get the size of the ImageView */
    int targetW = recipeImage.getWidth();
    int targetH = recipeImage.getHeight();

    /* Get the size of the image */
    BitmapFactory.Options bmOptions = new BitmapFactory.Options();
    bmOptions.inJustDecodeBounds = true;
    BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
    int photoW = bmOptions.outWidth;
    int photoH = bmOptions.outHeight;

    /* Figure out which way needs to be reduced less */
    int scaleFactor = 2;
    if ((targetW > 0) && (targetH > 0)) {
        scaleFactor = Math.max(photoW / targetW, photoH / targetH);
    }

    /* Set bitmap options to scale the image decode target */
    bmOptions.inJustDecodeBounds = false;
    bmOptions.inSampleSize = scaleFactor;
    bmOptions.inPurgeable = true;

    Matrix matrix = new Matrix();
    matrix.postRotate(getRotation());
```

```
    /* Decode the JPEG file into a Bitmap */
    Bitmap bitmap = BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
    bitmap = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(), matrix,
false);

    /* Associate the Bitmap to the ImageView */
    recipeImage.setImageBitmap(bitmap);
}

private float getRotation() {
    try {
        ExifInterface ei = new ExifInterface(mCurrentPhotoPath);
        int orientation = ei.getAttributeInt(ExifInterface.TAG_ORIENTATION,
ExifInterface.ORIENTATION_NORMAL);

        switch (orientation) {
            case ExifInterface.ORIENTATION_ROTATE_90:
                return 90f;
            case ExifInterface.ORIENTATION_ROTATE_180:
                return 180f;
            case ExifInterface.ORIENTATION_ROTATE_270:
                return 270f;
            default:
                return 0f;
        }
    } catch (Exception e) {
        Log.e("Add Recipe", "getRotation", e);
        return 0f;
    }
}

private void galleryAddPic() {
    Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
    File f = new File(mCurrentPhotoPath);
    Uri contentUri = Uri.fromFile(f);
    mediaScanIntent.setData(contentUri);
    sendBroadcast(mediaScanIntent);
}

private void handleBigCameraPhoto() {

    if (mCurrentPhotoPath != null) {
        setPic();
        galleryAddPic();
    }
}


@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == Activity.RESULT_OK) {
        handleBigCameraPhoto();
    }
}
```

Tomar foto

Agregue un permiso para acceder a la cámara al archivo AndroidManifest:

```
<uses-permission android:name="android.permission.CAMERA"></uses-permission>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Archivo xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<SurfaceView android:id="@+id/surfaceView" android:layout_height="0dip"
android:layout_width="0dip"></SurfaceView>
<ImageView android:layout_width="wrap_content" android:layout_height="wrap_content"
android:id="@+id/imageView"></ImageView>
</LinearLayout>
```

Actividad

```
import java.io.IOException;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.hardware.Camera;
import android.hardware.Camera.Parameters;
import android.os.Bundle;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.widget.ImageView;

public class TakePicture extends Activity implements SurfaceHolder.Callback
{
    //a variable to store a reference to the Image View at the main.xml file
    private ImageView iv_image;
    //a variable to store a reference to the Surface View at the main.xml file
    private SurfaceView sv;

    //a bitmap to display the captured image
    private Bitmap bmp;

    //Camera variables
    //a surface holder
    private SurfaceHolder sHolder;
    //a variable to control the camera
    private Camera mCamera;
    //the camera parameters
    private Parameters parameters;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //get the Image View at the main.xml file
        iv_image = (ImageView) findViewById(R.id.imageView);
```

```
        //get the Surface View at the main.xml file
        sv = (SurfaceView) findViewById(R.id.surfaceView);

        //Get a surface
        sHolder = sv.getHolder();

        //add the callback interface methods defined below as the Surface View callbacks
        sHolder.addCallback(this);

        //tells Android that this surface will have its data constantly replaced
        sHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }

    @Override
    public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3)
    {
         //get camera parameters
         parameters = mCamera.getParameters();

         //set camera parameters
         mCamera.setParameters(parameters);
         mCamera.startPreview();

         //sets what code should be executed after the picture is taken
         Camera.PictureCallback mCall = new Camera.PictureCallback()
         {
             @Override
             public void onPictureTaken(byte[] data, Camera camera)
             {
                 //decode the data obtained by the camera into a Bitmap
                 bmp = BitmapFactory.decodeByteArray(data, 0, data.length);
                String filename=Environment.getExternalStorageDirectory()
                                + File.separator + "testimage.jpg";
                FileOutputStream out = null;
                try {
                    out = new FileOutputStream(filename);
                    bmp.compress(Bitmap.CompressFormat.PNG, 100, out); // bmp is your Bitmap
instance
                    // PNG is a lossless format, the compression factor (100) is ignored
                } catch (Exception e) {
                    e.printStackTrace();
                } finally {
                    try {
                        if (out != null) {
                            out.close();
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
                 //set the iv_image
                 iv_image.setImageBitmap(bmp);
             }
         };

         mCamera.takePicture(null, null, mCall);
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder)
    {
```

```
        // The Surface has been created, acquire the camera and tell it where
        // to draw the preview.
        mCamera = Camera.open();
        try {
           mCamera.setPreviewDisplay(holder);

        } catch (IOException exception) {
            mCamera.release();
            mCamera = null;
        }
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder)
    {
        //stop the preview
        mCamera.stopPreview();
        //release the camera
        mCamera.release();
        //unbind the camera from this object
        mCamera = null;
    }
}
```

Cómo iniciar la cámara o la galería y guardar el resultado de la cámara en el almacenamiento

En primer lugar, necesita `Uri` y carpetas temporales y códigos de solicitud:

```
public final int REQUEST_SELECT_PICTURE = 0x01;
public final int REQUEST_CODE_TAKE_PICTURE = 0x2;
public static String TEMP_PHOTO_FILE_NAME ="photo_";
Uri mImageCaptureUri;
File mFileTemp;
```

Entonces inicia mFileTemp:

```
public void initTempFile(){
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {

        mFileTemp = new File(Environment.getExternalStorageDirectory() + File.separator
                + getResources().getString(R.string.app_foldername) + File.separator
                + getResources().getString(R.string.pictures_folder)
                , TEMP_PHOTO_FILE_NAME
                + System.currentTimeMillis() + ".jpg");
        mFileTemp.getParentFile().mkdirs();
    } else {
        mFileTemp = new File(getFilesDir() + File.separator
                + getResources().getString(R.string.app_foldername)
                + File.separator + getResources().getString(R.string.pictures_folder)
                , TEMP_PHOTO_FILE_NAME + System.currentTimeMillis() + ".jpg");
        mFileTemp.getParentFile().mkdirs();
    }
}
```

Apertura de la `Camera` y la `Gallery` intenciones:

```
public void openCamera(){
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    try {
        mImageCaptureUri = null;
        String state = Environment.getExternalStorageState();
        if (Environment.MEDIA_MOUNTED.equals(state)) {
            mImageCaptureUri = Uri.fromFile(mFileTemp);

        } else {

            mImageCaptureUri = InternalStorageContentProvider.CONTENT_URI;

        }
        intent.putExtra(MediaStore.EXTRA_OUTPUT, mImageCaptureUri);
        intent.putExtra("return-data", true);
        startActivityForResult(intent, REQUEST_CODE_TAKE_PICTURE);
    } catch (Exception e) {

        Log.d("error", "cannot take picture", e);
    }
}

public void openGallery(){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN
            && ActivityCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE)
            != PackageManager.PERMISSION_GRANTED) {
        requestPermission(Manifest.permission.READ_EXTERNAL_STORAGE,
                getString(R.string.permission_read_storage_rationale),
                REQUEST_STORAGE_READ_ACCESS_PERMISSION);
    } else {
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        intent.addCategory(Intent.CATEGORY_OPENABLE);
        startActivityForResult(Intent.createChooser(intent, getString(R.string.select_image)),
REQUEST_SELECT_PICTURE);
    }

}
```

Luego en el método `onActivityResult` :

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    if (resultCode != RESULT_OK) {
        return;
    }
    Bitmap bitmap;

    switch (requestCode) {

        case REQUEST_SELECT_PICTURE:
            try {
                Uri uri = data.getData();
                try {
                    bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), uri);
                    Bitmap bitmapScaled = Bitmap.createScaledBitmap(bitmap, 800, 800, true);
                    Drawable drawable=new BitmapDrawable(bitmapScaled);
```

```
                    mImage.setImageDrawable(drawable);
                    mImage.setVisibility(View.VISIBLE);
                } catch (IOException e) {
                    Log.v("act result", "there is an error : "+e.getContent());
                }
            } catch (Exception e) {
                Log.v("act result", "there is an error : "+e.getContent());
            }
            break;
        case REQUEST_CODE_TAKE_PICTURE:
            try{
               Bitmap bitmappicture = MediaStore.Images.Media.getBitmap(getContentResolver() ,
mImageCaptureUri);
               mImage.setImageBitmap(bitmappicture);
               mImage.setVisibility(View.VISIBLE);
           }catch (IOException e){
              Log.v("error camera",e.getMessage());
           }
           break;
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

Necesitas estos permisos en `AndroidManifest.xml` :

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />
```

Y necesita manejar permisos de tiempo de ejecución tales como lectura / escritura de almacenamiento externo, etc.

Estoy comprobando el permiso `READ_EXTERNAL_STORAGE` en mi método `openGallery` :

Mi método de `requestPermission` :

```
protected void requestPermission(final String permission, String rationale, final int
requestCode) {
    if (ActivityCompat.shouldShowRequestPermissionRationale(this, permission)) {
        showAlertDialog(getString(R.string.permission_title_rationale), rationale,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        ActivityCompat.requestPermissions(BasePermissionActivity.this,
                                new String[]{permission}, requestCode);
                    }
                }, getString(android.R.string.ok), null, getString(android.R.string.cancel));
    } else {
        ActivityCompat.requestPermissions(this, new String[]{permission}, requestCode);
    }
}
```

A continuación, anule el método `onRequestPermissionsResult` :

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
```

```
@NonNull int[] grantResults) {
    switch (requestCode) {
        case REQUEST_STORAGE_READ_ACCESS_PERMISSION:
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                handleGallery();
            }
            break;
        default:
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

Método `showAlertDialog` :

```
protected void showAlertDialog(@Nullable String title, @Nullable String message,
                               @Nullable DialogInterface.OnClickListener
onPositiveButtonClickListener,
                               @NonNull String positiveText,
                               @Nullable DialogInterface.OnClickListener
onNegativeButtonClickListener,
                               @NonNull String negativeText) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle(title);
    builder.setMessage(message);
    builder.setPositiveButton(positiveText, onPositiveButtonClickListener);
    builder.setNegativeButton(negativeText, onNegativeButtonClickListener);
    mAlertDialog = builder.show();
}
```

Establecer la resolución de la cámara

Establecer alta resolución programáticamente.

```
Camera mCamera = Camera.open();
Camera.Parameters params = mCamera.getParameters();

// Check what resolutions are supported by your camera
List<Size> sizes = params.getSupportedPictureSizes();

// Iterate through all available resolutions and choose one.
// The chosen resolution will be stored in mSize.
Size mSize;
for (Size size : sizes) {
    Log.i(TAG, "Available resolution: "+size.width+" "+size.height);
        mSize = size;
    }
}

Log.i(TAG, "Chosen resolution: "+mSize.width+" "+mSize.height);
params.setPictureSize(mSize.width, mSize.height);
mCamera.setParameters(params);
```

Decodifique el mapa de bits correctamente girado desde el uri obtenido con la intención

```
private static final String TAG = "IntentBitmapFetch";
private static final String COLON_SEPARATOR = ":";
private static final String IMAGE = "image";

@Nullable
public Bitmap getBitmap(@NonNull Uri bitmapUri, int maxDimen) {
    InputStream is = context.getContentResolver().openInputStream(bitmapUri);
    Bitmap bitmap = BitmapFactory.decodeStream(is, null, getBitmapOptions(bitmapUri,
maxDimen));

    int imgRotation = getImageRotationDegrees(bitmapUri);

    int endRotation = (imgRotation < 0) ? -imgRotation : imgRotation;
    endRotation %= 360;
    endRotation = 90 * (endRotation / 90);
    if (endRotation > 0 && bitmap != null) {
        Matrix m = new Matrix();
        m.setRotate(endRotation);
        Bitmap tmp = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(),
m, true);
        if (tmp != null) {
            bitmap.recycle();
            bitmap = tmp;
        }
    }

    return bitmap;
}

private BitmapFactory.Options getBitmapOptions(Uri uri, int imageMaxDimen){
    BitmapFactory.Options options = new BitmapFactory.Options();
    if (imageMaxDimen > 0) {
        options.inJustDecodeBounds = true;
        decodeImage(null, uri, options);
        options.inSampleSize = calculateScaleFactor(options, imageMaxDimen);
        options.inJustDecodeBounds = false;
        options.inPreferredConfig = Bitmap.Config.RGB_565;
        addInBitmapOptions(options);
    }
}

private int calculateScaleFactor(@NonNull BitmapFactory.Options bitmapOptionsMeasureOnly, int
imageMaxDimen) {
    int inSampleSize = 1;
    if (bitmapOptionsMeasureOnly.outHeight > imageMaxDimen ||
bitmapOptionsMeasureOnly.outWidth > imageMaxDimen) {
        final int halfHeight = bitmapOptionsMeasureOnly.outHeight / 2;
        final int halfWidth = bitmapOptionsMeasureOnly.outWidth / 2;
        while ((halfHeight / inSampleSize) > imageMaxDimen && (halfWidth / inSampleSize) >
imageMaxDimen) {
            inSampleSize *= 2;
        }
    }
    return inSampleSize;
}

    public int getImageRotationDegrees(@NonNull Uri imgUri) {
    int photoRotation = ExifInterface.ORIENTATION_UNDEFINED;

    try {
        boolean hasRotation = false;
```

```
        //If image comes from the gallery and is not in the folder DCIM (Scheme: content://)
        String[] projection = {MediaStore.Images.ImageColumns.ORIENTATION};
        Cursor cursor = context.getContentResolver().query(imgUri, projection, null, null,
null);
        if (cursor != null) {
            if (cursor.getColumnCount() > 0 && cursor.moveToFirst()) {
                photoRotation = cursor.getInt(cursor.getColumnIndex(projection[0]));
                hasRotation = photoRotation != 0;
                Log.d("Cursor orientation: "+ photoRotation);
            }
            cursor.close();
        }

        //If image comes from the camera (Scheme: file://) or is from the folder DCIM (Scheme:
content://)
        if (!hasRotation) {
            ExifInterface exif = new ExifInterface(getAbsolutePath(imgUri));
            int exifRotation = exif.getAttributeInt(ExifInterface.TAG_ORIENTATION,
                    ExifInterface.ORIENTATION_NORMAL);
            switch (exifRotation) {
                case ExifInterface.ORIENTATION_ROTATE_90: {
                    photoRotation = 90;
                    break;
                }
                case ExifInterface.ORIENTATION_ROTATE_180: {
                    photoRotation = 180;
                    break;
                }
                case ExifInterface.ORIENTATION_ROTATE_270: {
                    photoRotation = 270;
                    break;
                }
            }
            Log.d(TAG, "Exif orientation: "+ photoRotation);
        }
    } catch (IOException e) {
        Log.e(TAG, "Error determining rotation for image"+ imgUri, e);
    }
    return photoRotation;
}

@TargetApi(Build.VERSION_CODES.KITKAT)
private String getAbsolutePath(Uri uri) {
    //Code snippet edited from: http://stackoverflow.com/a/20559418/2235133
    String filePath = uri.getPath();
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT &&
DocumentsContract.isDocumentUri(context, uri)) {
        // Will return "image:x*"
        String[] wholeID = TextUtils.split(DocumentsContract.getDocumentId(uri),
COLON_SEPARATOR);
        // Split at colon, use second item in the array
        String type = wholeID[0];
        if (IMAGE.equalsIgnoreCase(type)) {//If it not type image, it means it comes from a
remote location, like Google Photos
            String id = wholeID[1];
            String[] column = {MediaStore.Images.Media.DATA};
            // where id is equal to
            String sel = MediaStore.Images.Media._ID + "=?";
            Cursor cursor = context.getContentResolver().
                    query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
                            column, sel, new String[]{id}, null);
```

```
            if (cursor != null) {
                int columnIndex = cursor.getColumnIndex(column[0]);
                if (cursor.moveToFirst()) {
                    filePath = cursor.getString(columnIndex);
                }
                cursor.close();
            }
            Log.d(TAG, "Fetched absolute path for uri" + uri);
        }
    }
    return filePath;
}
```

Lea Camara y galeria en línea: https://riptutorial.com/es/android/topic/4789/camara-y-galeria

Capítulo 44: Cambios de orientación

Observaciones

Referencia: https://guides.codepath.com/android/Handling-Configuration-Changes#references

Examples

Ahorro y restauración del estado de actividad

A medida que su actividad comienza a detenerse, el sistema llama a Guardar Estado de estado de `onSaveInstanceState()` para que su actividad pueda guardar información de estado con una colección de pares clave-valor. La implementación predeterminada de este método guarda automáticamente la información sobre el estado de la jerarquía de vista de la actividad, como el texto en un widget `EditText` o la posición de desplazamiento de un `ListView` .

Para guardar información de estado adicional para su actividad, debe implementar `onSaveInstanceState()` y agregar pares clave-valor al objeto Bundle. Por ejemplo:

```
public class MainActivity extends Activity {
    static final String SOME_VALUE = "int_value";
    static final String SOME_OTHER_VALUE = "string_value";

    @Override
    protected void onSaveInstanceState(Bundle savedInstanceState) {
        // Save custom values into the bundle
        savedInstanceState.putInt(SOME_VALUE, someIntValue);
        savedInstanceState.putString(SOME_OTHER_VALUE, someStringValue);
        // Always call the superclass so it can save the view hierarchy state
        super.onSaveInstanceState(savedInstanceState);
    }
}
```

El sistema llamará a ese método antes de que se destruya una Actividad. Luego, más tarde, el sistema invocará `onRestoreInstanceState` donde podremos restaurar el estado del paquete:

```
@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
    // Always call the superclass so it can restore the view hierarchy
    super.onRestoreInstanceState(savedInstanceState);
    // Restore state members from saved instance
    someIntValue = savedInstanceState.getInt(SOME_VALUE);
    someStringValue = savedInstanceState.getString(SOME_OTHER_VALUE);
}
```

El estado de instancia también se puede restaurar en el método estándar de Actividad # onCreate, pero es conveniente hacerlo en `onRestoreInstanceState` que garantiza que se haya realizado toda la inicialización y permita que las subclases decidan si usar la implementación predeterminada. Lea esta publicación de stackoverflow para más detalles.

Tenga en cuenta que no se garantiza que se llame a `onSaveInstanceState` y `onRestoreInstanceState` . Android invoca `onSaveInstanceState()` cuando existe la posibilidad de que la actividad se destruya. Sin embargo, hay casos en los que se llama `onSaveInstanceState` pero la actividad no se destruye y, como resultado, no se invoca `onRestoreInstanceState` .

Guardando y restaurando el estado del fragmento

Los fragmentos también tienen un método `onSaveInstanceState()` que se llama cuando es necesario guardar su estado:

```
public class MySimpleFragment extends Fragment {
    private int someStateValue;
    private final String SOME_VALUE_KEY = "someValueToSave";

    // Fires when a configuration change occurs and fragment needs to save state
    @Override
    protected void onSaveInstanceState(Bundle outState) {
        outState.putInt(SOME_VALUE_KEY, someStateValue);
        super.onSaveInstanceState(outState);
    }
}
```

Luego podemos extraer datos de este estado guardado en `onCreateView` :

```
public class MySimpleFragment extends Fragment {
   // ...

   // Inflate the view for the fragment based on layout XML
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.my_simple_fragment, container, false);
        if (savedInstanceState != null) {
            someStateValue = savedInstanceState.getInt(SOME_VALUE_KEY);
            // Do something with value if needed
        }
        return view;
   }
}
```

Para que el estado del fragmento se guarde correctamente, debemos asegurarnos de que no estamos recreando innecesariamente el fragmento en los cambios de configuración. Esto significa tener cuidado de no reinicializar los fragmentos existentes cuando ya existen. Cualquier fragmento que se inicialice en una Actividad debe buscarse por etiqueta después de un cambio de configuración:

```
public class ParentActivity extends AppCompatActivity {
    private MySimpleFragment fragmentSimple;
    private final String SIMPLE_FRAGMENT_TAG = "myfragmenttag";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        if (savedInstanceState != null) { // saved instance state, fragment may exist
           // look up the instance that already exists by tag
```

```
            fragmentSimple = (MySimpleFragment)
               getSupportFragmentManager().findFragmentByTag(SIMPLE_FRAGMENT_TAG);
        } else if (fragmentSimple == null) {
            // only create fragment if they haven't been instantiated already
            fragmentSimple = new MySimpleFragment();
        }
    }
}
```

Esto requiere que tengamos cuidado de incluir una etiqueta para la búsqueda cada vez que coloquemos un fragmento en la actividad dentro de una transacción:

```
public class ParentActivity extends AppCompatActivity {
    private MySimpleFragment fragmentSimple;
    private final String SIMPLE_FRAGMENT_TAG = "myfragmenttag";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // ... fragment lookup or instantation from above...
        // Always add a tag to a fragment being inserted into container
        if (!fragmentSimple.isInLayout()) {
            getSupportFragmentManager()
                .beginTransaction()
                .replace(R.id.container, fragmentSimple, SIMPLE_FRAGMENT_TAG)
                .commit();
        }
    }
}
```

Con este patrón simple, podemos reutilizar correctamente los fragmentos y restaurar su estado a través de los cambios de configuración.

Fragmentos de retención

En muchos casos, podemos evitar problemas cuando una Actividad se vuelve a crear simplemente usando fragmentos. Si sus puntos de vista y su estado están dentro de un fragmento, podemos conservar fácilmente el fragmento cuando la actividad se vuelva a crear:

```
public class RetainedFragment extends Fragment {
    // data object we want to retain
    private MyDataObject data;

    // this method is only called once for this fragment
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // retain this fragment when activity is re-initialized
        setRetainInstance(true);
    }

    public void setData(MyDataObject data) {
        this.data = data;
    }

    public MyDataObject getData() {
        return data;
```

```
    }
}
```

Este enfoque evita que el fragmento se destruya durante el ciclo de vida de la actividad. En su lugar, se mantienen dentro del Administrador de fragmentos. Ver los documentos oficiales de Android para más información .

Ahora puede verificar si el fragmento ya existe por etiqueta antes de crear uno y el fragmento conservará su estado en todos los cambios de configuración. Consulte la guía Manipulación de cambios en el tiempo de ejecución para obtener más detalles .

Orientación de la pantalla de bloqueo

Si desea bloquear el cambio de orientación de la pantalla de cualquier pantalla (actividad) de su aplicación de Android, solo necesita configurar la propiedad `android:screenOrientation` de una `<activity>` dentro del **AndroidManifest.xml** :

```
<activity
    android:name="com.techblogon.screenorientationexample.MainActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
    <!-- ... -->
</activity>
```

Ahora esa actividad está obligada a mostrarse siempre en modo " **retrato** ".

Gestionar manualmente los cambios de configuración

Si su aplicación no necesita actualizar los recursos durante un cambio de configuración específico y usted tiene una limitación de rendimiento que requiere que evite el reinicio de la actividad, entonces puede declarar que su actividad maneja el cambio de configuración en sí, lo que evita que el sistema reinicie su actividad.

Sin embargo, esta técnica debe considerarse un último recurso cuando debe evitar reinicios debido a un cambio de configuración y no se recomienda para la mayoría de las aplicaciones. Para adoptar este enfoque, debemos agregar el nodo `android:configChanges` a la actividad dentro de **AndroidManifest.xml** :

```
<activity android:name=".MyActivity"
          android:configChanges="orientation|screenSize|keyboardHidden"
          android:label="@string/app_name">
```

Ahora, cuando una de estas configuraciones cambia, la actividad no se reinicia sino que recibe una llamada a `onConfigurationChanged()` :

```
// Within the activity which receives these changes
// Checks the current device orientation, and toasts accordingly
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
```

```
    // Checks the orientation of the screen
    if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
        Toast.makeText(this, "landscape", Toast.LENGTH_SHORT).show();
    } else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT){
        Toast.makeText(this, "portrait", Toast.LENGTH_SHORT).show();
    }
}
```

Ver la documentación de Manipulación del cambio . Para obtener más información sobre qué cambios de configuración puede manejar en su actividad, consulte la documentación de android: configChanges y la clase de configuración .

Manejo AsyncTask

Problema:

- Si después de que se inicie `AsyncTask` se produce una rotación de pantalla, la actividad propietaria se destruye y se `AsyncTask` crear.
- Cuando finaliza `AsyncTask` , desea actualizar la interfaz de usuario que puede que ya no sea válida.

Solución:

Usando los cargadores , uno puede superar fácilmente la actividad de destrucción / recreación.

Ejemplo:

Actividad principal:

```
public class MainActivity extends AppCompatActivity
        implements LoaderManager.LoaderCallbacks<Bitmap> {

    //Unique id for the loader
    private static final int MY_LOADER = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        LoaderManager loaderManager = getSupportLoaderManager();

        if(loaderManager.getLoader(MY_LOADER) == null) {
            loaderManager.initLoader(MY_LOADER, null, this).forceLoad();
        }
    }
```

```
    @Override
    public Loader<Bitmap> onCreateLoader(int id, Bundle args) {
        //Create a new instance of your Loader<Bitmap>
        MyLoader loader = new MyLoader(MainActivity.this);
        return loader;
    }

    @Override
    public void onLoadFinished(Loader<Bitmap> loader, Bitmap data) {
        // do something in the parent activity/service
        // i.e. display the downloaded image
        Log.d("MyAsyncTask", "Received result: ");
    }

    @Override
    public void onLoaderReset(Loader<Bitmap> loader) {

    }
}
```

AsyncTaskLoader:

```
public class MyLoader extends AsyncTaskLoader<Bitmap> {
    private WeakReference<Activity> motherActivity;

    public MyLoader(Activity activity) {
        super(activity);
        //We don't use this, but if you want you can use it, but remember, WeakReference
        motherActivity = new WeakReference<>(activity);
    }

    @Override
    public Bitmap loadInBackground() {
        // Do work. I.e download an image from internet to be displayed in gui.
        // i.e. return the downloaded gui
        return result;
    }
}
```

Nota:

Es importante utilizar la biblioteca de compatibilidad v4 o no, pero no use parte de una y parte de la otra, ya que dará lugar a errores de compilación. Para verificarlo, puede consultar las importaciones de `android.support.v4.content` y `android.content` (no debería tener ambos).

Bloquear la rotación de la pantalla programáticamente

Es muy común que durante el desarrollo, uno pueda encontrar **muy útil bloquear / desbloquear la pantalla del dispositivo durante partes específicas del código** .

*Por ejemplo, al mostrar un cuadro de diálogo con información, es posible que el desarrollador desee **bloquear** la rotación de la pantalla para evitar que se **cierre** el cuadro de diálogo y que se vuelva a **generar** la actividad actual para **desbloquearla** nuevamente cuando se **cierre** el cuadro*

de diálogo.

Aunque podemos lograr un bloqueo de rotación desde el manifiesto haciendo:

```
<activity
    android:name=".TheActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
</activity>
```

Uno puede hacerlo programáticamente también haciendo lo siguiente:

```
public void lockDeviceRotation(boolean value) {
    if (value) {
        int currentOrientation = getResources().getConfiguration().orientation;
        if (currentOrientation == Configuration.ORIENTATION_LANDSCAPE) {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSOR_LANDSCAPE);
        } else {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSOR_PORTRAIT);
        }
    } else {
        getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR2) {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_USER);
        } else {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
        }
    }
}
```

Y luego llamando a lo siguiente, para bloquear y desbloquear respectivamente la rotación del dispositivo

```
lockDeviceRotation(true)
```

y

```
lockDeviceRotation(false)
```

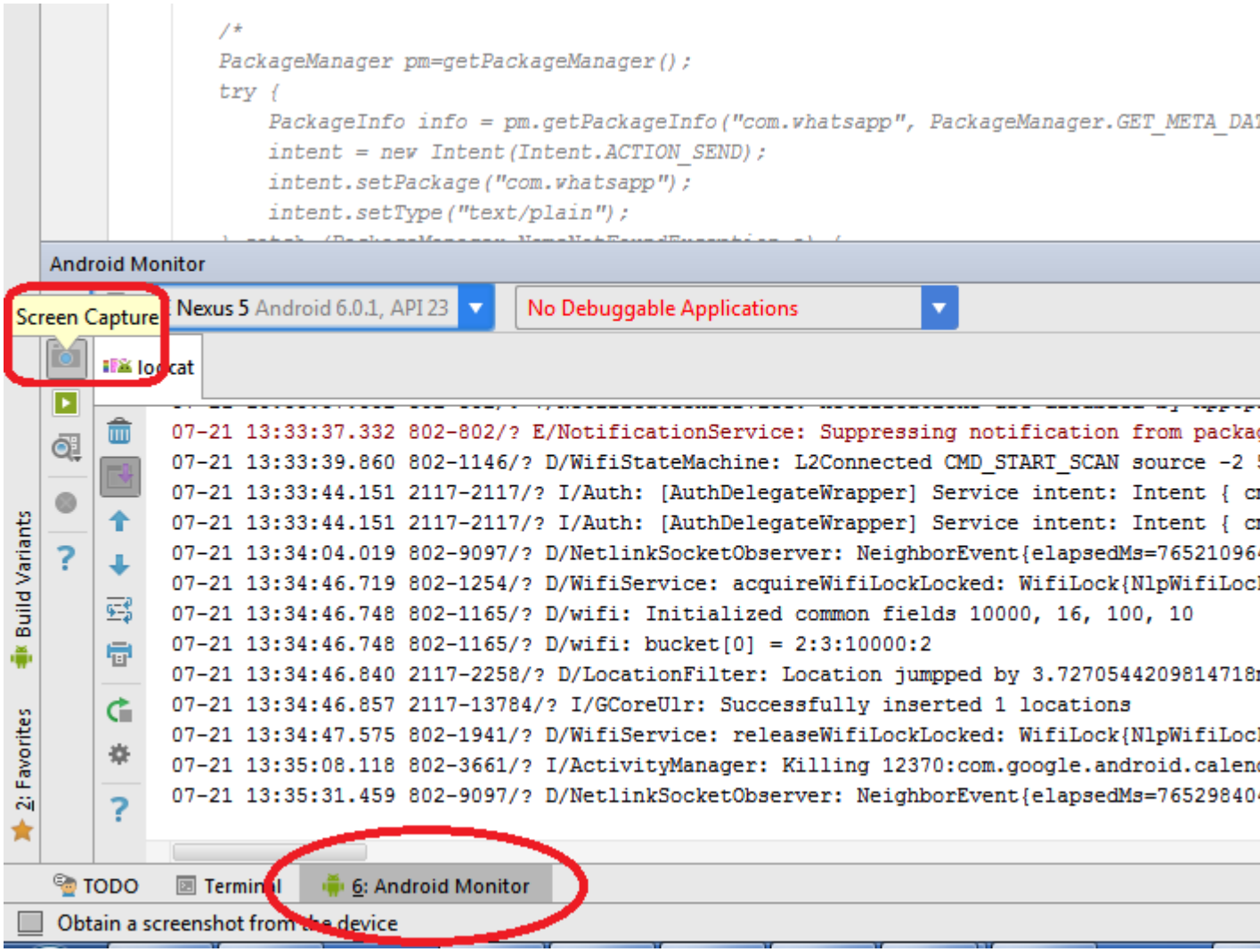
Lea Cambios de orientación en línea: https://riptutorial.com/es/android/topic/4621/cambios-de-orientacion

Capítulo 45: Captura de capturas de pantalla

Examples

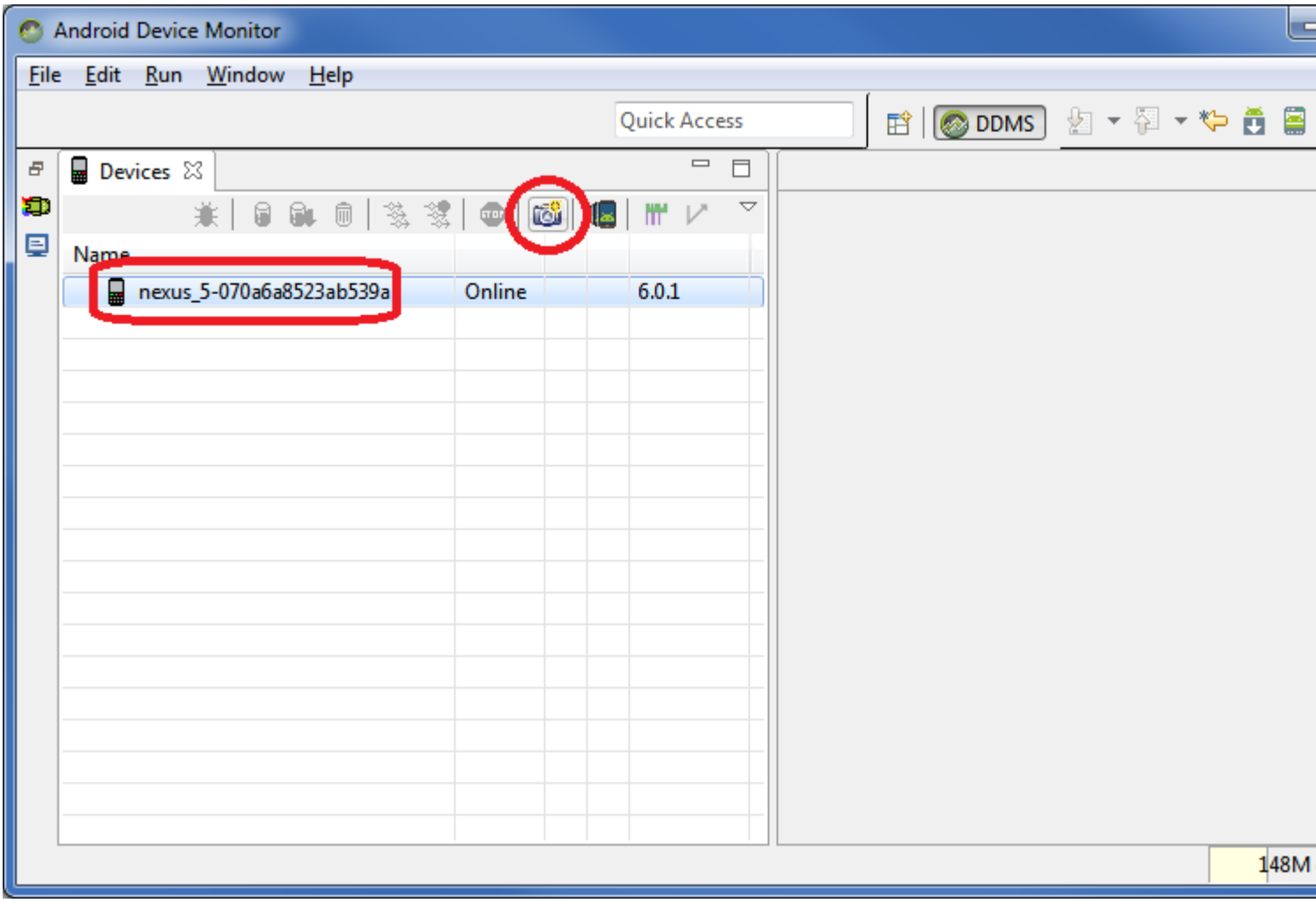
Captura de captura de pantalla a través de Android Studio

1. Abrir la pestaña del monitor de Android
2. Haga clic en el botón de captura de pantalla



Captura de captura de pantalla a través del monitor del dispositivo Android

1. Abra el Monitor de dispositivo Android (*es decir, C: <ANDROID_SDK_LOCATION> \ tools \ monitor.bat*)
2. Seleccione su dispositivo
3. Haga clic en el botón de captura de pantalla



Captura de pantalla de captura a través de ADB

El siguiente ejemplo guarda una captura de pantalla en el almacenamiento interno de los dispositivos.

```
adb shell screencap /sdcard/screen.png
```

Captura de captura de pantalla a través de ADB y guardando directamente en tu PC

Si usa Linux (o Windows con Cygwin), puede ejecutar:

```
adb shell screencap -p | sed 's/\r$//' > screenshot.png
```

Tomando una captura de pantalla de una vista particular

Si desea tomar una captura de pantalla de una Vista `v` particular, puede usar el siguiente código:

```
Bitmap viewBitmap = Bitmap.createBitmap(v.getWidth(), v.getHeight(), Bitmap.Config.RGB_565);
Canvas viewCanvas = new Canvas(viewBitmap);
Drawable backgroundDrawable = v.getBackground();
```

```
if(backgroundDrawable != null){
    // Draw the background onto the canvas.
    backgroundDrawable.draw(viewCanvas);
}
else{
    viewCanvas.drawColor(Color.GREEN);
    // Draw the view onto the canvas.
    v.draw(viewCanvas)
}

// Write the bitmap generated above into a file.
String fileStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
OutputStream outputStream = null;
try{
    imgFile = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES), fileStamp
+ ".png");
    outputStream = new FileOutputStream(imgFile);
    viewBitmap.compress(Bitmap.CompressFormat.PNG, 40, outputStream);
    outputStream.close();
}
catch(Exception e){
    e.printStackTrace();
}
```

Lea Captura de capturas de pantalla en línea:
https://riptutorial.com/es/android/topic/4506/captura-de-capturas-de-pantalla

Capítulo 46: CardView

Introducción

Un FrameLayout con una esquina redondeada de fondo y sombra.

CardView usa la propiedad de elevación en Lollipop para las sombras y recurre a una implementación de sombra emulada personalizada en plataformas más antiguas.

Debido a la naturaleza costosa del recorte de esquinas redondeadas, en plataformas antes de Lollipop, CardView no recorta sus elementos secundarios que se intersecan con esquinas redondeadas. En su lugar, agrega relleno para evitar dicha intersección (consulte setPreventCornerOverlap (booleano) para cambiar este comportamiento).

Parámetros

Parámetro	Detalles
cardBackgroundColor	Color de fondo para CardView.
cardCornerRadius	Radio de esquina para CardView.
TarjetaElevación	Elevación para CardView.
cardMaxElevation	Elevación máxima para CardView.
cardPreventCornerOverlap	Agregue relleno a CardView en v20 y antes para evitar las intersecciones entre el contenido de la Tarjeta y las esquinas redondeadas.
cardUseCompatPadding	Agregue el relleno en API v21 + también para tener las mismas medidas con versiones anteriores. Puede ser un valor booleano, como "verdadero" o "falso".
contentPadding	Acolchado interior entre los bordes de la tarjeta y los hijos de la CardView.
contentPaddingBottom	Acolchado interno entre el borde inferior de la tarjeta y los elementos secundarios de CardView.
contentPaddingLeft	Acolchado interno entre el borde izquierdo de la Tarjeta y los hijos de la CardView.
contentPaddingRight	Elevación para CardView.
TarjetaElevación	Acolchado interno entre el borde derecho de la Tarjeta y los hijos de la CardView.

Parámetro	Detalles
contentPaddingTop	Acolchado interno entre el borde superior de la tarjeta y los hijos de la CardView.

Observaciones

`CardView` utiliza la elevación real y las sombras dinámicas en Lollipop (API 21) y superior. Sin embargo, antes de que Lollipop `CardView` a una implementación instantánea programática.

Si intenta hacer que un `ImageView` encaje dentro de las esquinas redondeadas de un `CardView` , puede notar que no parece correcto antes del Lollipop (API 21). Para solucionar este problema, debe llamar a `setPreventCornerOverlap(false)` en su `CardView` , o agregar la `app:cardPreventCornerOverlap="false"` a su diseño.

Antes de usar `CardView` , debe agregar la dependencia de la biblioteca de soporte en el archivo `build.gradle` :

```
dependencies{
    compile 'com.android.support:cardview-v7:25.2.0'
}
```

Un número de la última versión se puede encontrar aquí

Documentación oficial:

https://developer.android.com/reference/android/support/v7/widget/CardView.html
https://developer.android.com/training/material/lists-cards.html

Examples

Empezando con CardView

`CardView` es miembro de la biblioteca de soporte de Android y proporciona un diseño para tarjetas.

Para agregar `CardView` a su proyecto, agregue la siguiente línea a sus dependencias de `build.gradle` .

```
compile 'com.android.support:cardview-v7:25.1.1'
```

Un número de la última versión se puede encontrar aquí

En su diseño, puede agregar lo siguiente para obtener una tarjeta.

```
<android.support.v7.widget.CardView
        xmlns:card_view="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content">

        <!-- one child layout containing other layouts or views -->

</android.support.v7.widget.CardView>
```

Luego puede agregar otros diseños dentro de este y se incluirán en una tarjeta.

Además, CardView se puede rellenar con cualquier elemento de la interfaz de usuario y se puede manipular desde el código .

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/card_view"
    android:layout_margin="5dp"
    card_view:cardBackgroundColor="#81C784"
    card_view:cardCornerRadius="12dp"
    card_view:cardElevation="3dp"
    card_view:contentPadding="4dp" >

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp" >

        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:id="@+id/item_image"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_marginRight="16dp"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/item_title"
            android:layout_toRightOf="@+id/item_image"
            android:layout_alignParentTop="true"
            android:textSize="30sp"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/item_detail"
            android:layout_toRightOf="@+id/item_image"
            android:layout_below="@+id/item_title"
            />

    </RelativeLayout>
</android.support.v7.widget.CardView>
```

Personalizando el CardView

CardView proporciona una elevación y un radio de esquina predeterminados para que las tarjetas tengan una apariencia uniforme en las plataformas.

Puede personalizar estos valores predeterminados utilizando estos atributos en el archivo xml:

1. `card_view:cardElevation` atributo `card_view:cardElevation` agrega elevación en CardView.
2. `card_view:cardBackgroundColor` atributo `card_view:cardBackgroundColor` se usa para personalizar el color de fondo del fondo de CardView (puedes darle cualquier color).
3. `card_view:cardCornerRadius` atributo `card_view:cardCornerRadius` se usa para curvar 4 bordes de CardView
4. `card_view:contentPadding` atributo `card_view:contentPadding` agrega el relleno entre la tarjeta y los hijos de la tarjeta

Nota: card_view es un espacio de nombres definido en la vista de diseño principal superior. xmlns: card_view = " http://schemas.android.com/apk/res-auto "

Aquí un ejemplo:

```
<android.support.v7.widget.CardView
        xmlns:card_view="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        card_view:cardElevation="4dp"
        card_view:cardBackgroundColor="@android:color/white"
        card_view:cardCornerRadius="8dp"
        card_view:contentPadding="16dp">

        <!-- one child layout containing other layouts or views -->

</android.support.v7.widget.CardView>
```

También puedes hacerlo mediante programación:

```
card.setCardBackgroundColor(....);
card.setCardElevation(...);
card.setRadius(....);
card.setContentPadding();
```

Compruebe el javadoc oficial para propiedades adicionales.

Añadiendo animación de rizo

Para habilitar la animación ripple en un CardView, agregue los siguientes atributos:

```
<android.support.v7.widget.CardView
  ...
  android:clickable="true"
  android:foreground="?android:attr/selectableItemBackground">
  ...
</android.support.v7.widget.CardView>
```

Uso de imágenes como fondo en CardView (problemas con el dispositivo Pre-Lollipop)

Mientras usa Imagen / Color como fondo en un CardView, puede terminar con pequeños rellenos blancos (si el color predeterminado de la Tarjeta es blanco) en los bordes. Esto ocurre debido a las esquinas redondeadas predeterminadas en la Vista de tarjeta. Aquí es cómo evitar esos márgenes en dispositivos Pre-lollipop.

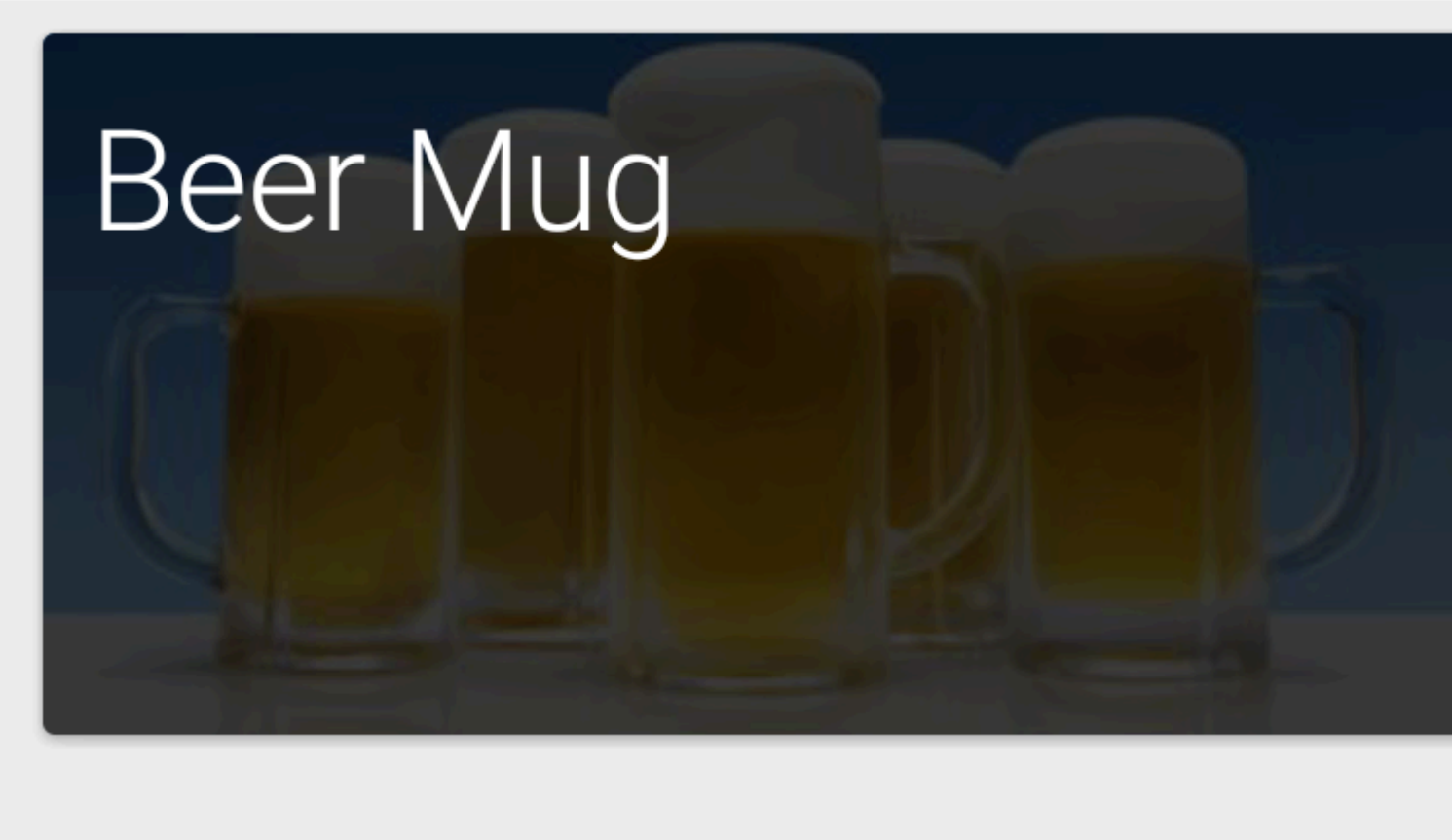
Necesitamos usar un atributo `card_view:cardPreventCornerOverlap="false"` en el CardView. 1). En XML usa el siguiente fragmento de código.

```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    card_view:cardPreventCornerOverlap="false"
    android:layout_height="wrap_content">
      <ImageView
            android:id="@+id/row_wallet_redeem_img"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:adjustViewBounds="true"
            android:scaleType="centerCrop"
            android:src="@drawable/bg_image" />

</android.support.v7.widget.CardView>
```

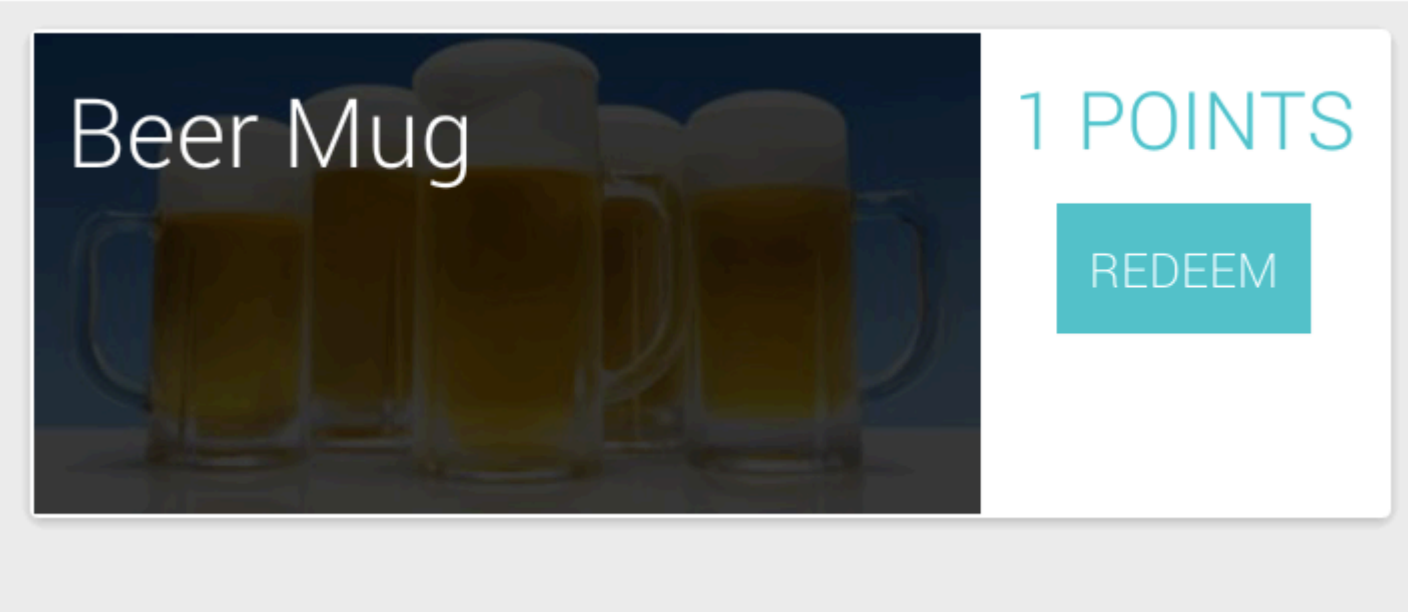
2. En Java como esta `cardView.setPreventCornerOverlap(false)` .

Al hacerlo, elimina un relleno no deseado en los bordes de la Tarjeta. Aquí hay algunos ejemplos visuales relacionados con esta implementación.

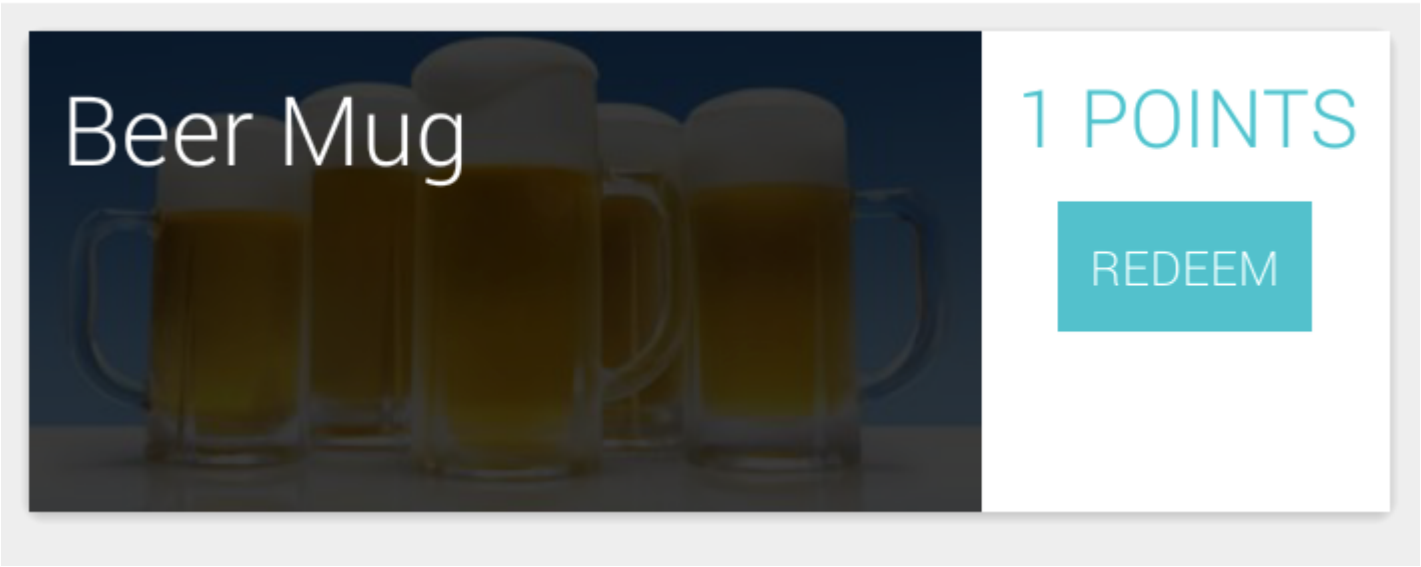
1 tarjeta con fondo de imagen en API 21 (perfectamente bien)



2 Tarjeta con fondo de imagen en API 19 sin atributo (observe los rellenos alrededor de la imagen)



3 Tarjeta FIJA con fondo de imagen en API 19 con atributo
`cardView.setPreventCornerOverlap(false)` (Problema ahora solucionado)



Lea también sobre esto en Documentación aquí
Publicación original de SOF aquí

Animar CardView color de fondo con TransitionDrawable

```
public void setCardColorTran(CardView card) {
    ColorDrawable[] color = {new ColorDrawable(Color.BLUE), new ColorDrawable(Color.RED)};
    TransitionDrawable trans = new TransitionDrawable(color);
    if(Build.VERSION.SDK_INT > Build.VERSION_CODES.ICE_CREAM_SANDWICH_MR1) {
        card.setBackground(trans);
    } else {
        card.setBackgroundDrawable(trans);
    }
    trans.startTransition(5000);
}
```

Lea CardView en línea: https://riptutorial.com/es/android/topic/726/cardview

Capítulo 47: Cargador

Introducción

El cargador es una buena opción para prevenir la pérdida de memoria si desea cargar datos en segundo plano cuando se llama al método oncreate. Por ejemplo, cuando ejecutamos Asynctask en el método oncreate y rotamos la pantalla, la actividad volverá a crear lo que ejecutará otra AsyncTask nuevamente, así que probablemente dos Asyntask se ejecuten en paralelo en lugar de un cargador similar que continuará el proceso en segundo plano que ejecutamos antes.

Parámetros

Clase	Descripción
LoaderManager	Una clase abstracta asociada con una Actividad o Fragmento para administrar una o más instancias de Loader.
LoaderManager.LoaderCallbacks	Una interfaz de devolución de llamada para que un cliente interactúe con el LoaderManager.
Cargador	Una clase abstracta que realiza la carga asíncrona de datos.
AsyncTaskLoader	Cargador abstracto que proporciona una AsyncTask para hacer el trabajo.
CursorLoader	Una subclase de AsyncTaskLoader que consulta el ContentResolver y devuelve un cursor.

Observaciones

Introducidos en Android 3.0, los cargadores facilitan la carga asincrónica de datos en una actividad o fragmento. Los cargadores tienen estas características:

- Están disponibles para cada actividad y fragmento .
- Proporcionan carga asíncrona de datos.
- Supervisan la fuente de sus datos y entregan nuevos resultados cuando cambia el contenido.
- Se vuelven a conectar automáticamente al cursor del último cargador cuando se recrean después de un cambio de configuración. Por lo tanto, no necesitan volver a consultar sus datos.

Cuando no usar cargadores

No debe usar los cargadores si necesita completar las tareas en segundo plano. Android destruye los cargadores junto con las actividades / fragmentos a los que pertenecen. Si desea realizar algunas tareas, que deben ejecutarse hasta su finalización, no use los cargadores. Deberías usar servicios para este tipo de cosas en su lugar.

Examples

AsyncTaskLoader básico

`AsyncTaskLoader` es un `Loader` abstracto que proporciona una `AsyncTask` para realizar el trabajo.

Aquí algunas implementaciones básicas:

```
final class BasicLoader extends AsyncTaskLoader<String> {

    public BasicLoader(Context context) {
        super(context);
    }

    @Override
    public String loadInBackground() {
        // Some work, e.g. load something from internet
        return "OK";
    }

    @Override
    public void deliverResult(String data) {
        if (isStarted()) {
            // Deliver result if loader is currently started
            super.deliverResult(data);
        }
    }

    @Override
    protected void onStartLoading() {
        // Start loading
        forceLoad();
    }

    @Override
    protected void onStopLoading() {
        cancelLoad();
    }

    @Override
    protected void onReset() {
        super.onReset();

        // Ensure the loader is stopped
        onStopLoading();
    }
}
```

Normalmente, `Loader` se inicializa dentro del método `onCreate()` la actividad, o dentro del `onActivityCreated()` del fragmento. También usualmente la actividad o el fragmento implementa la interfaz `LoaderManager.LoaderCallbacks` :

```
public class MainActivity extends Activity implements LoaderManager.LoaderCallbacks<String> {

    // Unique id for loader
    private static final int LDR_BASIC_ID = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Initialize loader; Some data can be passed as second param instead of Bundle.Empty
        getLoaderManager().initLoader(LDR_BASIC_ID, Bundle.EMPTY, this);
    }

    @Override
    public Loader<String> onCreateLoader(int id, Bundle args) {
        return new BasicLoader(this);
    }

    @Override
    public void onLoadFinished(Loader<String> loader, String data) {
        Toast.makeText(this, data, Toast.LENGTH_LONG).show();
    }

    @Override
    public void onLoaderReset(Loader<String> loader) {
    }
}
```

En este ejemplo, cuando se complete el cargador, se mostrará una tostada con el resultado.

AsyncTaskLoader con caché

Es una buena práctica almacenar en caché el resultado cargado para evitar la carga múltiple de los mismos datos.
Para invalidar la memoria caché, se debe llamar a `onContentChanged()` . Si el cargador ya se ha iniciado, se `forceLoad()` , de lo contrario (si el cargador está en estado detenido) el cargador podrá comprender el cambio de contenido con la `takeContentChanged()` .

Nota: se debe llamar a `onContentChanged()` desde el hilo principal del proceso.

Javadocs dice acerca de takeContentChanged ():

> Tome la marca actual que indica si el contenido del cargador ha cambiado mientras se detuvo. Si es así, devuelve true y se borra la bandera.

```
public abstract class BaseLoader<T> extends AsyncTaskLoader<T> {

    // Cached result saved here
    private final AtomicReference<T> cache = new AtomicReference<>();
```

```
    public BaseLoader(@NonNull final Context context) {
        super(context);
    }

    @Override
    public final void deliverResult(final T data) {
        if (!isReset()) {
            // Save loaded result
            cache.set(data);
            if (isStarted()) {
                super.deliverResult(data);
            }
        }
    }

    @Override
    protected final void onStartLoading() {
        // Register observers
        registerObserver();

        final T cached = cache.get();
        // Start new loading if content changed in background
        // or if we never loaded any data
        if (takeContentChanged() || cached == null) {
            forceLoad();
        } else {
            deliverResult(cached);
        }
    }

    @Override
    public final void onStopLoading() {
        cancelLoad();
    }

    @Override
    protected final void onReset() {
        super.onReset();
        onStopLoading();
        // Clear cache and remove observers
        cache.set(null);
        unregisterObserver();
    }

    /* virtual */
    protected void registerObserver() {
        // Register observers here, call onContentChanged() to invalidate cache
    }

    /* virtual */
    protected void unregisterObserver() {
        // Remove observers
    }
}
```

Recarga

Para invalidar sus datos antiguos y reiniciar el cargador existente, puede usar el método `restartLoader()` :

```
private void reload() {
    getLoaderManager().reastartLoader(LOADER_ID, Bundle.EMPTY, this);
}
```

Pasar parámetros utilizando un paquete

Puedes pasar los parámetros por Bundle:

```
Bundle myBundle = new Bundle();
myBundle.putString(MY_KEY, myValue);
```

Obtenga el valor en onCreateLoader:

```
@Override
public Loader<String> onCreateLoader(int id, final Bundle args) {
    final String myParam = args.getString(MY_KEY);
    ...
}
```

Lea Cargador en línea: https://riptutorial.com/es/android/topic/4390/cargador

Capítulo 48: Cargador de Imagen Universal

Observaciones

Ejemplos URI aceptables:

```
"http://www.example.com/image.png" // from Web
"file:///mnt/sdcard/image.png" // from SD card
"file:///mnt/sdcard/video.mp4" // from SD card (video thumbnail)
"content://media/external/images/media/13" // from content provider
"content://media/external/video/media/13" // from content provider (video thumbnail)
"assets://image.png" // from assets
"drawable://" + R.drawable.img // from drawables (non-9patch images)
```

Examples

Inicializar Universal Image Loader

1. Agregue la siguiente dependencia al archivo *build.gradle* :

   ```
   compile 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
   ```

2. Agregue los siguientes permisos al archivo *AndroidManifest.xml* :

   ```
   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   ```

3. Inicialice el cargador de imagen universal. Esto debe hacerse antes del primer uso:

   ```
   ImageLoaderConfiguration config = new ImageLoaderConfiguration.Builder(this)
       // ...
       .build();
   ImageLoader.getInstance().init(config);
   ```

   Las opciones de configuración completas se pueden encontrar aquí .

Uso básico

1. Cargue una imagen, decodifíquela en un mapa de bits y visualice el mapa de bits en un `ImageView` (o en cualquier otra vista que implemente la interfaz de `ImageAware`):

   ```
   ImageLoader.getInstance().displayImage(imageUri, imageView);
   ```

2. Cargue una imagen, decodifíquela en un mapa de bits y devuelva el mapa de bits a una devolución de llamada:

```
ImageLoader.getInstance().loadImage(imageUri, new SimpleImageLoadingListener() {
    @Override
    public void onLoadingComplete(String imageUri, View view, Bitmap loadedImage) {
        // Do whatever you want with the bitmap.
    }
});
```

3. Cargue una imagen, decodifíquela en un mapa de bits y devuelva el mapa de bits de forma sincrónica:

```
Bitmap bmp = ImageLoader.getInstance().loadImageSync(imageUri);
```

Lea Cargador de Imagen Universal en línea:
https://riptutorial.com/es/android/topic/2760/cargador-de-imagen-universal

Capítulo 49: Cargando Bitmaps Efectivamente

Introducción

Este tema se concentra principalmente en cargar los mapas de bits de manera efectiva en dispositivos Android.

Cuando se trata de cargar un mapa de bits, la pregunta viene de dónde se carga. Aquí vamos a discutir sobre cómo cargar el mapa de bits desde el recurso en el dispositivo Android. por ejemplo, desde la galería.

Vamos a pasar por esto con el ejemplo que se discute a continuación.

Sintaxis

- `<uses-permission>` **->** Etiqueta utilizada para el permiso.
- `android:name` **->** Un atributo usado para dar nombre al permiso que vamos a solicitar.
- `android.permission.READ_EXTERNAL_STORAGE` **->** Es permisos del sistema
- ejemplo "android.permission.CAMERA" o "android.permission.READ_CONTACTS"

Examples

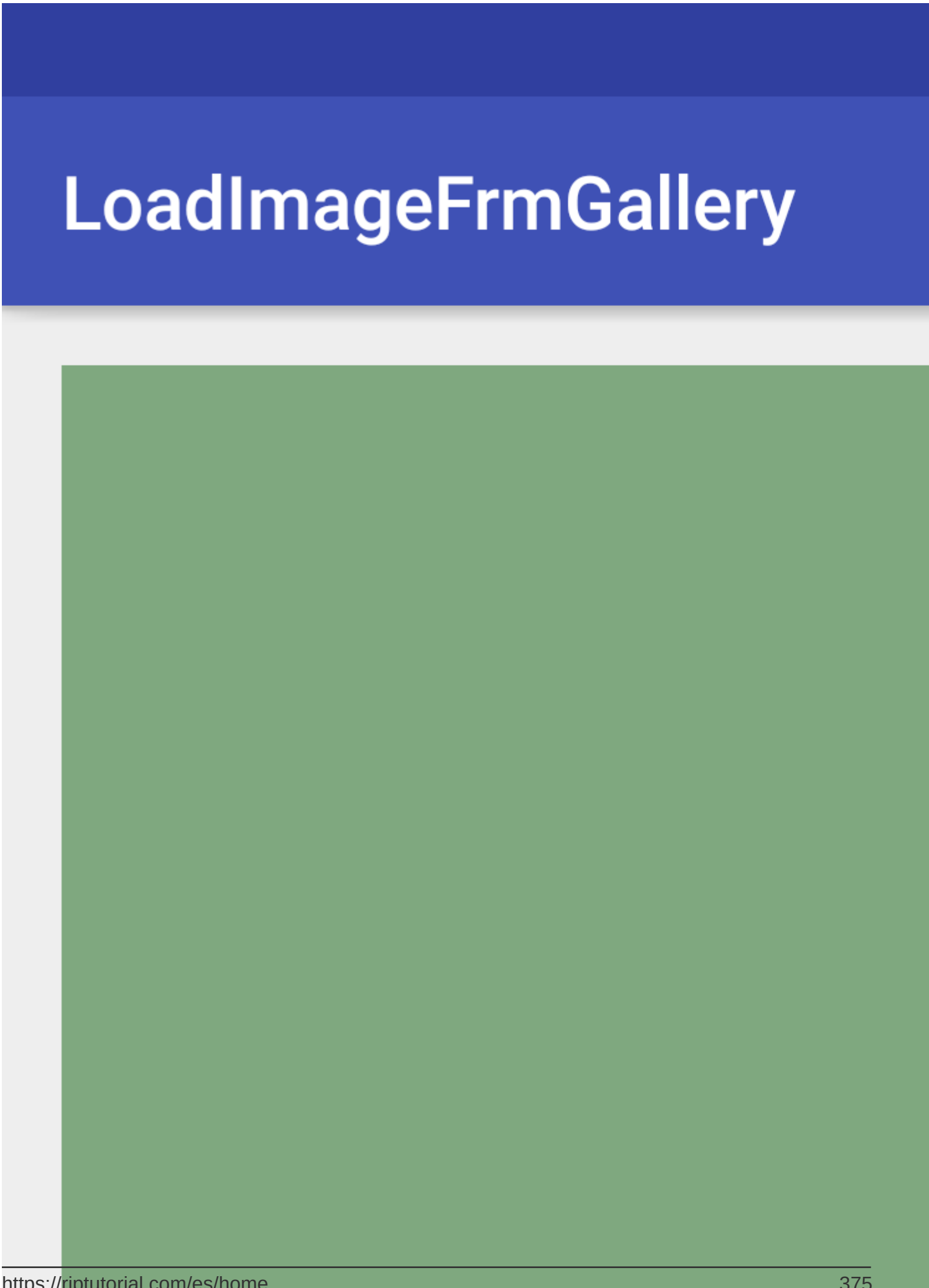
Cargue la imagen desde el recurso desde el dispositivo Android. Usando intenciones.

Usando intenciones para cargar la imagen desde la galería.

1. Inicialmente necesitas tener el permiso

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

2. Utilice el siguiente Código para tener el diseño tal como está diseñado a continuación.


LoadImageFrmGallery

https://riptutorial.com/es/android/topic/10902/cargando-bitmaps-efectivamente

Capítulo 50: carril rápido

Observaciones

fastlane es una herramienta para desarrolladores de iOS, Mac y Android para automatizar tareas tediosas como generar capturas de pantalla, tratar con perfiles de aprovisionamiento y lanzar su aplicación.

Docs: https://docs.fastlane.tools/

Código fuente: https://github.com/fastlane/fastlane

Examples

Archivo rápido para crear y cargar múltiples versiones a Beta by Crashlytics

Este es un ejemplo de configuración de **Fastfile** para una aplicación de múltiples sabores. Te da la opción de crear e implementar todos los sabores o un solo sabor. Después de la implementación, informa a **Slack** el estado de la implementación y envía una notificación a los evaluadores en Beta por el grupo de evaluadores de Crashlytics.

Para construir y desplegar todos los sabores usa:

```
fastlane android beta
```

Para construir un solo APK y desplegar uso:

```
fastlane android beta app:flavorName
```

Con un solo archivo Fastlane, puede administrar aplicaciones iOS, Android y Mac. Si está utilizando este archivo solo para una `platform` aplicaciones no es necesario.

Cómo funciona

1. `android` argumento de `android` le dice a Fastlane que usaremos `:android` plataforma `:android` .
2. En el interior `:android` plataforma de `:android` puede tener varios carriles. Actualmente solo tengo `:beta` lane. El segundo argumento del comando anterior especifica el carril que queremos usar.
3. `options[:app]`
4. Hay dos tareas de **Gradle** . Primero, corre `gradle clean` . Si proporcionó un sabor con la clave de la `app` , fastfile ejecuta `gradle assembleReleaseFlavor` . De lo contrario, ejecuta `gradle assembleRelease` para compilar todos los sabores de compilación.
5. Si estamos construyendo para todos los tipos, una matriz de nombres de archivos APK generados se almacena en `SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS` . Usamos esto para recorrer los archivos generados y desplegarlos en **Beta por Crashlytics** . `notifications`

campos de `notifications` y `groups` son opcionales. Se utilizan para notificar a los probadores registrados para la aplicación en **Beta por Crashlytics** .

6. Si está familiarizado con Crashlytics, puede que sepa que para activar una aplicación en el portal, debe ejecutarla en un dispositivo y usarla primero. De lo contrario, Crashlytics asumirá la aplicación inactiva y lanzará un error. En este escenario, lo capturo e **informo** a **Slack** como un error, por lo que sabrá qué aplicación está inactiva.
7. Si la implementación es exitosa, **fastlane** enviará un mensaje de éxito a **Slack** .
8. `#{/([^\/]*)$/.match(apk)}` esta expresión regular se usa para obtener el nombre del sabor de la ruta APK. Puede eliminarlo si no funciona para usted.
9. `get_version_name` y `get_version_code` son dos complementos de **Fastlane** para recuperar el nombre y el código de la versión de la aplicación. Tienes que instalar estas gemas si quieres usarlas, o puedes eliminarlas. Lea más acerca de los complementos aquí.
10. La instrucción `else` se ejecutará si está creando y desplegando un único APK. No tenemos que proporcionar `apk_path` a Crashlytics ya que solo tenemos una aplicación.
11. `error do` block al final se usa para recibir notificaciones si algo sale mal durante la ejecución.

Nota

No olvide reemplazar `SLACK_URL` , `API_TOKEN` , `GROUP_NAME` y `BUILD_SECRET` con sus propias credenciales.

```
fastlane_version "1.46.1"

default_platform :android

platform :android do

    before_all do
        ENV["SLACK_URL"] = "https://hooks.slack.com/servic...."
    end

    lane :beta do |options|
        # Clean and build the Release version of the app.
        # Usage `fastlane android beta app:flavorName`

        gradle(task: "clean")

        gradle(task: "assemble",
               build_type: "Release",
               flavor: options[:app])

        # If user calls `fastlane android beta` command, it will build all projects and push
them to Crashlytics
        if options[:app].nil?
            lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].each do | apk |

                puts "Uploading APK to Crashlytics: " + apk

                begin
                    crashlytics(
                      api_token: "[API_TOKEN]",
                      build_secret: "[BUILD_SECRET]",
                      groups: "[GROUP_NAME]",
                      apk_path: apk,
                      notifications: "true"
```

```
                    )

                    slack(
                      message: "Successfully deployed new build for #{/([^\/]*)$/.match(apk)}
#{get_version_name} - #{get_version_code}",
                      success: true,
                      default_payloads: [:git_branch, :lane, :test_result]
                    )
                rescue => ex
                    # If the app is inactive in Crashlytics, deployment will fail. Handle it
here and report to slack
                    slack(
                        message: "Error uploading => #{/([^\/]*)$/.match(apk)}
#{get_version_name} - #{get_version_code}: #{ex}",
                        success: false,
                        default_payloads: [:git_branch, :lane, :test_result]
                    )
                end
            end

            after_all do |lane|
                # This block is called, only if the executed lane was successful
                slack(
                    message: "Operation completed for
#{lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].size} app(s) for #{get_version_name}
- #{get_version_code}",
                    default_payloads: [:git_branch, :lane, :test_result],
                    success: true
                )
            end
        else
            # Single APK upload to Beta by Crashlytics
            crashlytics(
                api_token: "[API_TOKEN]",
                build_secret: "[BUILD_SECRET]",
                groups: "[GROUP_NAME]",
                notifications: "true"
            )

            after_all do |lane|
                # This block is called, only if the executed lane was successful
                slack(
                    message: "Successfully deployed new build for #{options[:app]}
#{get_version_name} - #{get_version_code}",
                    default_payloads: [:git_branch, :lane, :test_result],
                    success: true
                )
            end
        end

        error do |lane, exception|
            slack(
                message: exception.message,
                success: false,
                default_payloads: [:git_branch, :lane, :test_result]
            )
        end
    end
end
```

Fastfile lane para crear e instalar todos los sabores para un tipo de compilación dado en un dispositivo

Agregue este carril a su **archivo Fastfile** y ejecute `fastlane installAll type:{BUILD_TYPE}` en la línea de comandos. Reemplace `BUILD_TYPE` con el tipo de compilación que desea compilar.

Por ejemplo: `fastlane installAll type:Debug`

Este comando construirá todos los sabores del tipo dado y lo instalará en su dispositivo. Actualmente, no funciona si tiene más de un dispositivo conectado. Asegúrate de tener solo uno. En el futuro, estoy planeando agregar una opción para seleccionar el dispositivo de destino.

```
lane :installAll do |options|

    gradle(task: "clean")

    gradle(task: "assemble",
       build_type: options[:type])

    lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].each do | apk |

        puts "Uploading APK to Device: " + apk

        begin
            adb(
                command: "install -r #{apk}"
            )
        rescue => ex
            puts ex
        end
    end
end
```

Lea carril rápido en línea: https://riptutorial.com/es/android/topic/8215/carril-rapido

Capítulo 51: Ciclo de vida de la interfaz de usuario

Examples

Guardar datos en el recorte de memoria

```
public class ExampleActivity extends Activity {

    private final static String EXAMPLE_ARG = "example_arg";
    private int mArg;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_example);

        if(savedInstanceState != null) {
            mArg = savedInstanceState.getInt(EXAMPLE_ARG);
        }
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putInt(EXAMPLE_ARG, mArg);
    }
}
```

Explicación

¿Entonces, Que esta pasando aquí?

El sistema Android siempre se esforzará por borrar la mayor cantidad de memoria posible. Por lo tanto, si su actividad se reduce a un segundo plano, y otra actividad de primer plano exige su participación, el sistema Android llamará a `onTrimMemory()` sobre su actividad.

Pero eso no significa que todas tus propiedades deban desaparecer. Lo que debes hacer es guardarlos en un objeto Bundle. El objeto del paquete se maneja mucho mejor en cuanto a memoria. Dentro de un paquete, cada objeto se identifica mediante una secuencia de texto única: en el ejemplo anterior, la variable de valor entero `mArg` se mantiene bajo el nombre de referencia `EXAMPLE_ARG` . Y cuando se recrea la actividad, extraiga sus valores antiguos del objeto Bundle en lugar de recrearlos desde cero

Lea Ciclo de vida de la interfaz de usuario en línea:
https://riptutorial.com/es/android/topic/3440/ciclo-de-vida-de-la-interfaz-de-usuario

Capítulo 52: Cifrado / descifrado de datos

Introducción

En este tema se explica cómo funciona el cifrado y el descifrado en Android.

Examples

Cifrado AES de datos mediante contraseña de forma segura.

El siguiente ejemplo encripta un bloque de datos dado usando AES . La clave de cifrado se obtiene de forma segura (sal aleatoria, 1000 rondas de SHA-256). El cifrado utiliza AES en modo CBC con IV aleatorio.

Tenga en cuenta que los datos almacenados en la clase `EncryptedData` (`salt` , `iv` y `encryptedData`) se pueden concatenar en una matriz de un solo byte. A continuación, puede guardar los datos o transmitirlos al destinatario.

```
private static final int SALT_BYTES = 8;
private static final int PBK_ITERATIONS = 1000;
private static final String ENCRYPTION_ALGORITHM = "AES/CBC/PKCS5Padding";
private static final String PBE_ALGORITHM = "PBEwithSHA256and128BITAES-CBC-BC";

private EncryptedData encrypt(String password, byte[] data) throws NoSuchPaddingException,
NoSuchAlgorithmException, InvalidKeySpecException, InvalidKeyException, BadPaddingException,
IllegalBlockSizeException, InvalidAlgorithmParameterException {
    EncryptedData encData = new EncryptedData();
    SecureRandom rnd = new SecureRandom();
    encData.salt = new byte[SALT_BYTES];
    encData.iv = new byte[16]; // AES block size
    rnd.nextBytes(encData.salt);
    rnd.nextBytes(encData.iv);

    PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray(), encData.salt, PBK_ITERATIONS);
    SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance(PBE_ALGORITHM);
    Key key = secretKeyFactory.generateSecret(keySpec);
    Cipher cipher = Cipher.getInstance(ENCRYPTION_ALGORITHM);
    IvParameterSpec ivSpec = new IvParameterSpec(encData.iv);
    cipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
    encData.encryptedData = cipher.doFinal(data);
    return encData;
}

private byte[] decrypt(String password, byte[] salt, byte[] iv, byte[] encryptedData) throws
NoSuchAlgorithmException, InvalidKeySpecException, NoSuchPaddingException,
InvalidKeyException, BadPaddingException, IllegalBlockSizeException,
InvalidAlgorithmParameterException {
    PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray(), salt, PBK_ITERATIONS);
    SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance(PBE_ALGORITHM);
    Key key = secretKeyFactory.generateSecret(keySpec);
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    cipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
```

```
    return cipher.doFinal(encryptedData);
}

private static class EncryptedData {
    public byte[] salt;
    public byte[] iv;
    public byte[] encryptedData;
}
```

El siguiente código de ejemplo muestra cómo probar el cifrado y el descifrado:

```
try {
    String password = "test12345";
    byte[] data = "plaintext11223344556677889900".getBytes("UTF-8");
    EncryptedData encData = encrypt(password, data);
    byte[] decryptedData = decrypt(password, encData.salt, encData.iv, encData.encryptedData);
    String decDataAsString = new String(decryptedData, "UTF-8");
    Toast.makeText(this, decDataAsString, Toast.LENGTH_LONG).show();
} catch (Exception e) {
    e.printStackTrace();
}
```

Lea Cifrado / descifrado de datos en línea: https://riptutorial.com/es/android/topic/3471/cifrado---descifrado-de-datos

Capítulo 53: CleverTap

Introducción

Hacks rápidos para el SDK de análisis y compromiso proporcionado por CleverTap - Android

Observaciones

Obtenga sus credenciales de CleverTap en https://clevertap.com .

Examples

Obtener una instancia del SDK para grabar eventos

```
CleverTapAPI cleverTap;
try {
  cleverTap = CleverTapAPI.getInstance(getApplicationContext());
} catch (CleverTapMetaDataNotFoundException e) {
  // thrown if you haven't specified your CleverTap Account ID or Token in your
AndroidManifest.xml
} catch (CleverTapPermissionsNotSatisfied e) {
  // thrown if you haven’t requested the required permissions in your AndroidManifest.xml
}
```

Configuración del nivel de depuración

En su clase de aplicación personalizada, anule el método `onCreate()` , agregue la línea a continuación:

```
CleverTapAPI.setDebugLevel(1);
```

Lea CleverTap en línea: https://riptutorial.com/es/android/topic/9337/clevertap

Capítulo 54: Colores

Examples

Manipulación de color

Para manipular los colores, modificaremos los valores argb (Alfa, Rojo, Verde y Azul) de un color.

Primero extrae los valores RGB de tu color.

```
int yourColor = Color.parse("#ae1f67");

int red = Color.red(yourColor);
int green = Color.green(yourColor);
int blue = Color.blue(yourColor);
```

Ahora puede reducir o aumentar los valores de rojo, verde y azul y combinarlos para volver a ser un color:

```
 int newColor = Color.rgb(red, green, blue);
```

O si desea agregarle algo de alfa, puede agregarlo mientras crea el color:

```
 int newColor = Color.argb(alpha, red, green, blue);
```

Los valores alfa y RGB deben estar en el rango [0-225].

Lea Colores en línea: https://riptutorial.com/es/android/topic/4986/colores

Capítulo 55: Comenzando con OpenGL ES 2.0+

Introducción

Este tema trata sobre la configuración y el uso de **OpenGL ES 2.0+** en Android. OpenGL ES es el estándar para gráficos acelerados 2D y 3D en sistemas integrados, incluidas consolas, teléfonos inteligentes, dispositivos y vehículos.

Examples

Configurando GLSurfaceView y OpenGL ES 2.0+

Para usar OpenGL ES en su aplicación, debe agregar esto al manifiesto:

```
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
```

Cree su GLSurfaceView extendido:

```
import static android.opengl.GLES20.*; // To use all OpenGL ES 2.0 methods and constants
statically

public class MyGLSurfaceView extends GLSurfaceView {

    public MyGLSurfaceView(Context context, AttributeSet attrs) {
        super(context, attrs);

        setEGLContextClientVersion(2); // OpenGL ES version 2.0
        setRenderer(new MyRenderer());
        setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
    }

    public final class MyRenderer implements GLSurfaceView.Renderer{
        public final void onSurfaceCreated(GL10 unused, EGLConfig config) {
            // Your OpenGL ES init methods
            glClearColor(1f, 0f, 0f, 1f);
        }
        public final void onSurfaceChanged(GL10 unused, int width, int height) {
            glViewport(0, 0, width, height);
        }

        public final void onDrawFrame(GL10 unused) {
            // Your OpenGL ES draw methods
            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        }
    }
}
```

Agregue `MyGLSurfaceView` a su diseño:

```
<com.example.app.MyGLSurfaceView
    android:id="@+id/gles_renderer"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Para usar la versión más reciente de OpenGL ES, simplemente cambie el número de versión en su manifiesto, en la importación estática y cambie `setEGLContextClientVersion` .

Compilación y vinculación de sombreadores GLSL-ES desde un archivo de activos

La carpeta de Activos es el lugar más común para almacenar sus archivos de sombreado GLSL-ES. Para usarlos en su aplicación OpenGL ES, debe cargarlos en una cadena en primer lugar. Esta función crea una cadena desde el archivo de activos:

```
private String loadStringFromAssetFile(Context myContext, String filePath){
    StringBuilder shaderSource = new StringBuilder();
    try {
        BufferedReader reader = new BufferedReader(new
InputStreamReader(myContext.getAssets().open(filePath)));
        String line;
        while((line = reader.readLine()) != null){
            shaderSource.append(line).append("\n");
        }
        reader.close();
        return shaderSource.toString();
    } catch (IOException e) {
        e.printStackTrace();
        Log.e(TAG, "Could not load shader file");
        return null;
    }
}
```

Ahora necesita crear una función que compile un sombreado almacenado en una picadura:

```
private int compileShader(int shader_type, String shaderString){

    // This compiles the shader from the string
    int shader = glCreateShader(shader_type);
    glShaderSource(shader, shaderString);
    glCompileShader(shader);

    // This checks for for compilation errors
    int[] compiled = new int[1];
    glGetShaderiv(shader, GL_COMPILE_STATUS, compiled, 0);
    if (compiled[0] == 0) {
        String log = glGetShaderInfoLog(shader);

        Log.e(TAG, "Shader compilation error: ");
        Log.e(TAG, log);
    }
    return shader;
}
```

Ahora puedes cargar, compilar y enlazar tus shaders:

```
// Load shaders from file
String vertexShaderString = loadStringFromAssetFile(context, "your_vertex_shader.glsl");
String fragmentShaderString = loadStringFromAssetFile(context, "your_fragment_shader.glsl");

// Compile shaders
int vertexShader = compileShader(GL_VERTEX_SHADER, vertexShaderString);
int fragmentShader = compileShader(GL_FRAGMENT_SHADER, fragmentShaderString);

// Link shaders and create shader program
int shaderProgram = glCreateProgram();
glAttachShader(shaderProgram , vertexShader);
glAttachShader(shaderProgram , fragmentShader);
glLinkProgram(shaderProgram);

// Check for linking errors:
int linkStatus[] = new int[1];
glGetProgramiv(shaderProgram, GL_LINK_STATUS, linkStatus, 0);
if (linkStatus[0] != GL_TRUE) {
    String log = glGetProgramInfoLog(shaderProgram);

    Log.e(TAG,"Could not link shader program: ");
    Log.e(TAG, log);
}
```

Si no hay errores, su programa de sombreado está listo para usar:

```
glUseProgram(shaderProgram);
```

Lea Comenzando con OpenGL ES 2.0+ en línea:
https://riptutorial.com/es/android/topic/8662/comenzando-con-opengl-es-2-0plus

Capítulo 56: Cómo almacenar contraseñas de forma segura

Examples

Usando AES para el cifrado de contraseña salada

Este ejemplo utiliza el algoritmo AES para cifrar contraseñas. La longitud de la sal puede ser de hasta 128 bits.

Estamos utilizando la clase `SecureRandom` para generar un salt, que se combina con la contraseña para generar una clave secreta. Las clases utilizadas ya existen en los paquetes de Android `javax.crypto` y `java.security` .

Una vez que se genera una clave, debemos conservar esta clave en una variable o almacenarla. Lo almacenamos entre las preferencias compartidas en el valor `S_KEY` . Luego, una contraseña se cifra utilizando el método `doFinal` de la clase `Cipher` una vez que se inicializa en `ENCRYPT_MODE` . A continuación, la contraseña cifrada se convierte de una matriz de bytes a una cadena y se almacena entre las preferencias compartidas. La clave utilizada para generar una contraseña encriptada se puede usar para descifrar la contraseña de una manera similar:

```
public class MainActivity extends AppCompatActivity {
    public static final String PROVIDER = "BC";
    public static final int SALT_LENGTH = 20;
    public static final int IV_LENGTH = 16;
    public static final int PBE_ITERATION_COUNT = 100;

    private static final String RANDOM_ALGORITHM = "SHA1PRNG";
    private static final String HASH_ALGORITHM = "SHA-512";
    private static final String PBE_ALGORITHM = "PBEWithSHA256And256BitAES-CBC-BC";
    private static final String CIPHER_ALGORITHM = "AES/CBC/PKCS5Padding";
    public static final String SECRET_KEY_ALGORITHM = "AES";
    private static final String TAG = "EncryptionPassword";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String originalPassword = "ThisIsAndroidStudio%$";
        Log.e(TAG, "originalPassword => " + originalPassword);
        String encryptedPassword = encryptAndStorePassword(originalPassword);
        Log.e(TAG, "encryptedPassword => " + encryptedPassword);
        String decryptedPassword = decryptAndGetPassword();
        Log.e(TAG, "decryptedPassword => " + decryptedPassword);
    }

    private String decryptAndGetPassword() {
        SharedPreferences prefs = getSharedPreferences("pswd", MODE_PRIVATE);
        String encryptedPasswrd = prefs.getString("token", "");
        String passwrd = "";
        if (encryptedPasswrd!=null && !encryptedPasswrd.isEmpty()) {
            try {
```

```
                String output = prefs.getString("S_KEY", "");
                byte[] encoded = hexStringToByteArray(output);
                SecretKey aesKey = new SecretKeySpec(encoded, SECRET_KEY_ALGORITHM);
                passwrd = decrypt(aesKey, encryptedPasswrd);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return passwrd;
    }

    public String encryptAndStorePassword(String password) {
        SharedPreferences.Editor editor = getSharedPreferences("pswd", MODE_PRIVATE).edit();
        String encryptedPassword = "";
        if (password!=null && !password.isEmpty()) {
            SecretKey secretKey = null;
            try {
                secretKey = getSecretKey(password, generateSalt());

                byte[] encoded = secretKey.getEncoded();
                String input = byteArrayToHexString(encoded);
                editor.putString("S_KEY", input);
                encryptedPassword = encrypt(secretKey, password);
            } catch (Exception e) {
                e.printStackTrace();
            }
            editor.putString("token", encryptedPassword);
            editor.commit();
        }
        return encryptedPassword;
    }

    public static String encrypt(SecretKey secret, String cleartext) throws Exception {
        try {
            byte[] iv = generateIv();
            String ivHex = byteArrayToHexString(iv);
            IvParameterSpec ivspec = new IvParameterSpec(iv);

            Cipher encryptionCipher = Cipher.getInstance(CIPHER_ALGORITHM, PROVIDER);
            encryptionCipher.init(Cipher.ENCRYPT_MODE, secret, ivspec);
            byte[] encryptedText = encryptionCipher.doFinal(cleartext.getBytes("UTF-8"));
            String encryptedHex = byteArrayToHexString(encryptedText);

            return ivHex + encryptedHex;

        } catch (Exception e) {
            Log.e("SecurityException", e.getCause().getLocalizedMessage());
            throw new Exception("Unable to encrypt", e);
        }
    }

    public static String decrypt(SecretKey secret, String encrypted) throws Exception {
        try {
            Cipher decryptionCipher = Cipher.getInstance(CIPHER_ALGORITHM, PROVIDER);
            String ivHex = encrypted.substring(0, IV_LENGTH * 2);
            String encryptedHex = encrypted.substring(IV_LENGTH * 2);
            IvParameterSpec ivspec = new IvParameterSpec(hexStringToByteArray(ivHex));
            decryptionCipher.init(Cipher.DECRYPT_MODE, secret, ivspec);
            byte[] decryptedText =
decryptionCipher.doFinal(hexStringToByteArray(encryptedHex));
            String decrypted = new String(decryptedText, "UTF-8");
```

```
            return decrypted;
        } catch (Exception e) {
            Log.e("SecurityException", e.getCause().getLocalizedMessage());
            throw new Exception("Unable to decrypt", e);
        }
    }

    public static String generateSalt() throws Exception {
        try {
            SecureRandom random = SecureRandom.getInstance(RANDOM_ALGORITHM);
            byte[] salt = new byte[SALT_LENGTH];
            random.nextBytes(salt);
            String saltHex = byteArrayToHexString(salt);
            return saltHex;
        } catch (Exception e) {
            throw new Exception("Unable to generate salt", e);
        }
    }

    public static String byteArrayToHexString(byte[] b) {
        StringBuffer sb = new StringBuffer(b.length * 2);
        for (int i = 0; i < b.length; i++) {
            int v = b[i] & 0xff;
            if (v < 16) {
                sb.append('0');
            }
            sb.append(Integer.toHexString(v));
        }
        return sb.toString().toUpperCase();
    }

    public static byte[] hexStringToByteArray(String s) {
        byte[] b = new byte[s.length() / 2];
        for (int i = 0; i < b.length; i++) {
            int index = i * 2;
            int v = Integer.parseInt(s.substring(index, index + 2), 16);
            b[i] = (byte) v;
        }
        return b;
    }

    public static SecretKey getSecretKey(String password, String salt) throws Exception {
        try {
            PBEKeySpec pbeKeySpec = new PBEKeySpec(password.toCharArray(),
hexStringToByteArray(salt), PBE_ITERATION_COUNT, 256);
            SecretKeyFactory factory = SecretKeyFactory.getInstance(PBE_ALGORITHM, PROVIDER);
            SecretKey tmp = factory.generateSecret(pbeKeySpec);
            SecretKey secret = new SecretKeySpec(tmp.getEncoded(), SECRET_KEY_ALGORITHM);
            return secret;
        } catch (Exception e) {
            throw new Exception("Unable to get secret key", e);
        }
    }

    private static byte[] generateIv() throws NoSuchAlgorithmException,
NoSuchProviderException {
        SecureRandom random = SecureRandom.getInstance(RANDOM_ALGORITHM);
        byte[] iv = new byte[IV_LENGTH];
        random.nextBytes(iv);
        return iv;
    }
```

```
}
```

Lea Cómo almacenar contraseñas de forma segura en línea:
https://riptutorial.com/es/android/topic/9093/como-almacenar-contrasenas-de-forma-segura

Capítulo 57: Cómo utilizar SparseArray

Introducción

Un `SparseArray` es una alternativa para un `Map` . Un `Map` requiere que sus claves sean objetos. El fenómeno del autoboxing ocurre cuando queremos usar un valor `int` primitivo como clave. El compilador convierte automáticamente los valores primitivos a sus tipos encajonados (por ejemplo, `int` a `Integer`). La diferencia en la huella de memoria es notable: `int` usa 4 bytes, `Integer` usa 16 bytes. Un `SparseArray` utiliza `int` como valor clave.

Observaciones

Ventaja:

- Menor uso de memoria (por las claves primitivas).
- No hay auto-boxeo.

Desventaja:

- SparseArray usa la búsqueda binaria para encontrar el valor (O (log n)), por lo que puede que no sea la mejor solución si tiene que trabajar con un gran número de elementos (use HashMap).

Hay varias variantes de la familia como: -SparseArray <Integer, Object> -SparseBooleanArray <Integer, Boolean> -SparseIntArray <Integer, Integer> -SparseLongArray <Integer, Long> Long Long >

Operaciones SparseArray

- agregar elemento - put (int, x): agrega una asignación de la clave especificada al valor especificado, reemplazando la asignación anterior de la clave especificada si hubiera una. - añadir (int, x): coloca un par de clave / valor en la matriz, optimizando para el caso donde la clave es mayor que todas las claves existentes en la matriz. Debe usar append () en el caso de claves secuenciales para optimizar el rendimiento. De lo contrario, poner () está bien.

- remove element - delete (int): elimina la asignación de la clave especificada, si la hubiera. - removeAt (int): elimina la asignación en el índice dado. - removeAtRange (int, int): elimina un rango de asignaciones como un lote.

- access element - get (int): obtiene el int mapeado de la clave especificada, o 0 si no se ha realizado dicho mapeo. - get (int, E): obtiene el int mapeado de la clave especificada, o el valor especificado si no se ha realizado tal mapeo. - valueAt (int): dado un índice en el rango 0 ... tamaño () - 1, devuelve el valor de la asignación de clave-valor indexth que almacena este SparseIntArray. Los índices se ordenan en orden ascendente.

- búsqueda de índice / clave - keyAt (int): dado un índice en el rango 0 ... tamaño () - 1,

devuelve la clave de la asignación de valor-clave indexth que almacena este SparseIntArray. Los índices se ordenan en orden ascendente. - valueAt (int): dado un índice en el rango 0 ... tamaño () - 1, devuelve el valor de la asignación de clave-valor indexth que almacena este SparseIntArray. Los índices se ordenan en orden ascendente. - indexOfKey (int): devuelve el índice para el cual keyAt (int) devolvería la clave especificada, o un número negativo si la clave especificada no está asignada. - indexOfValue (E): devuelve un índice para el cual valueAt (int) devolvería la clave especificada, o un número negativo si no hay claves asignadas al valor especificado. Tenga en cuenta que esta es una búsqueda lineal, a diferencia de las búsquedas por clave, y que varias claves se pueden asignar al mismo valor y esto solo encontrará una de ellas. La diferencia en su huella de memoria es notable: el int usa 4 bytes, el entero usa 16 bytes. SparArray usa int como valor clave.

Examples

Ejemplo básico utilizando SparseArray

```
class Person {
    String name;

    public Person(String name) {
        this.name = name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Person person = (Person) o;

        return name != null ? name.equals(person.name) : person.name == null;
    }

    @Override
    public int hashCode() {
        return name != null ? name.hashCode() : 0;
    }

    @Override
    public String toString() {
        return "Person{" +
                "name='" + name + '\'' +
                '}';
    }
}

final Person steve = new Person("Steve");
Person[] persons = new Person[] { new Person("John"), new Person("Gwen"), steve, new
Person("Rob") };
int[] identifiers = new int[] {1234, 2345, 3456, 4567};

final SparseArray<Person> demo = new SparseArray<>();

// Mapping persons to identifiers.
for (int i = 0; i < persons.length; i++) {
```

```
    demo.put(identifiers[i], persons[i]);
}

// Find the person with identifier 1234.
Person id1234 = demo.get(1234); // Returns John.

// Find the person with identifier 6410.
Person id6410 = demo.get(6410); // Returns null.

// Find the 3rd person.
Person third = demo.valueAt(3); // Returns Rob.

// Find the 42th person.
//Person fortysecond = demo.valueAt(42); // Throws ArrayIndexOutOfBoundsException.

// Remove the last person.
demo.removeAt(demo.size() - 1); // Rob removed.

// Remove the person with identifier 1234.
demo.delete(1234); // John removed.

// Find the index of Steve.
int indexOfSteve = demo.indexOfValue(steve);

// Find the identifier of Steve.
int identifierOfSteve = demo.keyAt(indexOfSteve);
```

Tutorial en YouTube

Lea Cómo utilizar SparseArray en línea: https://riptutorial.com/es/android/topic/8824/como-utilizar-sparsearray

Capítulo 58: Componentes de la arquitectura de Android

Introducción

Android Architecture Components es una nueva colección de bibliotecas que te ayudan a diseñar aplicaciones robustas, comprobables y mantenibles. Las partes principales son: Ciclos de vida, ViewModel, LiveData, Room.

Examples

Añadir componentes de arquitectura

Proyecto build.gradle

```
allprojects {
    repositories {
        jcenter()
        // Add this if you use Gradle 4.0+
        google()
        // Add this if you use Gradle < 4.0
        maven { url 'https://maven.google.com' }
    }
}

ext {
    archVersion = '1.0.0-alpha5'
}
```

Aplicación para construir Gradle

```
// For Lifecycles, LiveData, and ViewModel
compile "android.arch.lifecycle:runtime:$archVersion"
compile "android.arch.lifecycle:extensions:$archVersion"
annotationProcessor "android.arch.lifecycle:compiler:$archVersion"

// For Room
compile "android.arch.persistence.room:runtime:$archVersion"
annotationProcessor "android.arch.persistence.room:compiler:$archVersion"

// For testing Room migrations
testCompile "android.arch.persistence.room:testing:$archVersion"

// For Room RxJava support
compile "android.arch.persistence.room:rxjava2:$archVersion"
```

Usando Lifecycle en AppCompatActivity

Extiende tu actividad de esta actividad

```
public abstract class BaseCompatLifecycleActivity extends AppCompatActivity implements
LifecycleRegistryOwner {
    // We need this class, because LifecycleActivity extends FragmentActivity not
AppCompatActivity

    @NonNull
    private final LifecycleRegistry lifecycleRegistry = new LifecycleRegistry(this);

    @NonNull
    @Override
    public LifecycleRegistry getLifecycle() {
        return lifecycleRegistry;
    }
}
```

ViewModel con transformaciones de LiveData

```
public class BaseViewModel extends ViewModel {
    private static final int TAG_SEGMENT_INDEX = 2;
    private static final int VIDEOS_LIMIT = 100;

    // We save input params here
    private final MutableLiveData<Pair<String, String>> urlWithReferrerLiveData = new
MutableLiveData<>();

    // transform specific uri param to "tag"
    private final LiveData<String> currentTagLiveData =
Transformations.map(urlWithReferrerLiveData, pair -> {
        Uri uri = Uri.parse(pair.first);
        List<String> segments = uri.getPathSegments();
        if (segments.size() > TAG_SEGMENT_INDEX)
            return segments.get(TAG_SEGMENT_INDEX);
        return null;
    });

    // transform "tag" to videos list
    private final LiveData<List<VideoItem>> videoByTagData =
Transformations.switchMap(currentTagLiveData, tag -> contentRepository.getVideoByTag(tag,
VIDEOS_LIMIT));

    ContentRepository contentRepository;

    public BaseViewModel() {
        // some inits
    }

    public void setUrlWithReferrer(String url, String referrer) {
        // set value activates observers and transformations
        urlWithReferrerLiveData.setValue(new Pair<>(url, referrer));
    }

    public LiveData<List<VideoItem>> getVideoByTagData() {
        return videoByTagData;
    }
}
```

En algún lugar de la interfaz de usuario:

```
public class VideoActivity extends BaseCompatLifecycleActivity {
    private VideoViewModel viewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Get ViewModel
        viewModel = ViewModelProviders.of(this).get(BaseViewModel.class);
        // Add observer
        viewModel.getVideoByTagData().observe(this, data -> {
            // some checks
            adapter.updateData(data);
        });

        ...
        if (savedInstanceState == null) {
            // init loading only at first creation
            // you just set params and
            viewModel.setUrlWithReferrer(url, referrer);
        }
}
```

Habitación peristence

La sala requiere cuatro partes: clase de base de datos, clases DAO, clases de entidad y clases de migración (ahora puede usar **solo métodos DDL**):

Clases de entidad

```
// Set custom table name, add indexes
@Entity(tableName = "videos",
        indices = {@Index("title")}
)
public final class VideoItem {
    @PrimaryKey // required
    public long articleId;
    public String title;
    public String url;
}

// Use ForeignKey for setup table relation
@Entity(tableName = "tags",
        indices = {@Index("score"), @Index("videoId"), @Index("value")},
        foreignKeys = @ForeignKey(entity = VideoItem.class,
                parentColumns = "articleId",
                childColumns = "videoId",
                onDelete = ForeignKey.CASCADE)
)
public final class VideoTag {
    @PrimaryKey
    public long id;
    public long videoId;
    public String displayName;
    public String value;
    public double score;
}
```

Clases de dao

```
@Dao
public interface VideoDao {
    // Create insert with custom conflict strategy
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void saveVideos(List<VideoItem> videos);

    // Simple update
    @Update
    void updateVideos(VideoItem... videos);

    @Query("DELETE FROM tags WHERE videoId = :videoId")
    void deleteTagsByVideoId(long videoId);

    // Custom query, you may use select/delete here
    @Query("SELECT v.* FROM tags t LEFT JOIN videos v ON v.articleId = t.videoId WHERE t.value
= :tag ORDER BY updatedAt DESC LIMIT :limit")
    LiveData<List<VideoItem>> getVideosByTag(String tag, int limit);
}
```

Clase de base de datos

```
// register your entities and DAOs
@Database(entities = {VideoItem.class, VideoTag.class}, version = 2)
public abstract class ContentDatabase extends RoomDatabase {
    public abstract VideoDao videoDao();
}
```

Migraciones

```
public final class Migrations {
    private static final Migration MIGRATION_1_2 = new Migration(1, 2) {
        @Override
        public void migrate(SupportSQLiteDatabase database) {
            final String[] sqlQueries = {
                    "CREATE TABLE IF NOT EXISTS `tags` (`id` INTEGER PRIMARY KEY
AUTOINCREMENT," +
                            " `videoId` INTEGER, `displayName` TEXT, `value` TEXT, `score`
REAL," +
                            " FOREIGN KEY(`videoId`) REFERENCES `videos`(`articleId`)" +
                            " ON UPDATE NO ACTION ON DELETE CASCADE )",
                    "CREATE  INDEX `index_tags_score` ON `tags` (`score`)",
                    "CREATE  INDEX `index_tags_videoId` ON `tags` (`videoId`)"};
            for (String query : sqlQueries) {
                database.execSQL(query);
            }
        }
    };

    public static final Migration[] ALL = {MIGRATION_1_2};

    private Migrations() {
    }
}
```

Usar en la clase de aplicación o proporcionar a través de Dagger

```
ContentDatabase provideContentDatabase() {
    return Room.databaseBuilder(context, ContentDatabase.class, "data.db")
            .addMigrations(Migrations.ALL).build();
}
```

Escribe tu repositorio:

```
public final class ContentRepository {
    private final ContentDatabase db;
    private final VideoDao videoDao;

    public ContentRepository(ContentDatabase contentDatabase, VideoDao videoDao) {
        this.db = contentDatabase;
        this.videoDao = videoDao;
    }

    public LiveData<List<VideoItem>> getVideoByTag(@Nullable String tag, int limit) {
        // you may fetch from network, save to database
        ....
        return videoDao.getVideosByTag(tag, limit);
    }
}
```

Usar en ViewModel:

```
ContentRepository contentRepository = ...;
contentRepository.getVideoByTag(tag, limit);
```

LiveData personalizado

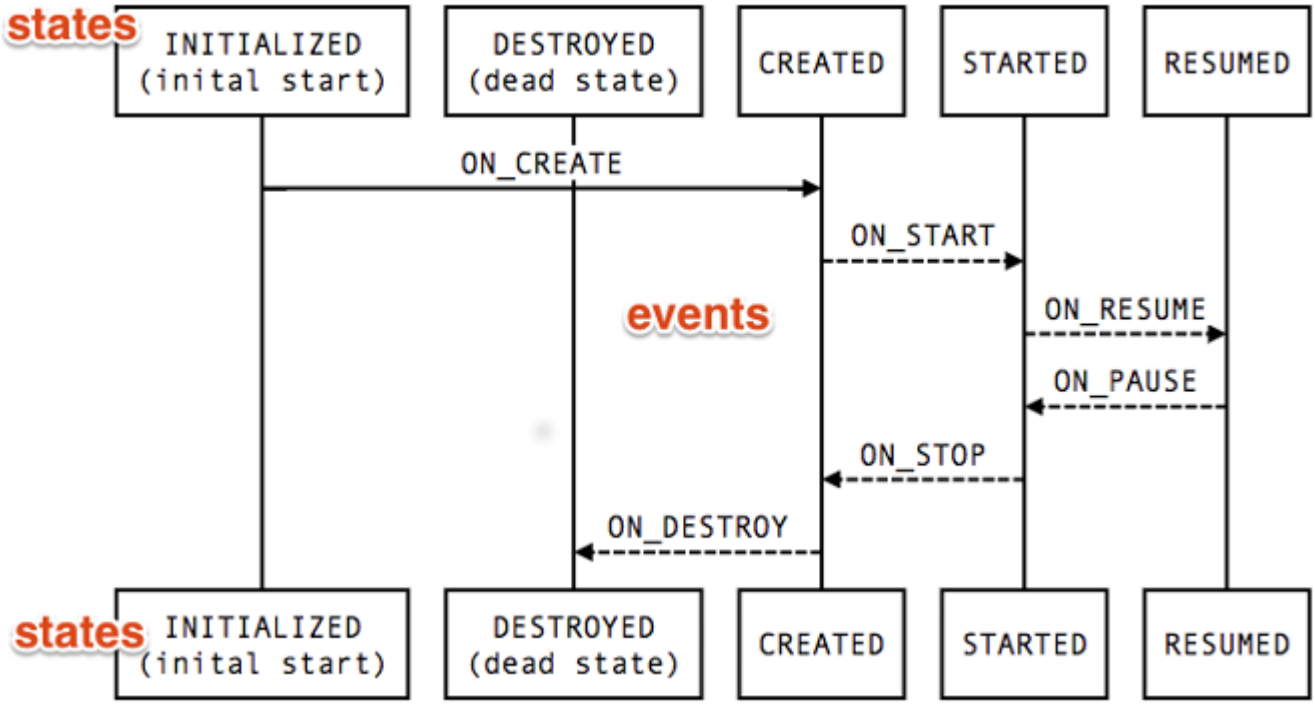
Puede escribir LiveData personalizado, si necesita lógica personalizada.
No escriba una clase personalizada, si solo necesita transformar datos (use la clase Transformaciones)

```
public class LocationLiveData extends LiveData<Location> {
    private LocationManager locationManager;

    private LocationListener listener = new LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            setValue(location);
        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {
            // Do something
        }

        @Override
        public void onProviderEnabled(String provider) {
            // Do something
        }

        @Override
        public void onProviderDisabled(String provider) {
            // Do something
```

```
        }
    };

    public LocationLiveData(Context context) {
        locationManager = (LocationManager)
context.getSystemService(Context.LOCATION_SERVICE);
    }

    @Override
    protected void onActive() {
        // We have observers, start working
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, listener);
    }

    @Override
    protected void onInactive() {
        // We have no observers, stop working
        locationManager.removeUpdates(listener);
    }
}
```

Componente personalizado de ciclo de vida

Cada ciclo de vida del componente UI cambió como se muestra en la imagen.



Puede crear un componente, que se notificará en el cambio de estado del ciclo de vida:

```
public class MyLocationListener implements LifecycleObserver {
    private boolean enabled = false;
    private Lifecycle lifecycle;
    public MyLocationListener(Context context, Lifecycle lifecycle, Callback callback) {
       ...
    }

    @OnLifecycleEvent(Lifecycle.Event.ON_START)
```

```
    void start() {
        if (enabled) {
           // connect
        }
    }

    public void enable() {
        enabled = true;
        if (lifecycle.getState().isAtLeast(STARTED)) {
            // connect if not connected
        }
    }

    @OnLifecycleEvent(Lifecycle.Event.ON_STOP)
    void stop() {
        // disconnect if connected
    }
}
```

Lea Componentes de la arquitectura de Android en línea:
https://riptutorial.com/es/android/topic/10872/componentes-de-la-arquitectura-de-android

Capítulo 59: Compresión de imagen

Examples

Cómo comprimir la imagen sin cambio de tamaño.

Consiga el **mapa** de **bits comprimido** de la clase Singleton:

```
ImageView imageView = (ImageView)findViewById(R.id.imageView);
Bitmap bitmap = ImageUtils.getInstant().getCompressedBitmap("Your_Image_Path_Here");
imageView.setImageBitmap(bitmap);
```

ImageUtils.java :

```
public class ImageUtils {

    public static ImageUtils mInstant;

    public static ImageUtils getInstant(){
        if(mInstant==null){
            mInstant = new ImageUtils();
        }
        return mInstant;
    }

    public  Bitmap getCompressedBitmap(String imagePath) {
        float maxHeight = 1920.0f;
        float maxWidth = 1080.0f;
        Bitmap scaledBitmap = null;
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inJustDecodeBounds = true;
        Bitmap bmp = BitmapFactory.decodeFile(imagePath, options);

        int actualHeight = options.outHeight;
        int actualWidth = options.outWidth;
        float imgRatio = (float) actualWidth / (float) actualHeight;
        float maxRatio = maxWidth / maxHeight;

        if (actualHeight > maxHeight || actualWidth > maxWidth) {
            if (imgRatio < maxRatio) {
                imgRatio = maxHeight / actualHeight;
                actualWidth = (int) (imgRatio * actualWidth);
                actualHeight = (int) maxHeight;
            } else if (imgRatio > maxRatio) {
                imgRatio = maxWidth / actualWidth;
                actualHeight = (int) (imgRatio * actualHeight);
                actualWidth = (int) maxWidth;
            } else {
                actualHeight = (int) maxHeight;
                actualWidth = (int) maxWidth;

            }
        }

        options.inSampleSize = calculateInSampleSize(options, actualWidth, actualHeight);
```

```
        options.inJustDecodeBounds = false;
        options.inDither = false;
        options.inPurgeable = true;
        options.inInputShareable = true;
        options.inTempStorage = new byte[16 * 1024];

        try {
            bmp = BitmapFactory.decodeFile(imagePath, options);
        } catch (OutOfMemoryError exception) {
            exception.printStackTrace();

        }
        try {
            scaledBitmap = Bitmap.createBitmap(actualWidth, actualHeight,
Bitmap.Config.ARGB_8888);
        } catch (OutOfMemoryError exception) {
            exception.printStackTrace();
        }

        float ratioX = actualWidth / (float) options.outWidth;
        float ratioY = actualHeight / (float) options.outHeight;
        float middleX = actualWidth / 2.0f;
        float middleY = actualHeight / 2.0f;

        Matrix scaleMatrix = new Matrix();
        scaleMatrix.setScale(ratioX, ratioY, middleX, middleY);

        Canvas canvas = new Canvas(scaledBitmap);
        canvas.setMatrix(scaleMatrix);
        canvas.drawBitmap(bmp, middleX - bmp.getWidth() / 2, middleY - bmp.getHeight() / 2,
new Paint(Paint.FILTER_BITMAP_FLAG));

        ExifInterface exif = null;
        try {
            exif = new ExifInterface(imagePath);
            int orientation = exif.getAttributeInt(ExifInterface.TAG_ORIENTATION, 0);
            Matrix matrix = new Matrix();
            if (orientation == 6) {
                matrix.postRotate(90);
            } else if (orientation == 3) {
                matrix.postRotate(180);
            } else if (orientation == 8) {
                matrix.postRotate(270);
            }
            scaledBitmap = Bitmap.createBitmap(scaledBitmap, 0, 0, scaledBitmap.getWidth(),
scaledBitmap.getHeight(), matrix, true);
        } catch (IOException e) {
            e.printStackTrace();
        }
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        scaledBitmap.compress(Bitmap.CompressFormat.JPEG, 85, out);

        byte[] byteArray = out.toByteArray();

        Bitmap updatedBitmap = BitmapFactory.decodeByteArray(byteArray, 0, byteArray.length);

        return updatedBitmap;
    }

    private int calculateInSampleSize(BitmapFactory.Options options, int reqWidth, int
reqHeight) {
```

```
        final int height = options.outHeight;
        final int width = options.outWidth;
        int inSampleSize = 1;

        if (height > reqHeight || width > reqWidth) {
            final int heightRatio = Math.round((float) height / (float) reqHeight);
            final int widthRatio = Math.round((float) width / (float) reqWidth);
            inSampleSize = heightRatio < widthRatio ? heightRatio : widthRatio;
        }
        final float totalPixels = width * height;
        final float totalReqPixelsCap = reqWidth * reqHeight * 2;

        while (totalPixels / (inSampleSize * inSampleSize) > totalReqPixelsCap) {
            inSampleSize++;
        }
        return inSampleSize;
    }
}
```

Las dimensiones son las mismas después de comprimir Bitmap .

¿Cómo lo comprobé?

```
Bitmap beforeBitmap = BitmapFactory.decodeFile("Your_Image_Path_Here");
Log.i("Before Compress Dimension", beforeBitmap.getWidth()+"-"+beforeBitmap.getHeight());

Bitmap afterBitmap = ImageUtils.getInstant().getCompressedBitmap("Your_Image_Path_Here");
Log.i("After Compress Dimension", afterBitmap.getWidth() + "-" + afterBitmap.getHeight());
```

Salida:

```
Before Compress : Dimension: 1080-1452
After Compress : Dimension: 1080-1452
```

Lea Compresión de imagen en línea: https://riptutorial.com/es/android/topic/5588/compresion-de-imagen

Capítulo 60: Compruebe la conectividad a internet

Introducción

Este método se utiliza para comprobar si el tiempo de conexión Wi-Fi está conectado o no.

Sintaxis

- isNetworkAvailable (): para comprobar si Internet está disponible en el dispositivo

Parámetros

Parámetro	Detalle
Contexto	Una referencia de contexto de actividad.

Observaciones

Si Internet está conectado, entonces el método devolverá verdadero o falso.

Examples

Compruebe si el dispositivo tiene conectividad a internet

Agregue los permisos de red necesarios al archivo de manifiesto de la aplicación:

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
```

```
/**
 * If network connectivity is available, will return true
 *
 * @param context the current context
 * @return boolean true if a network connection is available
 */
public static boolean isNetworkAvailable(Context context) {
    ConnectivityManager connectivity = (ConnectivityManager) context
            .getSystemService(Context.CONNECTIVITY_SERVICE);
    if (connectivity == null) {
        Log.d("NetworkCheck", "isNetworkAvailable: No");
        return false;
    }
```

```
    // get network info for all of the data interfaces (e.g. WiFi, 3G, LTE, etc.)
    NetworkInfo[] info = connectivity.getAllNetworkInfo();

    // make sure that there is at least one interface to test against
    if (info != null) {
        // iterate through the interfaces
        for (int i = 0; i < info.length; i++) {
            // check this interface for a connected state
            if (info[i].getState() == NetworkInfo.State.CONNECTED) {
                Log.d("NetworkCheck", "isNetworkAvailable: Yes");
                return true;
            }
        }
    }
    return false;
}
```

¿Cómo comprobar la fuerza de la red en Android?

```
        ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo Info = cm.getActiveNetworkInfo();
        if (Info == null || !Info.isConnectedOrConnecting()) {
            Log.i(TAG, "No connection");
        } else {
            int netType = Info.getType();
            int netSubtype = Info.getSubtype();

            if (netType == ConnectivityManager.TYPE_WIFI) {
                Log.i(TAG, "Wifi connection");
                WifiManager wifiManager = (WifiManager)
getApplication().getSystemService(Context.WIFI_SERVICE);
                List<ScanResult> scanResult = wifiManager.getScanResults();
                for (int i = 0; i < scanResult.size(); i++) {
                    Log.d("scanResult", "Speed of wifi"+scanResult.get(i).level);//The db
level of signal
                }


                // Need to get wifi strength
            } else if (netType == ConnectivityManager.TYPE_MOBILE) {
                Log.i(TAG, "GPRS/3G connection");
                // Need to get differentiate between 3G/GPRS
            }
        }
```

Cómo comprobar la fuerza de la red

Para verificar la fuerza exacta en decibelios use esto-

```
ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo Info = cm.getActiveNetworkInfo();
    if (Info == null || !Info.isConnectedOrConnecting()) {
        Log.i(TAG, "No connection");
    } else {
        int netType = Info.getType();
        int netSubtype = Info.getSubtype();
```

```
        if (netType == ConnectivityManager.TYPE_WIFI) {
            Log.i(TAG, "Wifi connection");
            WifiManager wifiManager = (WifiManager)
getApplication().getSystemService(Context.WIFI_SERVICE);
            List<ScanResult> scanResult = wifiManager.getScanResults();
            for (int i = 0; i < scanResult.size(); i++) {
                Log.d("scanResult", "Speed of wifi"+scanResult.get(i).level);//The db level of
signal
            }


            // Need to get wifi strength
        } else if (netType == ConnectivityManager.TYPE_MOBILE) {
            Log.i(TAG, "GPRS/3G connection");
            // Need to get differentiate between 3G/GPRS
        }
    }
```

Para verificar el tipo de red use esta Clase

```
 public class Connectivity {
        /*
         * These constants aren't yet available in my API level (7), but I need to
         * handle these cases if they come up, on newer versions
         */
        public static final int NETWORK_TYPE_EHRPD = 14; // Level 11
        public static final int NETWORK_TYPE_EVDO_B = 12; // Level 9
        public static final int NETWORK_TYPE_HSPAP = 15; // Level 13
        public static final int NETWORK_TYPE_IDEN = 11; // Level 8
        public static final int NETWORK_TYPE_LTE = 13; // Level 11

    /**
     * Check if there is any connectivity
     *
     * @param context
     * @return
     */
    public static boolean isConnected(Context context) {
        ConnectivityManager cm = (ConnectivityManager) context
                .getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo info = cm.getActiveNetworkInfo();
        return (info != null && info.isConnected());
    }

    /**
     * Check if there is fast connectivity
     *
     * @param context
     * @return
     */
    public static String isConnectedFast(Context context) {
        ConnectivityManager cm = (ConnectivityManager) context
                .getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo info = cm.getActiveNetworkInfo();

        if ((info != null && info.isConnected())) {
            return Connectivity.isConnectionFast(info.getType(),
                    info.getSubtype());
        } else
```

```
        return "No NetWork Access";

    }

    /**
     * Check if the connection is fast
     *
     * @param type
     * @param subType
     * @return
     */
    public static String isConnectionFast(int type, int subType) {
        if (type == ConnectivityManager.TYPE_WIFI) {
            System.out.println("CONNECTED VIA WIFI");
            return "CONNECTED VIA WIFI";
        } else if (type == ConnectivityManager.TYPE_MOBILE) {
            switch (subType) {
            case TelephonyManager.NETWORK_TYPE_1xRTT:
                return "NETWORK TYPE 1xRTT"; // ~ 50-100 kbps
            case TelephonyManager.NETWORK_TYPE_CDMA:
                return "NETWORK TYPE CDMA (3G) Speed: 2 Mbps"; // ~ 14-64 kbps
            case TelephonyManager.NETWORK_TYPE_EDGE:

                return "NETWORK TYPE EDGE (2.75G) Speed: 100-120 Kbps"; // ~
                                                                        // 50-100
                                                                        // kbps
            case TelephonyManager.NETWORK_TYPE_EVDO_0:
                return "NETWORK TYPE EVDO_0"; // ~ 400-1000 kbps
            case TelephonyManager.NETWORK_TYPE_EVDO_A:
                return "NETWORK TYPE EVDO_A"; // ~ 600-1400 kbps
            case TelephonyManager.NETWORK_TYPE_GPRS:
                return "NETWORK TYPE GPRS (2.5G) Speed: 40-50 Kbps"; // ~ 100
                                                                     // kbps
            case TelephonyManager.NETWORK_TYPE_HSDPA:
                return "NETWORK TYPE HSDPA (4G) Speed: 2-14 Mbps"; // ~ 2-14
                                                                   // Mbps
            case TelephonyManager.NETWORK_TYPE_HSPA:
                return "NETWORK TYPE HSPA (4G) Speed: 0.7-1.7 Mbps"; // ~
                                                                     // 700-1700
                                                                     // kbps
            case TelephonyManager.NETWORK_TYPE_HSUPA:
                return "NETWORK TYPE HSUPA (3G) Speed: 1-23 Mbps"; // ~ 1-23
                                                                   // Mbps
            case TelephonyManager.NETWORK_TYPE_UMTS:
                return "NETWORK TYPE UMTS (3G) Speed: 0.4-7 Mbps"; // ~ 400-7000
                                                                   // kbps
                // NOT AVAILABLE YET IN API LEVEL 7
            case Connectivity.NETWORK_TYPE_EHRPD:
                return "NETWORK TYPE EHRPD"; // ~ 1-2 Mbps
            case Connectivity.NETWORK_TYPE_EVDO_B:
                return "NETWORK_TYPE_EVDO_B"; // ~ 5 Mbps
            case Connectivity.NETWORK_TYPE_HSPAP:
                return "NETWORK TYPE HSPA+ (4G) Speed: 10-20 Mbps"; // ~ 10-20
                                                                    // Mbps
            case Connectivity.NETWORK_TYPE_IDEN:
                return "NETWORK TYPE IDEN"; // ~25 kbps
            case Connectivity.NETWORK_TYPE_LTE:
                return "NETWORK TYPE LTE (4G) Speed: 10+ Mbps"; // ~ 10+ Mbps
                // Unknown
            case TelephonyManager.NETWORK_TYPE_UNKNOWN:
                return "NETWORK TYPE UNKNOWN";
```

```
            default:
                return "";
            }
        } else {
            return "";
        }
    }

}
```

Lea Compruebe la conectividad a internet en línea:
https://riptutorial.com/es/android/topic/3918/compruebe-la-conectividad-a-internet

Capítulo 61: Compruebe la conexión de datos

Examples

Comprobar conexión de datos

Este método es para verificar la conexión de datos haciendo ping a cierta IP o nombre de dominio.

```
public Boolean isDataConnected() {
    try {
        Process p1 = java.lang.Runtime.getRuntime().exec("ping -c 1 8.8.8.8");
        int returnVal = p1.waitFor();
        boolean reachable = (returnVal==0);
        return reachable;
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return false;
}
```

Compruebe la conexión utilizando ConnectivityManager

```
    public static boolean isConnectedNetwork (Context context) {

        ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
        return cm.getActiveNetworkInfo () != null && cm.getActiveNetworkInfo
().isConnectedOrConnecting ();

    }
```

Use los intentos de la red para realizar tareas mientras se permiten los datos

Cuando su dispositivo se conecta a una red, se envía un intento. Muchas aplicaciones no verifican estos intentos, pero para hacer que su aplicación funcione correctamente, puede escuchar los intentos de cambio de red que le indicarán cuándo es posible la comunicación. Para verificar la conectividad de la red, puede, por ejemplo, usar la siguiente cláusula:

```
if
(intent.getAction().equals(android.net.ConnectivityManager.CONNECTIVITY_ACTION)){
 NetworkInfo info =
intent.getParcelableExtra(ConnectivityManager.EXTRA_NETWORK_INFO);
//perform your action when connected to a network
}
```

Lea Compruebe la conexión de datos en línea:
https://riptutorial.com/es/android/topic/8670/compruebe-la-conexion-de-datos

Capítulo 62: Conexiones Wi-Fi

Examples

Conectar con cifrado WEP

Este ejemplo se conecta a un punto de acceso Wi-Fi con cifrado WEP, dado un SSID y la contraseña.

```
public boolean ConnectToNetworkWEP(String networkSSID, String password)
{
    try {
        WifiConfiguration conf = new WifiConfiguration();
        conf.SSID = "\"" + networkSSID + "\"";   // Please note the quotes. String should
contain SSID in quotes
        conf.wepKeys[0] = "\"" + password + "\""; //Try it with quotes first

        conf.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.NONE);
        conf.allowedGroupCiphers.set(WifiConfiguration.AuthAlgorithm.OPEN);
        conf.allowedGroupCiphers.set(WifiConfiguration.AuthAlgorithm.SHARED);

        WifiManager wifiManager = (WifiManager)
this.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        int networkId = wifiManager.addNetwork(conf);

        if (networkId == -1){
            //Try it again with no quotes in case of hex password
            conf.wepKeys[0] = password;
            networkId = wifiManager.addNetwork(conf);
        }

        List<WifiConfiguration> list = wifiManager.getConfiguredNetworks();
        for( WifiConfiguration i : list ) {
            if(i.SSID != null && i.SSID.equals("\"" + networkSSID + "\"")) {
                wifiManager.disconnect();
                wifiManager.enableNetwork(i.networkId, true);
                wifiManager.reconnect();
                break;
            }
        }

        //WiFi Connection success, return true
        return true;
    } catch (Exception ex) {
        System.out.println(Arrays.toString(ex.getStackTrace()));
        return false;
    }
}
```

Conectar con cifrado WPA2

Este ejemplo se conecta a un punto de acceso Wi-Fi con cifrado WPA2.

```
public boolean ConnectToNetworkWPA(String networkSSID, String password) {
```

```
  try {
    WifiConfiguration conf = new WifiConfiguration();
    conf.SSID = "\"" + networkSSID + "\"";   // Please note the quotes. String should contain
SSID in quotes

    conf.preSharedKey = "\"" + password + "\"";

    conf.status = WifiConfiguration.Status.ENABLED;
    conf.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.TKIP);
    conf.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.CCMP);
    conf.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.WPA_PSK);
    conf.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.TKIP);
    conf.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.CCMP);

    Log.d("connecting", conf.SSID + " " + conf.preSharedKey);

    WifiManager wifiManager = (WifiManager)
this.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
    wifiManager.addNetwork(conf);

    Log.d("after connecting", conf.SSID + " " + conf.preSharedKey);

    List<WifiConfiguration> list = wifiManager.getConfiguredNetworks();
    for( WifiConfiguration i : list ) {
      if(i.SSID != null && i.SSID.equals("\"" + networkSSID + "\"")) {
        wifiManager.disconnect();
        wifiManager.enableNetwork(i.networkId, true);
        wifiManager.reconnect();
        Log.d("re connecting", i.SSID + " " + conf.preSharedKey);

        break;
      }
    }

    //WiFi Connection success, return true
    return true;
  } catch (Exception ex) {
    System.out.println(Arrays.toString(ex.getStackTrace()));
    return false;
  }
}
```

Escanear en busca de puntos de acceso

Este ejemplo busca puntos de acceso disponibles y redes ad hoc. `btnScan` activa una exploración iniciada por el método `WifiManager.startScan()` . Después de la exploración, `WifiManager` llama a la intención `SCAN_RESULTS_AVAILABLE_ACTION` y la clase `WifiScanReceiver` procesa el resultado de la exploración. Los resultados se muestran en un `TextView` .

```
public class MainActivity extends AppCompatActivity {

    private final static String TAG = "MainActivity";

    TextView txtWifiInfo;
    WifiManager wifi;
    WifiScanReceiver wifiReceiver;

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        wifi=(WifiManager)getSystemService(Context.WIFI_SERVICE);
        wifiReceiver = new WifiScanReceiver();

        txtWifiInfo = (TextView)findViewById(R.id.txtWifiInfo);
        Button btnScan = (Button)findViewById(R.id.btnScan);
        btnScan.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Log.i(TAG, "Start scan...");
                wifi.startScan();
            }
        });
    }

    protected void onPause() {
        unregisterReceiver(wifiReceiver);
        super.onPause();
    }

    protected void onResume() {
        registerReceiver(
            wifiReceiver,
            new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)
        );
        super.onResume();
    }

    private class WifiScanReceiver extends BroadcastReceiver {
        public void onReceive(Context c, Intent intent) {
            List<ScanResult> wifiScanList = wifi.getScanResults();
            txtWifiInfo.setText("");
            for(int i = 0; i < wifiScanList.size(); i++){
                String info = ((wifiScanList.get(i)).toString());
                txtWifiInfo.append(info+"\n\n");
            }
        }
    }
}
```

Permisos

Los siguientes permisos deben definirse en *AndroidManifest.xml* :

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
```

`android.permission.ACCESS_WIFI_STATE` es necesario para llamar a `WifiManager.getScanResults()` . Sin `android.permission.CHANGE_WIFI_STATE` no puede iniciar una exploración con `WifiManager.startScan()` .

Al compilar el proyecto para el nivel de API 23 o superior (Android 6.0 y versiones posteriores), se debe insertar `android.permission.ACCESS_FINE_LOCATION` o `android.permission.ACCESS_COARSE_LOCATION` . Además, ese permiso debe solicitarse, por ejemplo, en el método `onCreate` de su actividad

principal:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    String[] PERMS_INITIAL={
            Manifest.permission.ACCESS_FINE_LOCATION,
    };
    ActivityCompat.requestPermissions(this, PERMS_INITIAL, 127);
}
```

Lea Conexiones Wi-Fi en línea: https://riptutorial.com/es/android/topic/3288/conexiones-wi-fi

Capítulo 63: Configuración de Jenkins CI para proyectos de Android

Examples

Enfoque paso a paso para configurar Jenkins para Android

Esta es una guía paso a paso para configurar el proceso de compilación automatizado utilizando Jenkins CI para sus proyectos de Android. Los siguientes pasos asumen que usted tiene nuevo hardware con cualquier tipo de Linux instalado. También se tiene en cuenta que es posible que tenga una máquina remota.

PARTE I: configuración inicial en su máquina

1. Inicie sesión a través de *ssh* en su máquina de Ubuntu:

   ssh username@xxx.xxx.xxx

2. Descargue una versión del SDK de Android en su máquina:

   wget https://dl.google.com/android/android-sdk_r24.4.1-linux.tgz

3. Descomprima el archivo *tar* descargado:

   sudo apt-get install tar
   tar -xvf android-sdk_r24.4.1-linux.tgz

4. Ahora necesita instalar Java 8 en su máquina Ubuntu, que es un requisito para las compilaciones de Android en Nougat. Jenkins le solicitará que instale JDK y JRE 7 siguiendo los pasos a continuación:

   sudo apt-get install python-software-properties
   sudo add-apt-repository ppa: webupd8team / java
   sudo apt-get update
   apt-get install openjdk-8-jdk

5. Ahora instala Jenkins en tu máquina Ubuntu:

   wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | Sudo apt-key add -
   sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary /> /etc/apt/sources.list.d/jenkins.list'
   sudo apt-get update
   sudo apt-get install jenkins

6. Descargue la última versión de Gradle compatible para su configuración de Android:

wget https://services.gradle.org/distributions/gradle-2.14.1-all.zip
descomprimir gradle-2.14.1-all.zip

7. Configure Android en su máquina Ubuntu. Primero mueva a la carpeta de *herramientas* en la carpeta SDK de Android descargada en el paso 2:

   cd android-sdk-linux / tools **// listas disponibles SDK**
   actualización de Android sdk --no-ui **// Actualiza la versión del SDK**
   lista de android sdk -a | grep "SDK Build-tools" **// muestra las herramientas de construcción disponibles**
   actualización de Android sdk -a -u -t 4 **// actualiza la versión de las herramientas de compilación a una que figura como 4 según la versión anterior. cmd.**
   actualizar java

8. Instala *Git* o cualquier otro VCS en tu máquina:

   sudo apt-get install git

9. Ahora inicie sesión en Jenkins utilizando su navegador de internet. Escriba `ipAddress:8080` en la barra de direcciones.

10. Para recibir la contraseña del primer inicio de sesión, verifique el archivo correspondiente de la siguiente manera (necesitará los permisos para acceder a este archivo):

    cat / var / lib / jenkins / secrets / initialAdminPassword

PARTE II: configurar Jenkins para construir trabajos de Android

1. Una vez que haya iniciado sesión, vaya a la siguiente ruta:

   Jenkins> Gestionar Jenkins> Configuración global de herramientas

2. En esta ubicación, agregue `JAVA_HOME` con las siguientes entradas:

   Nombre = JAVA_HOME
   JAVA_HOME = / usr / lib / jvm / java-8-openjdk-amd64

3. También agregue los siguientes valores a *Git* y guarde las variables de entorno:

   Nombre = Predeterminado
   / usr / bin / git

4. Ahora ve a la siguiente ruta:

   Jenkins> Gestionar Jenkins> Configuración

5. En esta ubicación, agregue `ANDROID_HOME` a las "propiedades globales":

   Nombre = ANDROID_HOME
   Valor = / home / username / android-sdk-linux

Parte III: crea un trabajo de Jenkins para tu proyecto de Android

1. Haga clic en *Nuevo elemento* en la pantalla de inicio de Jenkins.
2. Añadir un *nombre* y una *descripción del proyecto* .
3. En la pestaña *General* , seleccione *Avanzado* . Luego seleccione *Usar espacio de trabajo personalizado* :

   Directorio / inicio / usuario / Código / ProjectFolder
4. En la gestión del código fuente seleccione *Git* . Estoy usando *Bitbucket* para el propósito de este ejemplo:

   URL del repositorio = https: // nombre de usuario: contraseña@bitbucket.org/project/projectname.git
5. Seleccione comportamientos adicionales para su repositorio:

   Limpiar antes de pagar
   Checkout a un subdirectorio. Subdirectorio local para repo / home / user / Code / ProjectFolder
6. Seleccione la rama que desea construir:

   */dominar
7. En la pestaña *Generar* , seleccione *Ejecutar shell* en *Agregar paso de compilación* .
8. En el *shell de ejecución* , agregue el siguiente comando:

   cd / home / user / Code / ProjectFolder && gradle clean assemble --no-daemon
9. Si desea ejecutar Lint en el proyecto, agregue otro paso de compilación en el *shell de Ejecución* :

   /home/user/gradle/gradle-2.14.1/bin/gradle lint

Ahora su sistema finalmente está configurado para construir proyectos de Android usando Jenkins. Esta configuración hace que su vida sea mucho más fácil para liberar compilaciones a equipos de control de calidad y UAT.

PD: Ya que Jenkins es un usuario diferente en su máquina de Ubuntu, debe otorgarle derechos para crear carpetas en su área de trabajo ejecutando el siguiente comando:

```
chown -R jenkins .git
```

Lea Configuración de Jenkins CI para proyectos de Android en línea:
https://riptutorial.com/es/android/topic/7830/configuracion-de-jenkins-ci-para-proyectos-de-android

Capítulo 64: Construyendo aplicaciones compatibles hacia atrás

Examples

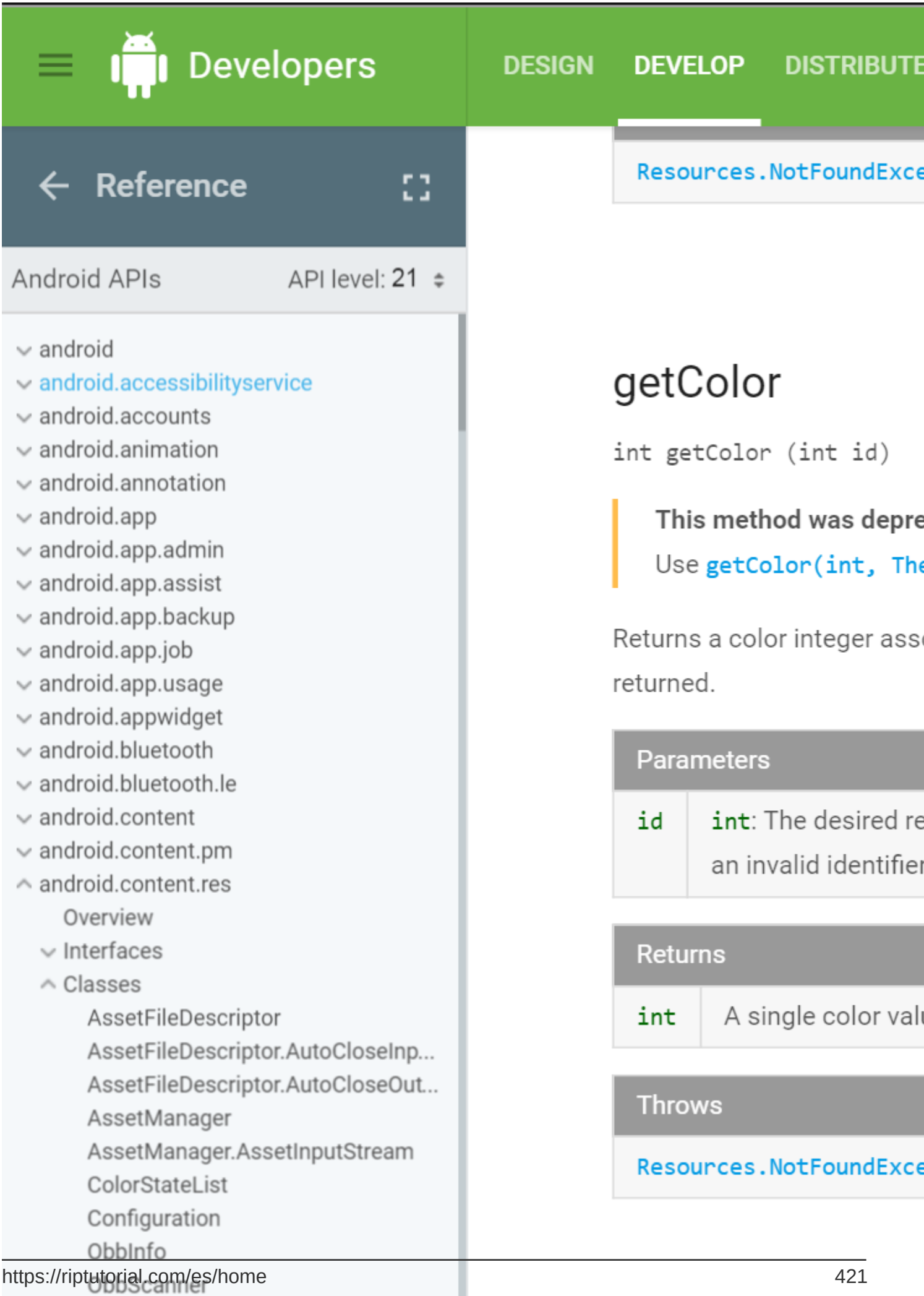
Cómo manejar API en desuso

Es poco probable que un desarrollador no se encuentre con una API en desuso durante un proceso de desarrollo. Un elemento de programa desaprobado es uno que los programadores no deben utilizar, generalmente porque es peligroso o porque existe una mejor alternativa. Los compiladores y analizadores (como LINT) advierten cuando un elemento de programa en desuso se utiliza o se anula en un código no en desuso.

Una API en desuso generalmente se identifica en Android Studio utilizando un tachado. En el siguiente ejemplo, el método `.getColor(int id)` está en desuso:

```
getResources().~~getColor~~(R.color.colorAccent));
```

Si es posible, se recomienda a los desarrolladores que utilicen API y elementos alternativos. Es posible verificar la compatibilidad hacia atrás de una biblioteca visitando la documentación de Android para la biblioteca y verificando la sección "Agregado en el nivel de API x":


Developers
DESIGN
DEVELOP
DISTRIBUTE
Reference
Android APIs
API level: 21
android
android.accessibilityservice
android.accounts
android.animation
android.annotation
android.app
android.app.admin
android.app.assist
android.app.backup
android.app.job
android.app.usage
android.appwidget
android.bluetooth
android.bluetooth.le
android.content
android.content.pm
android.content.res
Overview
Interfaces
Classes
AssetFileDescriptor
AssetFileDescriptor.AutoCloseInp...
AssetFileDescriptor.AutoCloseOut...
AssetManager
AssetManager.AssetInputStream
ColorStateList
Configuration
ObbInfo
ObbScanner
Resources.NotFoundExce
getColor
int getColor (int id)
This method was depre
Use getColor(int, The
Returns a color integer ass
returned.
Parameters
id
int: The desired re
an invalid identifier
Returns
int
A single color valu
Throws
Resources.NotFoundExce

https://riptutorial.com/es/android/topic/4291/construyendo-aplicaciones-compatibles-hacia-atras

Capítulo 65: Contador regresivo

Parámetros

Parámetro	Detalles
`long millisInFuture`	La duración total a la que se ejecutará el temporizador, también conocido como hasta qué punto en el futuro desea que finalice el temporizador. En milisegundos.
`long countDownInterval`	El intervalo en el que desea recibir actualizaciones de temporizador. En milisegundos.
`long millisUntilFinished`	Un parámetro proporcionado en `onTick()` que indica cuánto tiempo ha permanecido el CountDownTimer. En milisegundos

Observaciones

CountDownTimer es una clase bastante magra, que hace una cosa muy bien. Como solo puede iniciar / cancelar un CountDownTimer, debe implementar la funcionalidad de pausa / reanudación como se muestra en el segundo ejemplo. Para una funcionalidad más compleja, o para especificar un temporizador que debe ejecutarse indefinidamente, use el objeto Timer .

Examples

Creando un simple temporizador de cuenta regresiva

CountDownTimer es útil para realizar repetidamente una acción en un intervalo estable durante un tiempo determinado. En este ejemplo, actualizaremos una vista de texto cada segundo durante 30 segundos para indicar cuánto tiempo queda. Luego, cuando el temporizador finalice, configuraremos TextView para que diga "Listo".

```
TextView textView = (TextView)findViewById(R.id.text_view);

CountDownTimer countDownTimer = new CountDownTimer(30000, 1000) {
    public void onTick(long millisUntilFinished) {
        textView.setText(String.format(Locale.getDefault(), "%d sec.", millisUntilFinished /
1000L));
    }

    public void onFinish() {
        textView.setText("Done.");
    }
}.start();
```

Un ejemplo más complejo

En este ejemplo, haremos una pausa / reanudaremos el CountDownTimer basado en el ciclo de vida de la actividad.

```
private static final long TIMER_DURATION = 60000L;
private static final long TIMER_INTERVAL = 1000L;

private CountDownTimer mCountDownTimer;
private TextView textView;

private long mTimeRemaining;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    textView = (TextView)findViewById(R.id.text_view); // Define in xml layout.

    mCountDownTimer = new CountDownTimer(TIMER_DURATION, TIMER_INTERVAL) {

        @Override
        public void onTick(long millisUntilFinished) {
            textView.setText(String.format(Locale.getDefault(), "%d sec.", millisUntilFinished
/ 1000L));
            mTimeRemaining = millisUntilFinished; // Saving timeRemaining in Activity for
pause/resume of CountDownTimer.
        }

        @Override
        public void onFinish() {
            textView.setText("Done.");
        }
    }.start();
}


@Override
protected void onResume() {
    super.onResume();

    if (mCountDownTimer == null) { // Timer was paused, re-create with saved time.
        mCountDownTimer = new CountDownTimer(timeRemaining, INTERVAL) {
            @Override
            public void onTick(long millisUntilFinished) {
                textView.setText(String.format(Locale.getDefault(), "%d sec.",
millisUntilFinished / 1000L));
                timeRemaining = millisUntilFinished;
            }

            @Override
            public void onFinish() {
                textView.setText("Done.");
            }
        }.start();
    }
}

@Override
protected void onPause() {
    super.onPause();
```

```
    mCountDownTimer.cancel();
    mCountDownTimer = null;
}
```

Lea Contador regresivo en línea: https://riptutorial.com/es/android/topic/6063/contador-regresivo

Capítulo 66: Contexto

Introducción

Según la documentación de Google: "Interfaz con la información global sobre el entorno de una aplicación. Permite el acceso a clases y recursos específicos de la aplicación, así como llamadas ascendentes para operaciones a nivel de la aplicación, como actividades de lanzamiento, difusión y recepción de intentos, etc."

En pocas palabras, el contexto es el estado actual de su aplicación. Le permite proporcionar información a los objetos para que puedan estar al tanto de lo que está sucediendo en otras partes de su aplicación.

Sintaxis

- `getApplicationContext()`
- `getBaseContext()`
- `getContext()`
- `this`

Observaciones

Esta página de StackOverflow tiene varias explicaciones completas y bien escritas del concepto de contexto:

¿Qué es el contexto?

Examples

Ejemplos básicos

Uso estándar en la actividad:

```
Context context = getApplicationContext();
```

Uso estándar en Fragmento:

```
Context context = getActivity().getApplicationContext();
```

`this` (cuando se encuentra en una clase que se extiende desde Contexto, como las clases Aplicación, Actividad, Servicio e IntentService)

```
TextView textView = new TextView(this);
```

otro `this` ejemplo:

```
Intent intent = new Intent(this, MainActivity.class);
startActivity(intent);
```

Lea Contexto en línea: https://riptutorial.com/es/android/topic/9774/contexto

Capítulo 67: Conversión de voz a texto

Examples

Discurso a texto con el diálogo predeterminado de solicitud de Google

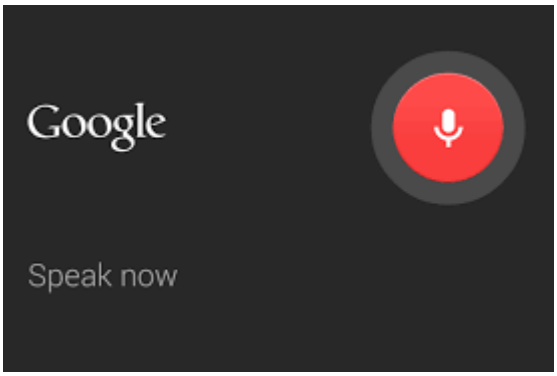
Activar la traducción de voz a texto.

```
private void startListening() {

     //Intent to listen to user vocal input and return result in same activity
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

   //Use a language model based on free-form speech recognition.
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());

    //Message to display in dialog box
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT,
            getString(R.string.speech_to_text_info));
    try {
        startActivityForResult(intent, REQ_CODE_SPEECH_INPUT);
    } catch (ActivityNotFoundException a) {
        Toast.makeText(getApplicationContext(),
                getString(R.string.speech_not_supported),
                Toast.LENGTH_SHORT).show();
    }
}
```

Obtener resultados traducidos en onActivityResult

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    switch (requestCode) {
    case REQ_CODE_SPEECH_INPUT: {
        if (resultCode == RESULT_OK && null != data) {

            ArrayList<String> result = data
                    .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
            txtSpeechInput.setText(result.get(0));
        }
        break;
    }

    }
}
```

Salida



Discurso a texto sin diálogo

El siguiente código se puede usar para activar la traducción de voz a texto sin mostrar un cuadro de diálogo:

```
public void startListeningWithoutDialog() {
    // Intent to listen to user vocal input and return the result to the same activity.
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

    // Use a language model based on free-form speech recognition.
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
    intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 5);
    intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,
            appContext.getPackageName());

    // Add custom listeners.
    CustomRecognitionListener listener = new CustomRecognitionListener();
    SpeechRecognizer sr = SpeechRecognizer.createSpeechRecognizer(appContext);
    sr.setRecognitionListener(listener);
    sr.startListening(intent);
}
```

La clase de escucha personalizada `CustomRecognitionListener` utilizada en el código anterior se implementa de la siguiente manera:

```
class CustomRecognitionListener implements RecognitionListener {
    private static final String TAG = "RecognitionListener";

    public void onReadyForSpeech(Bundle params) {
        Log.d(TAG, "onReadyForSpeech");
    }

    public void onBeginningOfSpeech() {
        Log.d(TAG, "onBeginningOfSpeech");
    }

    public void onRmsChanged(float rmsdB) {
        Log.d(TAG, "onRmsChanged");
    }

    public void onBufferReceived(byte[] buffer) {
        Log.d(TAG, "onBufferReceived");
    }
```

```
    public void onEndOfSpeech() {
        Log.d(TAG, "onEndofSpeech");
    }

    public void onError(int error) {
        Log.e(TAG, "error " + error);

        conversionCallaback.onErrorOccured(TranslatorUtil.getErrorText(error));
    }

    public void onResults(Bundle results) {
        ArrayList<String> result = data
                .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
        txtSpeechInput.setText(result.get(0));
    }

    public void onPartialResults(Bundle partialResults) {
        Log.d(TAG, "onPartialResults");
    }

    public void onEvent(int eventType, Bundle params) {
        Log.d(TAG, "onEvent " + eventType);
    }
}
```

Lea Conversión de voz a texto en línea: https://riptutorial.com/es/android/topic/6252/conversion-de-voz-a-texto

Capítulo 68: Convertir cadena vietnamita a la cadena inglesa Android

Examples

ejemplo:

```
String myStr = convert("Lê Minh Thoại là người Việt Nam");
```

convertido:

```
"Le Minh Thoai la nguoi Viet Nam"
```

Chuyển chuỗi Tiếng Việt thành chuỗi không dấu

```
  public static String convert(String str) {
    str = str.replaceAll("à|á|ạ|ả|ã|â|ầ|ấ|ậ|ẩ|ẫ|ă|ằ|ắ|ặ|ẳ|ẵ", "a");
    str = str.replaceAll("è|é|ẹ|ẻ|ẽ|ê|ề|ế|ệ|ể|ễ", "e");
    str = str.replaceAll("ì|í|ị|ỉ|ĩ", "i");
    str = str.replaceAll("ò|ó|ọ|ỏ|õ|ô|ồ|ố|ộ|ổ|ỗ|ơ|ờ|ớ|ợ|ở|ỡ", "o");
    str = str.replaceAll("ù|ú|ụ|ủ|ũ|ư|ừ|ứ|ự|ử|ữ", "u");
    str = str.replaceAll("ỳ|ý|ỵ|ỷ|ỹ", "y");
    str = str.replaceAll("đ", "d");

    str = str.replaceAll("À|Á|Ạ|Ả|Ã|Â|Ầ|Ấ|Ậ|Ẩ|Ẫ|Ă|Ằ|Ắ|Ặ|Ẳ|Ẵ", "A");
    str = str.replaceAll("È|É|Ẹ|Ẻ|Ẽ|Ê|Ề|Ế|Ệ|Ể|Ễ", "E");
    str = str.replaceAll("Ì|Í|Ị|Ỉ|Ĩ", "I");
    str = str.replaceAll("Ò|Ó|Ọ|Ỏ|Õ|Ô|Ồ|Ố|Ộ|Ổ|Ỗ|Ơ|Ờ|Ớ|Ợ|Ở|Ỡ", "O");
    str = str.replaceAll("Ù|Ú|Ụ|Ủ|Ũ|Ư|Ừ|Ứ|Ự|Ử|Ữ", "U");
    str = str.replaceAll("Ỳ|Ý|Ỵ|Ỷ|Ỹ", "Y");
    str = str.replaceAll("Đ", "D");
    return str;
}
```

Lea Convertir cadena vietnamita a la cadena inglesa Android en línea: https://riptutorial.com/es/android/topic/10946/convertir-cadena-vietnamita-a-la-cadena-inglesa-android

Capítulo 69: Coordinador de Aula y Comportamientos

Introducción

El CoordinatorLayout es un FrameLayout de gran potencia y el objetivo de este ViewGroup es coordinar las vistas que están dentro de él.

El atractivo principal de CoordinatorLayout es su capacidad para coordinar las animaciones y transiciones de las vistas dentro del propio archivo XML.

CoordinatorLayout está destinado a dos casos de uso principales:

: Como una decoración de aplicación de nivel superior o diseño de cromo

: Como contenedor para una interacción específica con una o más vistas secundarias

Observaciones

El `CoordinatorLayout` es un contenedor que extiende el `FrameLayout` .
Al adjuntar un comportamiento de `CoordinatorLayout.Behavior` a un hijo directo de `CoordinatorLayout` , podrá interceptar eventos táctiles, inserciones de ventanas, medidas, diseño y desplazamiento anidado.

Al especificar `Behaviors` para vistas secundarias de un `CoordinatorLayout` , puede proporcionar muchas interacciones diferentes dentro de un solo padre y esas vistas también pueden interactuar entre sí. Las clases de vista pueden especificar un comportamiento predeterminado cuando se usan como elementos secundarios de un `CoordinatorLayout` mediante la anotación `DefaultBehavior` .

Examples

Creando un comportamiento simple

Para crear un `Behavior` simplemente extienda la clase `CoordinatorLayout.Behavior` .

Extender el CoordinatorLayout.Behavior

Ejemplo:

```
public class MyBehavior<V extends View> extends CoordinatorLayout.Behavior<V> {

      /**
       * Default constructor.
```

```
     */
    public MyBehavior() {
    }

    /**
     * Default constructor for inflating a MyBehavior from layout.
     *
     * @param context The {@link Context}.
     * @param attrs The {@link AttributeSet}.
     */
    public MyBehavior(Context context, AttributeSet attrs) {
      super(context, attrs);
    }
}
```

Este comportamiento debe adjuntarse a una vista secundaria de un `CoordinatorLayout` a ser llamado.

Adjuntar un comportamiento programáticamente

```
MyBehavior myBehavior = new MyBehavior();
CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams)
view.getLayoutParams();
params.setBehavior(myBehavior);
```

Adjuntar un comportamiento en XML

Puede usar el atributo `layout_behavior` para adjuntar el comportamiento en XML:

```
<View
  android:layout_height="...."
  android:layout_width="...."
  app:layout_behavior=".MyBehavior" />
```

Adjuntar un comportamiento automáticamente

Si está trabajando con una vista personalizada, puede adjuntar el comportamiento utilizando la anotación `@CoordinatorLayout.DefaultBehavior` :

```
@CoordinatorLayout.DefaultBehavior(MyBehavior.class)
public class MyView extends ..... {

}
```

Usando el comportamiento de SwipeDismiss

`SwipeDismissBehavior` funciona en cualquier Vista e implementa la funcionalidad de deslizar para descartar en nuestros diseños con un `CoordinatorLayout` .

Solo usa:

```
        final SwipeDismissBehavior<MyView> swipe = new SwipeDismissBehavior();

        //Sets the swipe direction for this behavior.
        swipe.setSwipeDirection(
            SwipeDismissBehavior.SWIPE_DIRECTION_ANY);

        //Set the listener to be used when a dismiss event occurs
        swipe.setListener(
            new SwipeDismissBehavior.OnDismissListener() {
            @Override public void onDismiss(View view) {
                //......
            }

            @Override
            public void onDragStateChanged(int state) {
                //......
            }
        });

        //Attach the SwipeDismissBehavior to a view
        LayoutParams coordinatorParams =
            (LayoutParams) mView.getLayoutParams();
        coordinatorParams.setBehavior(swipe);
```

Crear dependencias entre vistas

Puede usar `CoordinatorLayout.Behavior` para crear dependencias entre vistas. Puede anclar una `View` a otra `View` mediante:

- utilizando el atributo `layout_anchor` .
- creando un `Behavior` personalizado e implementando el método `layoutDependsOn` devolviendo `true` .

Por ejemplo, para crear un `Behavior` para mover un `ImageView` cuando se mueve otro (barra de herramientas de ejemplo), realice los siguientes pasos:

- Crea el comportamiento personalizado :

  ```
  public class MyBehavior extends CoordinatorLayout.Behavior<ImageView> {...}
  ```

- Reemplace el método `layoutDependsOn` devolviendo `true` . Este método se llama cada vez que se produce un cambio en el diseño:

  ```
  @Override
  public boolean layoutDependsOn(CoordinatorLayout parent,
          ImageView child, View dependency) {
  ```

```
    // Returns true to add a dependency.
    return dependency instanceof Toolbar;
}
```

- Cuando el método `layoutDependsOn` devuelve `true` , se llama al método `onDependentViewChanged` :

```
@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, ImageView child, View
dependency) {
    // Implement here animations, translations, or movements; always related to the
provided dependency.
    float translationY = Math.min(0, dependency.getTranslationY() -
dependency.getHeight());
    child.setTranslationY(translationY);
}
```

Lea Coordinador de Aula y Comportamientos en línea:
https://riptutorial.com/es/android/topic/5714/coordinador-de-aula-y-comportamientos

Capítulo 70: Cosas de Android

Examples

Controlando un Servo Motor

Este ejemplo asume que tiene un servo con las siguientes características, que son típicas:

- Movimiento entre 0 y 180 grados.
- periodo de pulso de 20 ms
- Longitud mínima de pulso de 0.5 ms
- Longitud máxima de pulso de 2,5 ms.

Debe comprobar si esos valores coinciden con su hardware, ya que forzarlo a salir de su rango operativo especificado puede dañar el servo. Un servo dañado a su vez tiene el potencial de dañar su dispositivo Android Things. La clase de `ServoController` ejemplo consta de dos métodos, `setup()` y `setPosition()` :

```
public class ServoController {
    private double periodMs, maxTimeMs, minTimeMs;
    private Pwm pin;

    public void setup(String pinName) throws IOException {
        periodMs  = 20;
        maxTimeMs = 2.5;
        minTimeMs = 0.5;

        PeripheralManagerService service = new PeripheralManagerService();
        pin = service.openPwm(pinName);

        pin.setPwmFrequencyHz(1000.0d / periodMs);
        setPosition(90);
        pin.setEnabled(true);
    }

    public void setPosition(double degrees) {
        double pulseLengthMs = (degrees / 180.0 * (maxTimeMs - minTimeMs)) + minTimeMs;

        if (pulseLengthMs < minTimeMs) {
            pulseLengthMs = minTimeMs;
        } else if (pulseLengthMs > maxTimeMs) {
            pulseLengthMs = maxTimeMs;
        }

        double dutyCycle = pulseLengthMs / periodMs * 100.0;

        Log.i(TAG, "Duty cycle = " + dutyCycle + " pulse length = " + pulseLengthMs);

        try {
            pin.setPwmDutyCycle(dutyCycle);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
}
```

Puede descubrir nombres de pin que admiten PWM en su dispositivo de la siguiente manera:

```
PeripheralManagerService service = new PeripheralManagerService();

for (String pinName : service.getPwmList() ) {
    Log.i("ServoControlled","Pwm pin found: " + pinName);
}
```

Para hacer que su servo oscile por siempre entre 80 grados y 100 grados, simplemente puede usar el siguiente código:

```
final ServoController servoController = new ServoController(pinName);

Thread th = new Thread(new Runnable() {
    @Override
    public void run() {
        while (true) {
            try {
                servoController.setPosition(80);
                Thread.sleep(500);
                servoController.setPosition(100);
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
});
th.start();
```

Puede compilar e implementar todo el código anterior sin en realidad conectar ningún servomotor al dispositivo informático. Para el cableado, consulte la tabla de pines de su dispositivo informático (por ejemplo, una tabla de pines de la Raspberry Pi 3 está disponible aquí).

Entonces necesitas conectar tu servo a Vcc, Gnd y señal.

Lea Cosas de Android en línea: https://riptutorial.com/es/android/topic/8938/cosas-de-android

Capítulo 71: Crea ROMs personalizadas de Android

Examples

¡Preparando su máquina para construir!

Antes de poder construir cualquier cosa, se requiere que prepare su máquina para la construcción. Para esto necesitas instalar muchas librerías y módulos. La distribución de Linux más recomendada es Ubuntu, por lo que este ejemplo se centrará en instalar todo lo que se necesita en Ubuntu.

Instalando Java

Primero, agregue el siguiente Personal Package Archive (PPA): `sudo apt-add-repository ppa:openjdk-r/ppa` .

Luego, actualice las fuentes ejecutando: `sudo apt-get update` .

Instalando Dependencias Adicionales

Todas las dependencias adicionales requeridas se pueden instalar con el siguiente comando:

```
sudo apt-get install git-core python gnupg flex bison gperf libsdl1.2-dev libesd0-dev
libwxgtk2.8-dev squashfs-tools build-essential zip curl libncurses5-dev zlib1g-dev openjdk-8-
jre openjdk-8-jdk pngcrush schedtool libxml2 libxml2-utils xsltproc lzop libc6-dev schedtool
g++-multilib lib32z1-dev lib32ncurses5-dev gcc-multilib liblz4-* pngquant ncurses-dev texinfo
gcc gperf patch libtool automake g++ gawk subversion expat libexpat1-dev python-all-dev
binutils-static bc libcloog-isl-dev libcap-dev autoconf libgmp-dev build-essential gcc-
multilib g++-multilib pkg-config libmpc-dev libmpfr-dev lzma* liblzma* w3m android-tools-adb
maven ncftp figlet
```

Preparando el sistema para el desarrollo.

Ahora que todas las dependencias están instaladas, preparemos el sistema para el desarrollo ejecutando:

```
sudo curl --create-dirs -L -o /etc/udev/rules.d/51-android.rules -O -L
https://raw.githubusercontent.com/snowdream/51-android/master/51-android.rules
sudo chmod 644   /etc/udev/rules.d/51-android.rules
sudo chown root /etc/udev/rules.d/51-android.rules
sudo service udev restart
adb kill-server
```

```
sudo killall adb
```

Finalmente, configuremos el caché y el repositorio con los siguientes comandos:

```
sudo install utils/repo /usr/bin/
sudo install utils/ccache /usr/bin/
```

Tenga en cuenta: También podemos lograr esta configuración ejecutando los scripts automáticos creados por Akhil Narang (*akhilnarang*), uno de los mantenedores del sistema operativo Resurrection Remix . Estos scripts se pueden encontrar en GitHub .

Lea Crea ROMs personalizadas de Android en línea:
https://riptutorial.com/es/android/topic/9212/crea-roms-personalizadas-de-android

Capítulo 72: Creación de superposición (siempre en la parte superior) de Windows

Examples

Superposición de ventanas emergentes

Para poder colocar su vista en la parte superior de cada aplicación, debe asignar su vista al gestor de ventanas correspondiente. Para eso necesita el permiso de alerta del sistema, que puede solicitarse agregando la siguiente línea a su archivo de manifiesto:

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
```

Nota: si su aplicación se destruye, su vista se eliminará del administrador de ventanas. Por lo tanto, es mejor crear la vista y asignarla al administrador de ventanas mediante un servicio en primer plano.

Asignando una vista al WindowManager

Puede recuperar una instancia del administrador de ventanas de la siguiente manera:

```
WindowManager mWindowManager = (WindowManager)
mContext.getSystemService(Context.WINDOW_SERVICE);
```

Para definir la posición de su vista, debe crear algunos parámetros de diseño de la siguiente manera:

```
WindowManager.LayoutParams mLayoutParams = new WindowManager.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT,
        WindowManager.LayoutParams.TYPE_PHONE,
        WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON,
        PixelFormat.TRANSLUCENT);
mLayoutParams.gravity = Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL;
```

Ahora, puede asignar su vista junto con los parámetros de diseño creados a la instancia del administrador de ventanas de la siguiente manera:

```
mWindowManager.addView(yourView, mLayoutParams);
```

Voila! Su vista se ha colocado con éxito sobre todas las demás aplicaciones.

Nota: la vista no se colocará encima del protector del teclado.

Concesión del permiso SYSTEM_ALERT_WINDOW en Android 6.0 y superior

Desde Android 6.0 este permiso debe otorgarse dinámicamente,

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
```

Lanzar debajo del permiso denegado error en 6.0,

```
Caused by: android.view.WindowManager$BadTokenException: Unable to add window
android.view.ViewRootImpl$W@86fb55b -- permission denied for this window type
```

Solución: -

Solicitando permiso de superposición como a continuación,

```
if(!Settings.canDrawOverlays(this)){
    // ask for setting
     Intent intent = new Intent(Settings.ACTION_MANAGE_OVERLAY_PERMISSION,
     Uri.parse("package:" + getPackageName()));
     startActivityForResult(intent, REQUEST_OVERLAY_PERMISSION);
}
```

Compruebe el resultado,

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_OVERLAY_PERMISSION) {
        if (Settings.canDrawOverlays(this)) {
            // permission granted...
        }else{
            // permission not granted...
        }
    }
}
```

Lea Creación de superposición (siempre en la parte superior) de Windows en línea: https://riptutorial.com/es/android/topic/6214/creacion-de-superposicion--siempre-en-la-parte-superior--de-windows

Capítulo 73: Creación de vistas personalizadas

Examples

Creación de vistas personalizadas

Si necesita una vista completamente personalizada, tendrá que hacer una subclase de `View` (la superclase de todas las vistas de Android) y proporcionar los `onMeasure(...)` tamaño personalizado (`onMeasure(...)`) y drawing (`onDraw(...)`):

1. **Crea tu esqueleto de vista personalizada:** esto es básicamente el mismo para cada vista personalizada. Aquí creamos el esqueleto para una vista personalizada que puede dibujar un emoticono, llamado `SmileyView` :

```
public class SmileyView extends View {
    private Paint mCirclePaint;
    private Paint mEyeAndMouthPaint;

    private float mCenterX;
    private float mCenterY;
    private float mRadius;
    private RectF mArcBounds = new RectF();

    public SmileyView(Context context) {
        this(context, null, 0);
    }

    public SmileyView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        initPaints();
    }

    private void initPaints() {/* ... */}

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {/* ... */}

    @Override
    protected void onDraw(Canvas canvas) {/* ... */}
}
```

2. **Inicialice sus pinturas:** los objetos de `Paint` son los pinceles de su lienzo virtual que definen cómo se representan los objetos geométricos (por ejemplo, color, estilo de relleno y trazo, etc.). Aquí creamos dos `Paint` s, una pintura llenado amarillo para el círculo y una pintura de trazo negro para los ojos y la boca:

```
private void initPaints() {
    mCirclePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    mCirclePaint.setStyle(Paint.Style.FILL);
    mCirclePaint.setColor(Color.YELLOW);
    mEyeAndMouthPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    mEyeAndMouthPaint.setStyle(Paint.Style.STROKE);
    mEyeAndMouthPaint.setStrokeWidth(16 * getResources().getDisplayMetrics().density);
    mEyeAndMouthPaint.setStrokeCap(Paint.Cap.ROUND);
    mEyeAndMouthPaint.setColor(Color.BLACK);
}
```

3. **Implemente su propio `onMeasure(...)` :** esto es necesario para que los diseños principales (por ejemplo, `FrameLayout`) puedan alinear correctamente su vista personalizada. Proporciona un conjunto de `measureSpecs` de `measureSpecs` que puede usar para determinar la altura y el ancho de su vista. Aquí creamos un cuadrado asegurándonos de que la altura y el ancho son iguales:

```
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    int w = MeasureSpec.getSize(widthMeasureSpec);
    int h = MeasureSpec.getSize(heightMeasureSpec);

    int size = Math.min(w, h);
    setMeasuredDimension(size, size);
}
```

   Tenga en cuenta que `onMeasure(...)` debe contener al menos una llamada a `setMeasuredDimension(..)` o, de lo contrario, su vista personalizada se bloqueará con una `IllegalStateException` .

4. **Implemente su propio `onSizeChanged(...)` :** esto le permite capturar la altura y el ancho actuales de su vista personalizada para ajustar adecuadamente su código de representación. Aquí solo calculamos nuestro centro y nuestro radio:

```
@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
    mCenterX = w / 2f;
    mCenterY = h / 2f;
    mRadius = Math.min(w, h) / 2f;
}
```

5. **Implemente su propio `onDraw(...)` :** aquí es donde implementa la representación real de su vista. Proporciona un objeto `Canvas` que puede dibujar (consulte la documentación oficial de `Canvas` para conocer todos los métodos de dibujo disponibles).

```
@Override
protected void onDraw(Canvas canvas) {
    // draw face
    canvas.drawCircle(mCenterX, mCenterY, mRadius, mCirclePaint);
    // draw eyes
    float eyeRadius = mRadius / 5f;
    float eyeOffsetX = mRadius / 3f;
    float eyeOffsetY = mRadius / 3f;
```

```
    canvas.drawCircle(mCenterX - eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,
mEyeAndMouthPaint);
    canvas.drawCircle(mCenterX + eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,
mEyeAndMouthPaint);
    // draw mouth
    float mouthInset = mRadius /3f;
    mArcBounds.set(mouthInset, mouthInset, mRadius * 2 - mouthInset, mRadius * 2 -
mouthInset);
    canvas.drawArc(mArcBounds, 45f, 90f, false, mEyeAndMouthPaint);
}
```

6. **Agregue su vista personalizada a un diseño:** la vista personalizada ahora se puede incluir en cualquier archivo de diseño que tenga. Aquí simplemente lo `FrameLayout` dentro de un `FrameLayout` :

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.example.app.SmileyView
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>
```

Tenga en cuenta que se recomienda compilar su proyecto una vez finalizado el código de vista. Sin construirlo, no podrá ver la vista en una pantalla de vista previa en Android Studio.

Después de poner todo junto, debe recibir la siguiente pantalla después de iniciar la actividad que contiene el diseño anterior:



Agregando atributos a las vistas

Las vistas personalizadas también pueden tomar atributos personalizados que pueden usarse en archivos de recursos de diseño de Android. Para agregar atributos a su vista personalizada, debe hacer lo siguiente:

1. **Defina el nombre y el tipo de sus atributos:** esto se hace dentro de `res/values/attrs.xml` (`res/values/attrs.xml` si es necesario). El siguiente archivo define un atributo de color para el color de la cara de nuestro smiley y un atributo de enumeración para la expresión del smiley:

```
<resources>
    <declare-styleable name="SmileyView">
        <attr name="smileyColor" format="color" />
        <attr name="smileyExpression" format="enum">
            <enum name="happy" value="0"/>
            <enum name="sad" value="1"/>
        </attr>
    </declare-styleable>
    <!-- attributes for other views -->
</resources>
```

2. **Use sus atributos dentro de su diseño:** esto se puede hacer dentro de cualquier archivo de diseño que use su vista personalizada. El siguiente archivo de diseño crea una pantalla con un smiley amarillo feliz:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <com.example.app.SmileyView
        android:layout_height="56dp"
        android:layout_width="56dp"
        app:smileyColor="#ffff00"
        app:smileyExpression="happy" />
</FrameLayout>
```

   Consejo: los atributos personalizados no funcionan con las `tools:` prefijo en Android Studio 2.1 y versiones anteriores (y posiblemente en versiones futuras). En este ejemplo, reemplazar la `app:smileyColor` con `tools:smileyColor` resultaría en que `smileyColor` no se establezca durante el tiempo de ejecución ni en el momento del diseño.

3. **Lea sus atributos:** esto se hace dentro del código fuente de su vista personalizada. El siguiente fragmento de `SmileyView` demuestra cómo se pueden extraer los atributos:

```
public class SmileyView extends View {
    // ...

    public SmileyView(Context context) {
        this(context, null);
    }

    public SmileyView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
```

```
    }

    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);

        TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView,
defStyleAttr, 0);
        mFaceColor = a.getColor(R.styleable.SmileyView_smileyColor, Color.TRANSPARENT);
        mFaceExpression = a.getInteger(R.styleable.SmileyView_smileyExpression,
Expression.HAPPY);
        // Important: always recycle the TypedArray
        a.recycle();

        // initPaints(); ...
    }
}
```

4. **(Opcional) Agregar estilo predeterminado:** esto se hace agregando un estilo con los valores predeterminados y cargándolo dentro de su vista personalizada. El siguiente estilo de emoticono predeterminado representa un color amarillo feliz:

```
<!-- styles.xml -->
<style name="DefaultSmileyStyle">
    <item name="smileyColor">#ffff00</item>
    <item name="smileyExpression">happy</item>
</style>
```

   Que se aplica en nuestro `SmileyView` al agregarlo como el último parámetro de la llamada para obtener `obtainStyledAttributes` de `obtainStyledAttributes` (vea el código en el paso 3):

```
TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView,
defStyleAttr, R.style.DefaultSmileyViewStyle);
```

   Tenga en cuenta que cualquier valor de atributo establecido en el archivo de diseño inflado (ver código en el paso 2) anulará los valores correspondientes del estilo predeterminado.

5. **(Opcional) Proporcione estilos dentro de los temas:** esto se hace agregando un nuevo atributo de referencia de estilo que puede usarse dentro de sus temas y proporcionando un estilo para ese atributo. Aquí simplemente `smileyStyle` nuestro atributo de referencia `smileyStyle` :

```
<!-- attrs.xml -->
<attr name="smileyStyle" format="reference" />
```

   A continuación, proporcionamos un estilo en el tema de nuestra aplicación (aquí solo reutilizamos el estilo predeterminado del paso 4):

```
<!-- themes.xml -->
<style name="AppTheme" parent="AppBaseTheme">
    <item name="smileyStyle">@style/DefaultSmileyStyle</item>
</style>
```

Creando una vista compuesta

Una **vista compuesto** es una costumbre `ViewGroup` que se trata como una única vista por el código del programa circundante. Tal ViewGroup puede ser realmente útil en diseño similar a DDD , ya que puede corresponder a un agregado, en este ejemplo, un Contacto. Se puede reutilizar en cualquier lugar donde se muestre el contacto.

Esto significa que el código del controlador circundante, una Actividad, un Fragmento o un Adaptador, simplemente puede pasar el objeto de datos a la vista sin separarlo en una serie de widgets de IU diferentes.

Esto facilita la reutilización del código y permite un mejor diseño de acuerdo con los principios de SOLID .

El diseño XML

Esto suele ser donde empiezas. Tiene un bit de XML existente que reutiliza, tal vez como `<include/>` . Extráigalo en un archivo XML separado y envuelva la etiqueta raíz en un elemento `<merge>` :

```
<?xml version="1.0" encoding="utf-8"?>
<merge xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

        <ImageView
            android:id="@+id/photo"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:layout_alignParentRight="true" />

        <TextView
            android:id="@+id/name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_toLeftOf="@id/photo" />

        <TextView
            android:id="@+id/phone_number"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/name"
            android:layout_toLeftOf="@id/photo" />
</merge>
```

Este archivo XML sigue funcionando perfectamente en el editor de diseño en Android Studio. Puedes tratarlo como cualquier otro diseño.

El compuesto ViewGroup

Una vez que tenga el archivo XML, cree el grupo de vista personalizado.

```
import android.annotation.TargetApi;
```

```
import android.content.Context;
import android.os.Build;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.RelativeLayout;
import android.widget.ImageView;
import android.widget.TextView;

import myapp.R;

/**
 * A compound view to show contacts.
 *
 * This class can be put into an XML layout or instantiated programmatically, it
 * will work correctly either way.
 */
public class ContactView extends RelativeLayout {

    // This class extends RelativeLayout because that comes with an automatic
    // (MATCH_PARENT, MATCH_PARENT) layout for its child item. You can extend
    // the raw android.view.ViewGroup class if you want more control. See the
    // note in the layout XML why you wouldn't want to extend a complex view
    // such as RelativeLayout.

    // 1. Implement superclass constructors.
    public ContactView(Context context) {
        super(context);
        init(context, null);
    }

    // two extra constructors left out to keep the example shorter

    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public ContactView(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes)
{
        super(context, attrs, defStyleAttr, defStyleRes);
        init(context, attrs);
    }

    // 2. Initialize the view by inflating an XML using `this` as parent
    private TextView mName;
    private TextView mPhoneNumber;
    private ImageView mPhoto;

    private void init(Context context, AttributeSet attrs) {
        LayoutInflater.from(context).inflate(R.layout.contact_view, this, true);
        mName = (TextView) findViewById(R.id.name);
        mPhoneNumber = (TextView) findViewById(R.id.phone_number);
        mPhoto = (ImageView) findViewById(R.id.photo);
    }

    // 3. Define a setter that's expressed in your domain model. This is what the example is
    //    all about. All controller code can just invoke this setter instead of fiddling with
    //    lots of strings, visibility options, colors, animations, etc. If you don't use a
    //    custom view, this code will usually end up in a static helper method (bad) or copies

    //    of this code will be copy-pasted all over the place (worse).
    public void setContact(Contact contact) {
        mName.setText(contact.getName());
        mPhoneNumber.setText(contact.getPhoneNumber());
```

```
        if (contact.hasPhoto()) {
            mPhoto.setVisibility(View.VISIBLE);
            mPhoto.setImageBitmap(contact.getPhoto());
        } else {
            mPhoto.setVisibility(View.GONE);
        }
    }
}
```

El método `init(Context, AttributeSet)` es donde leería cualquier atributo XML personalizado tal como se explica en Agregar atributos a las vistas .

Con estas piezas en su lugar, puedes usarlo en tu aplicación.

Uso en XML

Aquí hay un ejemplo de `fragment_contact_info.xml` que ilustra cómo pondría un único ContactView encima de una lista de mensajes:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <!-- The compound view becomes like any other view XML element -->
    <myapp.ContactView
        android:id="@+id/contact"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/message_list"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"/>

</LinearLayout>
```

Uso en Código

Aquí hay un ejemplo de `RecyclerView.Adapter` que muestra una lista de contactos. Este ejemplo ilustra cuánto más limpio está el código del controlador cuando está completamente libre de manipulación de vistas.

```
package myapp;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.ViewGroup;

public class ContactsAdapter extends RecyclerView.Adapter<ContactsViewHolder> {

    private final Context context;

    public ContactsAdapter(final Context context) {
        this.context = context;
```

```
    }

    @Override
    public ContactsViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        ContactView v = new ContactView(context); // <--- this
        return new ContactsViewHolder(v);
    }

    @Override
    public void onBindViewHolder(ContactsViewHolder holder, int position) {
        Contact contact = this.getItem(position);
        holder.setContact(contact);  // <--- this
    }

    static class ContactsViewHolder extends RecyclerView.ViewHolder {

        public ContactsViewHolder(ContactView itemView) {
            super(itemView);
        }

        public void setContact(Contact contact) {
            ((ContactView) itemView).setContact(contact); // <--- this
        }
    }
}
```

Consejos de rendimiento de CustomView

No asignar nuevos objetos en **onDraw**

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = new Paint(); //Do not allocate here
}
```

En lugar de dibujar dibujables en lienzo ...

```
drawable.setBounds(boundsRect);

drawable.draw(canvas);
```

Use un mapa de bits para un dibujo más rápido:

```
canvas.drawBitmap(bitmap, srcRect, boundsRect, paint);
```

No vuelva a dibujar la vista completa para actualizar solo una pequeña parte de ella. En su lugar, vuelva a dibujar la parte específica de la vista.

```
invalidate(boundToBeRefreshed);
```

Si su vista está haciendo una animación continua, por ejemplo, una `onStop()` muestra cada segundo, al menos detenga la animación en `onStop()` de la actividad y comience de nuevo en

`onStart()` de la actividad.

No realice ningún cálculo dentro del método `onDraw` de una vista, en lugar de eso, debe terminar de dibujar antes de llamar a `invalidate()` . Al utilizar esta técnica, puede evitar que el cuadro se caiga en su vista.

Rotaciones

Las operaciones básicas de una vista son traducir, rotar, etc. Casi todos los desarrolladores se han enfrentado a este problema cuando usan mapas de bits o degradados en su vista personalizada. Si la vista va a mostrar una vista girada y el mapa de bits debe girarse en esa vista personalizada, muchos de nosotros pensamos que será caro. Muchos piensan que rotar un mapa de bits es muy costoso porque para hacer eso, es necesario traducir la matriz de píxeles del mapa de bits. Pero la verdad es que no es tan difícil! En lugar de rotar el mapa de bits, ¡simplemente gire el lienzo!

```
// Save the canvas state
int save = canvas.save();
// Rotate the canvas by providing the  center point as pivot and angle
canvas.rotate(pivotX, pivotY, angle);
// Draw whatever you want
// Basically whatever you draw here will be drawn as per the angle you rotated the canvas
canvas.drawBitmap(...);
// Now restore your your canvas to its original state
canvas.restore(save);
// Unless canvas is restored to its original state, further draw will also be rotated.
```

Vista compuesta para SVG / VectorDrawable as drawableRight

El motivo principal para desarrollar esta vista compuesta es que, por debajo de 5.0, los dispositivos no son compatibles con svg en drawable dentro de TextView / EditText. Uno más es pros, podemos establecer `height` y `width` de `drawableRight` dentro `EditText` . Lo he separado de mi proyecto y lo he creado en un módulo separado.

Nombre del módulo: custom_edit_drawable (nombre corto para prefijo-c_d_e)

Se utiliza el prefijo "c_d_e_" para que los recursos del módulo de la aplicación no los anulen por error. Ejemplo: Google utiliza el prefijo "abc" en la biblioteca de soporte.

construir.gradle

```
dependencies {
   compile 'com.android.support:appcompat-v7:25.3.1'
}
```

utilizar AppCompat> = 23

Archivo de diseño: c_e_d_compound_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
        android:id="@+id/edt_search"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:maxLines="1"
        android:paddingEnd="40dp"
        android:paddingLeft="5dp"
        android:paddingRight="40dp"
        android:paddingStart="5dp" />

    <!--make sure you are not using ImageView instead of this-->
    <android.support.v7.widget.AppCompatImageView
        android:id="@+id/drawbleRight_search"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:layout_gravity="right|center_vertical"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp" />
</FrameLayout>
```

Atributos personalizados: attrs.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="EditTextWithDrawable">
        <attr name="c_e_d_drawableRightSVG" format="reference" />
        <attr name="c_e_d_hint" format="string" />
        <attr name="c_e_d_textSize" format="dimension" />
        <attr name="c_e_d_textColor" format="color" />
    </declare-styleable>
</resources>
```

Código: EditTextWithDrawable.java

```
public class EditTextWithDrawable extends FrameLayout {
    public AppCompatImageView mDrawableRight;
    public EditText mEditText;

    public EditTextWithDrawable(Context context) {
        super(context);
        init(null);
    }

    public EditTextWithDrawable(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(attrs);
    }
```

```
    public EditTextWithDrawable(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init(attrs);
    }

    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public EditTextWithDrawable(Context context, AttributeSet attrs, int defStyleAttr, int
defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
        init(attrs);
    }

    private void init(AttributeSet attrs) {
        if (attrs != null && !isInEditMode()) {
            LayoutInflater inflater = (LayoutInflater) getContext()
                    .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            inflater.inflate(R.layout.c_e_d_compound_view, this, true);
            mDrawableRight = (AppCompatImageView) ((FrameLayout) getChildAt(0)).getChildAt(1);
            mEditText = (EditText) ((FrameLayout) getChildAt(0)).getChildAt(0);

            TypedArray attributeArray = getContext().obtainStyledAttributes(
                    attrs,
                    R.styleable.EditTextWithDrawable);

            int drawableRes =
                    attributeArray.getResourceId(
                            R.styleable.EditTextWithDrawable_c_e_d_drawableRightSVG, -1);
            if (drawableRes != -1) {
                mDrawableRight.setImageResource(drawableRes);
            }

            mEditText.setHint(attributeArray.getString(
                    R.styleable.EditTextWithDrawable_c_e_d_hint));
            mEditText.setTextColor(attributeArray.getColor(
                    R.styleable.EditTextWithDrawable_c_e_d_textColor, Color.BLACK));
            int textSize =
attributeArray.getDimensionPixelSize(R.styleable.EditTextWithDrawable_c_e_d_textSize, 15);
            mEditText.setTextSize(TypedValue.COMPLEX_UNIT_PX, textSize);
            android.view.ViewGroup.LayoutParams layoutParams =
mDrawableRight.getLayoutParams();
            layoutParams.width = (textSize * 3) / 2;
            layoutParams.height = (textSize * 3) / 2;
            mDrawableRight.setLayoutParams(layoutParams);

            attributeArray.recycle();
        }
    }
}
```

Ejemplo: Cómo usar la vista superior

Diseño: activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
```

```
    <com.customeditdrawable.AppEditTextWithDrawable
        android:id="@+id/edt_search_emp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:c_e_d_drawableRightSVG="@drawable/ic_svg_search"
        app:c_e_d_hint="@string/hint_search_here"
        app:c_e_d_textColor="@color/text_color_dark_on_light_bg"
        app:c_e_d_textSize="@dimen/text_size_small" />
</LinearLayout>
```

Actividad: MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    EditTextWithDrawable mEditTextWithDrawable;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mEditTextWithDrawable= (EditTextWithDrawable) findViewById(R.id.edt_search_emp);
    }
}
```

Respondiendo a los eventos táctiles

Muchas vistas personalizadas deben aceptar la interacción del usuario en forma de eventos táctiles. Puede obtener acceso a eventos táctiles anulando `onTouchEvent` . Hay una serie de acciones que puedes filtrar. Los principales son

- `ACTION_DOWN` : Esto se activa una vez cuando su dedo toca la vista por primera vez.
- `ACTION_MOVE` : se llama cada vez que su dedo se mueve un poco en la vista. Se llama muchas veces.
- `ACTION_UP` : esta es la última acción a la que se llama cuando levanta el dedo de la pantalla.

Puede agregar el siguiente método a su vista y luego observar la salida del registro cuando toca y mueve su dedo alrededor de su vista.

```
@Override
public boolean onTouchEvent(MotionEvent event) {

    int x = (int) event.getX();
    int y = (int) event.getY();
    int action = event.getAction();

    switch (action) {
        case MotionEvent.ACTION_DOWN:
            Log.i("CustomView", "onTouchEvent: ACTION_DOWN: x = " + x + ", y = " + y);
            break;

        case MotionEvent.ACTION_MOVE:
            Log.i("CustomView", "onTouchEvent: ACTION_MOVE: x = " + x + ", y = " + y);
            break;

        case MotionEvent.ACTION_UP:
```

```
            Log.i("CustomView", "onTouchEvent: ACTION_UP: x = " + x + ", y = " + y);
            break;
    }
    return true;
}
```

Otras lecturas:

- Documentación oficial de Android: Respondiendo a eventos táctiles.

Lea Creación de vistas personalizadas en línea:
https://riptutorial.com/es/android/topic/1446/creacion-de-vistas-personalizadas

Capítulo 74: Creando pantalla de bienvenida

Observaciones

El primer ejemplo (una pantalla de inicio básica) no es la forma más eficiente de manejarlo. Como tal, es la pantalla de inicio básica.

Examples

Una pantalla de bienvenida básica.

Una pantalla de inicio es como cualquier otra actividad, pero puede manejar todas sus necesidades de inicio en segundo plano. Ejemplo:

Manifiesto:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.package"
    android:versionCode="1"
    android:versionName="1.0" >

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".Splash"
            android:label="@string/app_name"
             >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

Ahora nuestra pantalla de inicio será llamada como la primera actividad.

Aquí hay un ejemplo de la pantalla de bienvenida que también maneja algunos elementos críticos de la aplicación:

```
public class Splash extends Activity{

    public final int SPLASH_DISPLAY_LENGTH = 3000;
```

```
    private void checkPermission() {
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.WAKE_LOCK) !=
PackageManager.PERMISSION_GRANTED ||
                ContextCompat.checkSelfPermission(this,Manifest.permission.INTERNET) !=
PackageManager.PERMISSION_GRANTED ||
                ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_NETWORK_STATE) != PackageManager.PERMISSION_GRANTED) {//Can add
more as per requirement


            ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.WAKE_LOCK,
                            Manifest.permission.INTERNET,
                            Manifest.permission.ACCESS_NETWORK_STATE},
                    123);
        }

    }
    @Override
    protected void onCreate(Bundle sis){
        super.onCreate(sis);
        //set the content view. The XML file can contain nothing but an image, such as a logo
or the app icon
        setContentView(R.layout.splash);


        //we want to display the splash screen for a few seconds before it automatically
        //disappears and loads the game. So we create a thread:
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {

                //request permissions. NOTE: Copying this and the manifest will cause the app
to crash as the permissions requested aren't defined in the manifest.
                    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M ) {
                        checkPermission();
                    }
                    String lang = [load or determine the system language and set to default if
it isn't available.]
                    Locale locale = new Locale(lang);
                    Locale.setDefault(locale);
                    Configuration config = new Configuration    ();
                    config.locale = locale;
                    Splash.this.getResources().updateConfiguration(config,
                            Splash.this.getResources().getDisplayMetrics())   ;

                    //after three seconds, it will execute all of this code.
                    //as such, we then want to redirect to the master-activity
                    Intent mainIntent = new Intent(Splash.this, MainActivity.class);
                    Splash.this.startActivity(mainIntent);

                //then we finish this class. Dispose of it as it is longer needed
                Splash.this.finish();
            }
        }, SPLASH_DISPLAY_LENGTH);

    }

    public void onPause(){
        super.onPause();
```

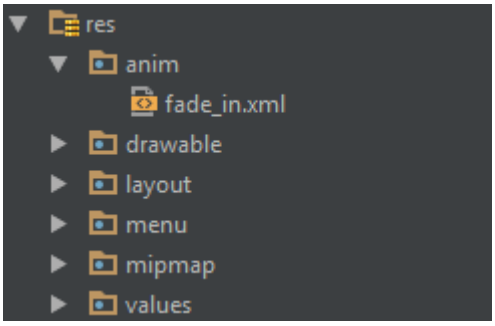
```
        finish();
    }

}
```

Pantalla de bienvenida con animación.

Este ejemplo muestra una pantalla de inicio simple pero efectiva con animación que puede crearse utilizando Android Studio.

Paso 1: Crea una animación.

Crea un nuevo directorio llamado *anim* en el directorio *res* . Haga clic derecho en él y cree un nuevo archivo de *recursos de animación* llamado *fade_in.xml* :



Luego, coloca el siguiente código en el archivo *fade_in.xml* :

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" android:fillAfter="true" >
    <alpha
        android:duration="1000"
        android:fromAlpha="0.0"
        android:interpolator="@android:anim/accelerate_interpolator"
        android:toAlpha="1.0" />
</set>
```

Paso 2: Crear una actividad

Crea una *actividad vacía* usando Android Studio llamado `Splash` . Luego, ponga el siguiente código en él:

```
public class Splash extends AppCompatActivity {
    Animation anim;
    ImageView imageView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        imageView=(ImageView)findViewById(R.id.imageView2); // Declare an imageView to show
the animation.
```

```
        anim = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.fade_in); //
Create the animation.
        anim.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {
            }

            @Override
            public void onAnimationEnd(Animation animation) {
                startActivity(new Intent(this,HomeActivity.class));
                // HomeActivity.class is the activity to go after showing the splash screen.
            }

            @Override
            public void onAnimationRepeat(Animation animation) {
            }
        });
        imageView.startAnimation(anim);
    }
}
```

A continuación, coloque el siguiente código en el archivo de diseño:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_splash"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="your_packagename"
    android:orientation="vertical"
    android:background="@android:color/white">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/imageView2"
        android:layout_weight="1"
        android:src="@drawable/Your_logo_or_image" />
</LinearLayout>
```

Paso 3: Reemplazar el lanzador predeterminado

Convierta su actividad de `Splash` en un iniciador agregando el siguiente código al archivo *AndroidManifest* :

```
<activity
    android:name=".Splash"
    android:theme="@style/AppTheme.NoActionBar">
```

```
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Luego, elimine la actividad del iniciador predeterminada eliminando el siguiente código del archivo *AndroidManifest* :

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Lea Creando pantalla de bienvenida en línea:
https://riptutorial.com/es/android/topic/9316/creando-pantalla-de-bienvenida

Capítulo 75: Creando tus propias bibliotecas para aplicaciones de Android

Examples

Creando proyecto de biblioteca

Para crear una biblioteca, debe usar `File -> New -> New Module -> Android Library` . Esto creará un proyecto de biblioteca básica.

Cuando haya terminado, debe tener un proyecto configurado de la siguiente manera:

```
[project root directory]
    [library root directory]
    [gradle]
    build.gradle //project level
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    settings.gradle //this is important!
```

Su archivo `settings.gradle` debe contener lo siguiente:

```
include ':[library root directory]'
```

Su `[library root directory]` debe contener lo siguiente:

```
[libs]
[src]
   [main]
      [java]
         [library package]
   [test]
      [java]
         [library package]
build.gradle //"app"-level
proguard-rules.pro
```

Su archivo "app" `build.gradle` debe contener lo siguiente:

```
apply plugin: 'com.android.library'

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.2"

    defaultConfig {
        minSdkVersion 14
        targetSdkVersion 23
```

```
    }
}
```

Con eso, tu proyecto debería estar funcionando bien!

Uso de biblioteca en proyecto como módulo.

Para usar la biblioteca, debe incluirla como una dependencia con la siguiente línea:

```
compile project(':[library root directory]')
```

Crear una biblioteca disponible en Jitpack.io

Realice los siguientes pasos para crear la biblioteca:

1. Crea una cuenta de GitHub.
2. Crea un repositorio Git que contenga el proyecto de tu biblioteca.
3. Modifique el archivo `build.gradle` del proyecto de su biblioteca agregando el siguiente código:

   ```
   apply plugin: 'com.github.dcendents.android-maven'

   ...

   // Build a jar with source files.
   task sourcesJar(type: Jar) {
       from android.sourceSets.main.java.srcDirs
       classifier = 'sources'
   }

   task javadoc(type: Javadoc) {
       failOnError  false
       source = android.sourceSets.main.java.sourceFiles
       classpath += project.files(android.getBootClasspath().join(File.pathSeparator))
       classpath += configurations.compile
   }

   // Build a jar with javadoc.
   task javadocJar(type: Jar, dependsOn: javadoc) {
       classifier = 'javadoc'
       from javadoc.destinationDir
   }

   artifacts {
       archives sourcesJar
       archives javadocJar
   }
   ```

   Asegúrese de confirmar / empujar los cambios anteriores a GitHub.

4. Crear una versión del código actual en Github.

5. Ejecute `gradlew install` en su código.

6. Su biblioteca ahora está disponible por la siguiente dependencia:

```
compile 'com.github.[YourUser]:[github repository name]:[release tag]'
```

Lea Creando tus propias bibliotecas para aplicaciones de Android en línea: https://riptutorial.com/es/android/topic/4118/creando-tus-propias-bibliotecas-para-aplicaciones-de-android

Capítulo 76: Crear una clase Singleton para un mensaje de Toast

Introducción

Los mensajes de Toast son la forma más sencilla de proporcionar comentarios al usuario. De forma predeterminada, Android proporciona un mensaje de color gris donde podemos configurar el mensaje y la duración del mensaje. Si necesitamos crear un mensaje de brindis más personalizable y reutilizable, podemos implementarlo por nosotros mismos con el uso de un diseño personalizado. Más importante aún cuando lo estamos implementando, el uso del patrón de diseño de Singelton facilitará el mantenimiento y desarrollo de la clase de mensaje de brindis personalizado.

Sintaxis

- Toast Toast (contextos de contexto)
- void setDuration (int duration)
- void setGravity (int gravity, int xOffset, int yOffset)
- void setView (Vista de vista)
- espectáculo nulo ()

Parámetros

Parámetro	detalles
contexto	Contexto relevante que necesita mostrar su mensaje de brindis. Si usa esto en la actividad, pase la palabra clave "this" o si usa en fragement pass como "getActivity ()".
ver	Cree una vista personalizada y pase esa vista a este objeto.
gravedad	Pasar la posición de gravedad de la tostadora. Toda la posición se ha agregado en la clase Gravedad como las variables estáticas. Las posiciones más comunes son Gravity.TOP, Gravity.BOTTOM, Gravity.LEFT, Gravity.RIGHT.
xOffset	Desplazamiento horizontal del mensaje tostado.
yOffset	Desplazamiento vertical del mensaje tostado.
duración	Duración del espectáculo de brindis. Podemos establecer Toast.LENGTH_SHORT o Toast.LENGTH_LONG

Observaciones

El mensaje de Toast es una forma sencilla de proporcionar comentarios al usuario sobre algo que está sucediendo. Si necesita una forma más avanzada de enviar comentarios, puede utilizar los diálogos o la barra de aperitivos.

Para obtener más detalles sobre el mensaje de brindis, consulte esta documentación.
https://developer.android.com/reference/android/widget/Toast.html

Examples

Crea tu propia clase de singleton para masajes de tostadas.

A continuación le indicamos cómo crear su propia clase de singleton para los mensajes de brindis. Si su aplicación necesita mostrar los mensajes de éxito, advertencia y peligro para diferentes casos de uso, puede usar esta clase después de haberla modificado según sus propias especificaciones.

```
    public class ToastGenerate {
        private static ToastGenerate ourInstance;

        public ToastGenerate (Context context) {
            this.context = context;
                                                }
        public static ToastGenerate getInstance(Context context) {
            if (ourInstance == null)
                ourInstance = new ToastGenerate(context);
            return ourInstance;
        }

       //pass message and message type to this method
        public void createToastMessage(String message,int type){

//inflate the custom layout
            LayoutInflater layoutInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

            LinearLayout toastLayout = (LinearLayout)
layoutInflater.inflate(R.layout.layout_custome_toast,null);
            TextView toastShowMessage = (TextView)
toastLayout.findViewById(R.id.textCustomToastTopic);

            switch (type){
                case 0:
                    //if the message type is 0 fail toaster method will call
                    createFailToast(toastLayout,toastShowMessage,message);
                    break;
                case 1:
                    //if the message type is 1 success toaster method will call
                    createSuccessToast(toastLayout,toastShowMessage,message);
                    break;

                case 2:
                    createWarningToast( toastLayout, toastShowMessage, message);
                    //if the message type is 2 warning toaster method will call
                    break;
                default:
                    createFailToast(toastLayout,toastShowMessage,message);
```

```
            }
        }

    //Failure toast message method
        private final void createFailToast(LinearLayout toastLayout,TextView
toastMessage,String message){

toastLayout.setBackgroundColor(context.getResources().getColor(R.color.button_alert_normal));
            toastMessage.setText(message);
            toastMessage.setTextColor(context.getResources().getColor(R.color.white));
            showToast(context,toastLayout);
        }

        //warning toast message method
        private final void createWarningToast( LinearLayout toastLayout, TextView
toastMessage, String message) {

toastLayout.setBackgroundColor(context.getResources().getColor(R.color.warning_toast));
            toastMessage.setText(message);
            toastMessage.setTextColor(context.getResources().getColor(R.color.white));
            showToast(context, toastLayout);
        }
    //success toast message method
        private final void createSuccessToast(LinearLayout toastLayout,TextView
toastMessage,String message){

toastLayout.setBackgroundColor(context.getResources().getColor(R.color.success_toast));

            toastMessage.setText(message);
            toastMessage.setTextColor(context.getResources().getColor(R.color.white));
            showToast(context,toastLayout);
        }

        private void showToast(View view){
            Toast toast = new Toast(context);
            toast.setGravity(Gravity.TOP,0,0); // show message in the top of the device
            toast.setDuration(Toast.LENGTH_SHORT);
            toast.setView(view);
            toast.show();
        }
    }
```

Lea Crear una clase Singleton para un mensaje de Toast en línea:
https://riptutorial.com/es/android/topic/10843/crear-una-clase-singleton-para-un-mensaje-de-toast

Capítulo 77: Cuadro de diálogo animado de alerta

Introducción

Cuadro de diálogo de alerta animado que se muestra con algunos efectos de animación ... Puede obtener un poco de animación para cuadros de diálogo como Fadein, Slideleft, Slidetop, SlideBottom, Sliderright, Fall, Newspager, Fliph, Flipv, RotateBottom, RotateLeft, Slit, Shake, Sidefill para hacer su aplicación atractiva ..

Examples

Poner código debajo para diálogo animado ...

```
animated_android_dialog_box.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="16dp">

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#1184be"
    android:onClick="animatedDialog1"
    android:text="Animated Fall Dialog"
    android:textColor="#fff" />

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginTop="16dp"
    android:background="#1184be"
    android:onClick="animatedDialog2"
    android:text="Animated Material Flip Dialog"
    android:textColor="#fff" />

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#1184be"
    android:onClick="animatedDialog3"
    android:text="Animated Material Shake Dialog"
    android:textColor="#fff" />

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
```

```
    android:layout_marginTop="16dp"
    android:background="#1184be"
    android:onClick="animatedDialog4"
    android:text="Animated Slide Top Dialog"
    android:textColor="#fff" />
```

AnimatedAndroidDialogExample.java

```
   public class AnimatedAndroidDialogExample extends AppCompatActivity {

    NiftyDialogBuilder materialDesignAnimatedDialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.animated_android_dialog_box);

        materialDesignAnimatedDialog = NiftyDialogBuilder.getInstance(this);
    }

    public void animatedDialog1(View view) {
        materialDesignAnimatedDialog
                .withTitle("Animated Fall Dialog Title")
                .withMessage("Add your dialog message here. Animated dialog description
place.")
                .withDialogColor("#FFFFFF")
                .withButton1Text("OK")
                .withButton2Text("Cancel")
                .withDuration(700)
                .withEffect(Effectstype.Fall)
                .show();
    }

    public void animatedDialog2(View view) {
        materialDesignAnimatedDialog
                .withTitle("Animated Flip Dialog Title")
                .withMessage("Add your dialog message here. Animated dialog description
place.")
                .withDialogColor("#1c90ec")
                .withButton1Text("OK")
                .withButton2Text("Cancel")
                .withDuration(700)
                .withEffect(Effectstype.Fliph)
                .show();
    }

    public void animatedDialog3(View view) {
        materialDesignAnimatedDialog
                .withTitle("Animated Shake Dialog Title")
                .withMessage("Add your dialog message here. Animated dialog description
place.")
                .withDialogColor("#1c90ec")
                .withButton1Text("OK")
                .withButton2Text("Cancel")
                .withDuration(700)
                .withEffect(Effectstype.Shake)
                .show();
    }

    public void animatedDialog4(View view) {
```

```
        materialDesignAnimatedDialog
                .withTitle("Animated Slide Top Dialog Title")
                .withMessage("Add your dialog message here. Animated dialog description
place.")
                .withDialogColor("#1c90ec")
                .withButton1Text("OK")
                .withButton2Text("Cancel")
                .withDuration(700)
                .withEffect(Effectstype.Slidetop)
                .show();
    }
}
```

Agregue las siguientes líneas en su build.gradle para incluir el NifyBuilder (CustomView)

construir.gradle

```
 dependencies {

     compile 'com.nineoldandroids:library:2.4.0'

     compile 'com.github.sd6352051.niftydialogeffects:niftydialogeffects:1.0.0@aar'

 }
```

Enlace de referencia: https://github.com/sd6352051/NiftyDialogEffects

Lea Cuadro de diálogo animado de alerta en línea:
https://riptutorial.com/es/android/topic/10654/cuadro-de-dialogo-animado-de-alerta

Capítulo 78: Cuchillo de mantequilla

Introducción

Butterknife es una herramienta de enlace de vista que utiliza anotaciones para generar código repetitivo para nosotros. Esta herramienta fue desarrollada por Jake Wharton en Square y se usa esencialmente para guardar líneas de código repetitivas como `findViewById(R.id.view)` cuando se trata de vistas, lo que hace que nuestro código se vea mucho más limpio.

Para ser claros, Butterknife **no** es **una biblioteca de inyección de dependencias** . Butterknife inyecta código en tiempo de compilación. Es muy similar al trabajo realizado por las anotaciones de Android.

Observaciones

Cuchillo de mantequilla

Encuadernación de campos y métodos para las vistas de Android, que utiliza el procesamiento de anotaciones para generar el código de repetición para usted.

- Elimine las llamadas a findViewById usando @BindView en los campos.
- Agrupa múltiples vistas en una lista o matriz. Operar en todos ellos a la vez con acciones, configuradores o propiedades.
- Elimine clases internas anónimas para oyentes anotando métodos con @OnClick y otros.
- Elimine las búsquedas de recursos mediante el uso de anotaciones de recursos en los campos.

Más información: http://jakewharton.github.io/butterknife/

Licencia

Copyright 2013 Jake Wharton

Licenciado bajo la Licencia Apache, Versión 2.0 (la "Licencia"); no puede utilizar este archivo, excepto en cumplimiento con la Licencia. Puede obtener una copia de la licencia en

http://www.apache.org/licenses/LICENSE-2.0

A menos que así lo exija la ley aplicable o se acuerde por escrito, el software distribuido bajo la Licencia se distribuye "TAL CUAL", SIN GARANTÍAS O CONDICIONES DE NINGÚN TIPO, ya sea explícita o implícita. Consulte la Licencia para el idioma específico que rige los permisos y las limitaciones de la Licencia.

Examples

Configurando ButterKnife en tu proyecto

Configure su `build.gradle` a nivel de `build.gradle` para incluir el complemento `android-apt` :

```
buildscript {
   repositories {
      mavenCentral()
   }

   dependencies {
      classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
   }
}
```

Luego, aplique el complemento de `android-apt` en su `build.gradle` nivel de `build.gradle` y agregue las dependencias de ButterKnife:

```
apply plugin: 'android-apt'

android {
    ...
}

dependencies {
      compile 'com.jakewharton:butterknife:8.5.1'
      annotationProcessor 'com.jakewharton:butterknife-compiler:8.5.1'
}
```

Nota: si está utilizando el nuevo compilador Jack con la versión 2.2.0 o posterior, no necesita el complemento `android-apt` y puede reemplazar apt con el procesador de `annotationProcessor` cuando declare la dependencia del compilador.

Para utilizar las anotaciones de ButterKnife, no debe olvidarse de vincularlas en `onCreate()` de sus Actividades o `onCreateView()` de sus Fragmentos:

```
class ExampleActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Binding annotations
        ButterKnife.bind(this);
        // ...
  }

}

// Or
class ExampleFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
```

```
        View view = inflater.inflate(getContentView(), container, false);
        // Binding annotations
        ButterKnife.bind(this, view);
        // ...
        return view;
    }

}
```

Las instantáneas de la versión de desarrollo están disponibles en el repositorio de instantáneas de Sonatype .

A continuación se muestran los pasos adicionales que tendría que tomar para usar ButterKnife en un proyecto de biblioteca.

Para usar ButterKnife en un proyecto de biblioteca, agregue el complemento a su `build.gradle` nivel de `build.gradle` :

```
buildscript {
    dependencies {
        classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
    }
}
```

… Y luego aplique a su módulo agregando estas líneas en la parte superior de su `build.gradle` nivel de `build.gradle` :

```
apply plugin: 'com.android.library'
// ...
apply plugin: 'com.jakewharton.butterknife'
```

Ahora asegúrese de usar `R2` lugar de `R` dentro de todas las anotaciones de ButterKnife.

```
class ExampleActivity extends Activity {

    // Bind xml resource to their View
    @BindView(R2.id.user) EditText username;
    @BindView(R2.id.pass) EditText password;

    // Binding resources from drawable,strings,dimens,colors
    @BindString(R.string.choose) String choose;
    @BindDrawable(R.drawable.send) Drawable send;
    @BindColor(R.color.cyan) int cyan;
    @BindDimen(R.dimen.margin) Float generalMargin;

    // Listeners
    @OnClick(R.id.submit)
    public void submit(View view) {
    // TODO submit data to server...
    }

    // bind with butterknife in onCreate
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);
        // TODO continue
    }

}
```

Encuadernar vistas usando ButterKnife

podemos anotar los campos con `@BindView` y una ID de vista para que Butter Knife encuentre y lance automáticamente la vista correspondiente en nuestro diseño.

Vistas obligatorias

Vistas obligatorias en actividad

```
class ExampleActivity extends Activity {
  @BindView(R.id.title) TextView title;
  @BindView(R.id.subtitle) TextView subtitle;
  @BindView(R.id.footer) TextView footer;

  @Override public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.simple_activity);
    ButterKnife.bind(this);
    // TODO Use fields...
  }
}
```

Encuadernación de vistas en fragmentos

```
public class FancyFragment extends Fragment {
  @BindView(R.id.button1) Button button1;
  @BindView(R.id.button2) Button button2;
  private Unbinder unbinder;

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    View view = inflater.inflate(R.layout.fancy_fragment, container, false);
    unbinder = ButterKnife.bind(this, view);
    // TODO Use fields...
    return view;
  }

  // in fragments or non activity bindings we need to unbind the binding when view is about to
be destroyed
  @Override
    public void onDestroy() {
        super.onDestroy();
        unbinder.unbind();
    }
```

```
}
```

Encuadernar vistas en diálogos

Podemos usar `ButterKnife.findById` para buscar vistas en una Vista, Actividad o Diálogo. Utiliza genéricos para inferir el tipo de retorno y realiza automáticamente la conversión.

```
View view = LayoutInflater.from(context).inflate(R.layout.thing, null);
TextView firstName = ButterKnife.findById(view, R.id.first_name);
TextView lastName = ButterKnife.findById(view, R.id.last_name);
ImageView photo = ButterKnife.findById(view, R.id.photo);
```

Vistas vinculantes en ViewHolder

```
static class ViewHolder {
    @BindView(R.id.title) TextView name;
    @BindView(R.id.job_title) TextView jobTitle;

    public ViewHolder(View view) {
      ButterKnife.bind(this, view);
    }
  }
```

Recursos vinculantes

Aparte de ser útil para vistas de unión, también se podría utilizar butterknife para unirse recursos tales como los que se definen dentro de `strings.xml` , `drawables.xml` , `colors.xml` , `dimens.xml` , etc.

```
public class ExampleActivity extends Activity {

    @BindString(R.string.title) String title;
    @BindDrawable(R.drawable.graphic) Drawable graphic;
    @BindColor(R.color.red) int red; // int or ColorStateList field
    @BindDimen(R.dimen.spacer) Float spacer; // int (for pixel size) or float (for exact
value) field

    @Override
    public void onCreate(Bundle savedInstanceState) {

        // ...

        ButterKnife.bind(this);
    }

}
```

Encuadernación de listas de vistas

Puede agrupar varias vistas en una lista o matriz. Esto es muy útil cuando necesitamos realizar una acción en varias vistas a la vez.

```
@BindViews({ R.id.first_name, R.id.middle_name, R.id.last_name })
List<EditText> nameViews;

//The apply method allows you to act on all the views in a list at once.
ButterKnife.apply(nameViews, DISABLE);
ButterKnife.apply(nameViews, ENABLED, false);


//We can use Action and Setter interfaces allow specifying simple behavior.
static final ButterKnife.Action<View> DISABLE = new ButterKnife.Action<View>() {
  @Override public void apply(View view, int index) {
    view.setEnabled(false);
  }
};
static final ButterKnife.Setter<View, Boolean> ENABLED = new ButterKnife.Setter<View,
Boolean>() {
  @Override public void set(View view, Boolean value, int index) {
    view.setEnabled(value);
  }
};
```

Fijaciones opcionales

De forma predeterminada, se requieren los enlaces `@Bind` y listener. Se lanza una excepción si no se puede encontrar la vista de destino. Pero si no estamos seguros de si habrá una vista o no, podemos agregar una anotación `@Nullable` a los campos o la anotación `@Optional` a los métodos para suprimir este comportamiento y crear un enlace opcional.

```
@Nullable
@BindView(R.id.might_not_be_there) TextView mightNotBeThere;

@Optional
@OnClick(R.id.maybe_missing)
void onMaybeMissingClicked() {
  // TODO ...
}
```

Oidores obligatorios usando ButterKnife

OnClick Listener:

```
@OnClick(R.id.login)
public void login(View view) {
  // Additional logic
}
```

Todos los argumentos del método de escucha son opcionales:

```
@OnClick(R.id.login)
```

```
public void login() {
   // Additional logic
}
```

Tipo específico será casteado automáticamente:

```
@OnClick(R.id.submit)
public void sayHi(Button button) {
  button.setText("Hello!");
}
```

ID múltiples en un solo enlace para el manejo de eventos comunes:

```
@OnClick({ R.id.door1, R.id.door2, R.id.door3 })
public void pickDoor(DoorView door) {
  if (door.hasPrizeBehind()) {
    Toast.makeText(this, "You win!", LENGTH_SHORT).show();
  } else {
    Toast.makeText(this, "Try again", LENGTH_SHORT).show();
  }
}
```

Las vistas personalizadas pueden vincularse a sus propios oyentes al no especificar una ID:

```
public class CustomButton extends Button {
  @OnClick
  public void onClick() {
    // TODO
  }
}
```

Vistas sin compromiso en ButterKnife

Los fragmentos tienen un ciclo de vida de vista diferente al de las actividades. Al vincular un fragmento en onCreateView, establezca las vistas en nulo en onDestroyView. Butter Knife devuelve una instancia de Unbinder cuando llama a bind para hacer esto por usted. Llame a su método de desvinculación en la devolución de llamada apropiada del ciclo de vida.

Un ejemplo:

```
public class MyFragment extends Fragment {
  @BindView(R.id.textView) TextView textView;
  @BindView(R.id.button) Button button;
  private Unbinder unbinder;

  @Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    View view = inflater.inflate(R.layout.my_fragment, container, false);
    unbinder = ButterKnife.bind(this, view);
    // TODO Use fields...
    return view;
  }

  @Override public void onDestroyView() {
```

```
    super.onDestroyView();
    unbinder.unbind();
  }
}
```

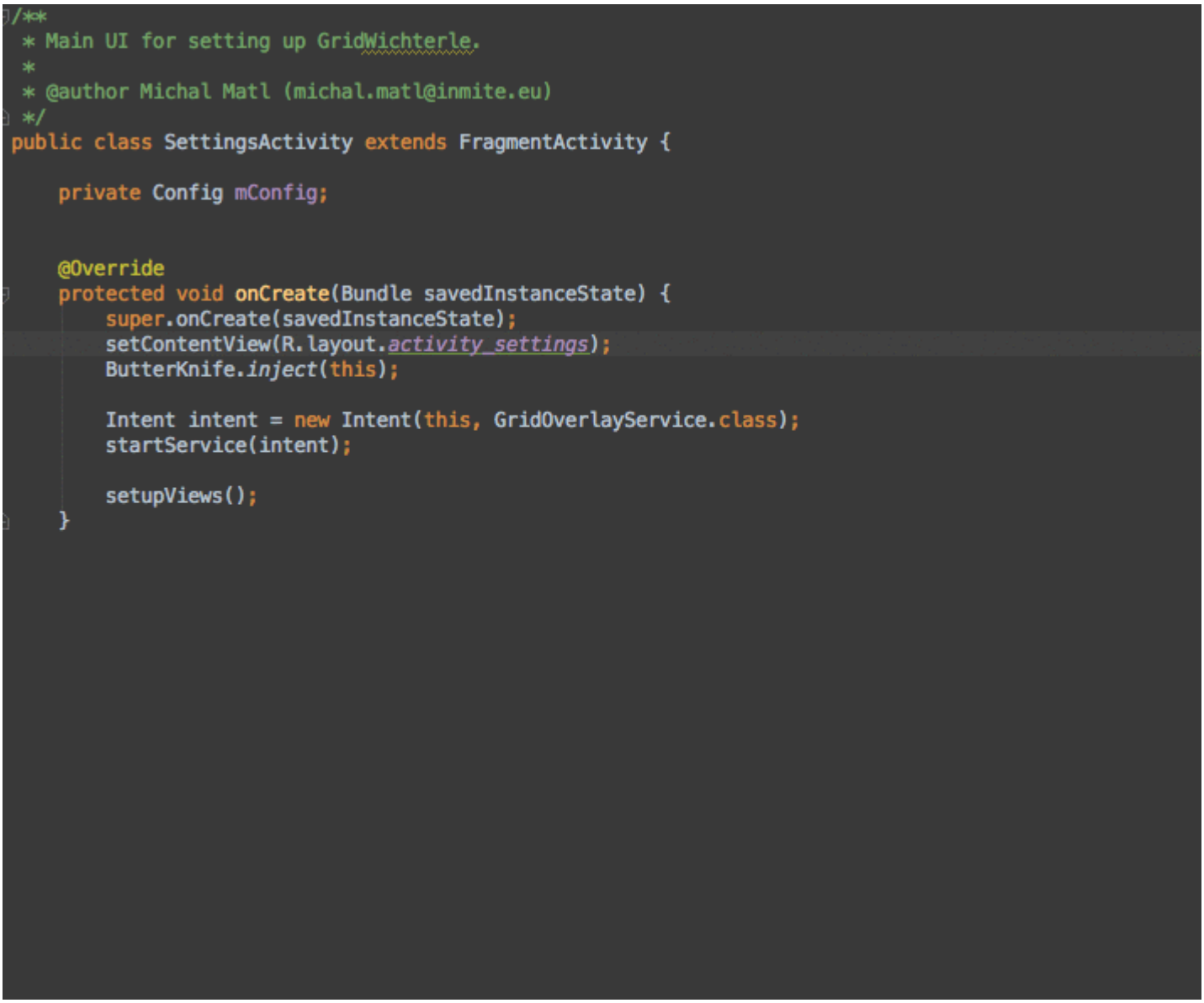
> Nota: No se requiere llamar a unbind () en onDestroyView (), pero se recomienda ya que ahorra bastante memoria si su aplicación tiene un backstack grande.

Android Studio ButterKnife Plugin

Android ButterKnife Zelezny

> Complemento para generar inyecciones de ButterKnife a partir de XML de diseño seleccionados en actividades / fragmentos / adaptadores.

Nota: asegúrese de hacer el clic derecho para ***su_xml_layou*** `(R.layout.your_xml_layou`), de lo contrario, el *menú Generar* no contendrá la opción de inyector de Butterknife.



```
/**
 * Main UI for setting up GridWichterle.
 *
 * @author Michal Matl (michal.matl@inmite.eu)
 */
public class SettingsActivity extends FragmentActivity {

    private Config mConfig;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);
        ButterKnife.inject(this);

        Intent intent = new Intent(this, GridOverlayService.class);
        startService(intent);

        setupViews();
    }
```

Enlace: Jetbrains Plugin Android ButterKnife Zelezny

Lea Cuchillo de mantequilla en línea: https://riptutorial.com/es/android/topic/1072/cuchillo-de-mantequilla

Capítulo 79: Cuentas y AccountManager

Examples

Comprensión de cuentas personalizadas / autenticación

El siguiente ejemplo es la cobertura de alto nivel de los conceptos clave y la configuración básica del esqueleto:

1. Recopila credenciales del usuario (normalmente de una pantalla de inicio de sesión que ha creado)
2. Autentica las credenciales con el servidor (almacena autenticación personalizada)
3. Almacena las credenciales en el dispositivo.

Extienda un AbstractAccountAuthenticator *(utilizado principalmente para recuperar la autenticación y volver a autenticarlos)*

```
public class AccountAuthenticator extends AbstractAccountAuthenticator {

  @Override
  public Bundle addAccount(AccountAuthenticatorResponse response, String accountType,
      String authTokenType, String[] requiredFeatures, Bundle options) {
    //intent to start the login activity
  }


  @Override
  public Bundle confirmCredentials(AccountAuthenticatorResponse response, Account account,
Bundle options) {
  }

  @Override
  public Bundle editProperties(AccountAuthenticatorResponse response, String accountType) {
  }

  @Override
  public Bundle getAuthToken(AccountAuthenticatorResponse response, Account account, String
authTokenType,
      Bundle options) throws NetworkErrorException {
    //retrieve authentication tokens from account manager storage or custom storage or re-
authenticate old tokens and return new ones
  }

  @Override
  public String getAuthTokenLabel(String authTokenType) {
  }

  @Override
  public Bundle hasFeatures(AccountAuthenticatorResponse response, Account account, String[]
features)
      throws NetworkErrorException {
    //check whether the account supports certain features
```

```
  }

  @Override
  public Bundle updateCredentials(AccountAuthenticatorResponse response, Account account,
String authTokenType,
      Bundle options) {
    //when the user's session has expired or requires their previously available credentials
to be updated, here is the function to do it.
  }
}
```

Crear un servicio *(el marco de Account Manager se conecta al AbstractAccountAuthenticator extendido a través de la interfaz del servicio)*

```
public class AuthenticatorService extends Service {

    private AccountAuthenticator authenticator;

    @Override
    public void onCreate(){
        authenticator = new AccountAuthenticator(this);
    }

    @Override
    public IBinder onBind(Intent intent) {
        return authenticator.getIBinder();
    }
}
```

Configuración XML del autenticador *(el marco de trabajo del administrador de cuentas requiere. Esto es lo que verá en Configuración -> Cuentas en Android)*

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="rename.with.your.applicationid"
    android:icon="@drawable/app_icon"
    android:label="@string/app_name"
    android:smallIcon="@drawable/app_icon" />
```

Cambios en el AndroidManifest.xml *(reúne todos los conceptos anteriores para que se pueda utilizar mediante programación a través del AccountManager)*

```
<application
...>
    <service
        android:name=".authenticator.AccountAuthenticatorService"
        android:exported="false"
        android:process=":authentication">
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator"/>
```

```
        </intent-filter>
        <meta-data
            android:name="android.accounts.AccountAuthenticator"
            android:resource="@xml/authenticator"/>
    </service>
</application>
```

El siguiente ejemplo contendrá cómo hacer uso de esta configuración.

Lea Cuentas y AccountManager en línea: https://riptutorial.com/es/android/topic/7003/cuentas-y-accountmanager

Capítulo 80: Daga 2

Sintaxis

- @Módulo
- @Component (dependencias = {OtherComponent.class}, modules = {ModuleA.class, ModuleB.class})
- DaggerMyComponent.create ()
- DaggerMyComponent.builder (). MyModule (newMyModule ()). Create ()

Observaciones

No confundir con daga por escuadra, el antecesor de daga 2.

Examples

Configuración de componentes para inyección de aplicación y actividad.

Un `AppComponent` básico que depende de un único `AppModule` para proporcionar objetos singleton para toda la aplicación.

```
@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {

    void inject(App app);

    Context provideContext();

    Gson provideGson();
}
```

Un módulo para usar junto con `AppComponent` que proporcionará sus objetos singleton, por ejemplo, una instancia de `Gson` para reutilizarla en toda la aplicación.

```
@Module
public class AppModule {

    private final Application mApplication;

    public AppModule(Application application) {
        mApplication = application;
    }

    @Singleton
    @Provides
    Gson provideGson() {
        return new Gson();
    }
```

```
    @Singleton
    @Provides
    Context provideContext() {
        return mApplication;
    }
}
```

Una aplicación subclasificada para configurar la daga y el componente singleton.

```
public class App extends Application {

    @Inject
    AppComponent mAppComponent;

    @Override
    public void onCreate() {
        super.onCreate();

        DaggerAppComponent.builder().appModule(new AppModule(this)).build().inject(this);
    }

    public AppComponent getAppComponent() {
        return mAppComponent;
    }
}
```

Ahora es un componente de ámbito de actividad que depende de `AppComponent` para obtener acceso a los objetos singleton.

```
@ActivityScope
@Component(dependencies = AppComponent.class, modules = ActivityModule.class)
public interface MainActivityComponent {

    void inject(MainActivity activity);
}
```

Y un módulo de `ActivityModule` reutilizable que proporcionará dependencias básicas, como un `FragmentManager`

```
@Module
public class ActivityModule {

    private final AppCompatActivity mActivity;

    public ActivityModule(AppCompatActivity activity) {
        mActivity = activity;
    }

    @ActivityScope
    public AppCompatActivity provideActivity() {
        return mActivity;
    }


    @ActivityScope
    public FragmentManager provideFragmentManager(AppCompatActivity activity) {
```

```
        return activity.getSupportFragmentManager();
    }
}
```

Poniendo todo junto, estamos configurados, podemos inyectar nuestra actividad y ¡asegúrate de usar la misma aplicación `Gson` toda la aplicación!

```
public class MainActivity extends AppCompatActivity {

    @Inject
    Gson mGson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        DaggerMainActivityComponent.builder()
                .appComponent(((App)getApplication()).getAppComponent())
                .activityModule(new ActivityModule(this))
                .build().inject(this);
    }
}
```

Alcances personalizados

```
@Scope
@Documented
@Retention(RUNTIME)
public @interface ActivityScope {
}
```

Los ámbitos son solo anotaciones y usted puede crear sus propios cuando sea necesario.

Inyección Constructor

Las clases sin dependencias se pueden crear fácilmente por daga.

```
public class Engine {

    @Inject // <-- Annotate your constructor.
    public Engine() {
    }
}
```

Esta clase puede ser proporcionada por *cualquier* componente. No tiene *dependencias en sí* y *no* está dentro del *alcance* . No hay ningún otro código necesario.

Las dependencias se declaran como parámetros en el constructor. Dagger llamará al constructor y suministrará las dependencias, siempre que esas dependencias puedan proporcionarse.

```
public class Car {
```

```
    private Engine engine;

    @Inject
    public Car(Engine engine) {
        this.engine = engine;
    }
}
```

Todos los componentes pueden proporcionar esta clase *si* este componente también puede proporcionar todas sus dependencias: `Engine` en este caso. Dado que el `Engine` también puede ser inyectado por el constructor, *cualquier* componente puede proporcionar un `Car` .

Puede utilizar la inyección de constructor siempre que el componente proporcione todas las dependencias. Un componente puede proporcionar una dependencia, si

- Se puede crear mediante inyección de constructor.
- Un módulo del componente puede proporcionarlo.
- puede ser proporcionado por el componente principal (si es un `@Subcomponent`)
- puede usar un objeto expuesto por un componente del que depende (dependencias del componente)

Usar @Subcomponent en lugar de @Component (dependencias = {...})

```
@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {
    void inject(App app);

    Context provideContext();
    Gson provideGson();

    MainActivityComponent mainActivityComponent(ActivityModule activityModule);
}

@ActivityScope
@Subcomponent(modules = ActivityModule.class)
public interface MainActivityComponent {
    void inject(MainActivity activity);
}

public class MainActivity extends AppCompatActivity {

    @Inject
    Gson mGson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ((App)getApplication()).getAppComponent()
                .mainActivityComponent(new ActivityModule(this)).inject(this);
    }
}
```

Cómo agregar Dagger 2 en build.gradle

Desde el lanzamiento de Gradle 2.2, ya no se usa el complemento android-apt. Se debe utilizar el siguiente método de configuración de Dagger 2. Para la versión anterior de Gradle, use el método anterior que se muestra a continuación.

Para Gradle> = 2.2

```
dependencies {
    // apt command comes from the android-apt plugin
    annotationProcessor 'com.google.dagger:dagger-compiler:2.8'
    compile 'com.google.dagger:dagger:2.8'
    provided 'javax.annotation:jsr250-api:1.0'
}
```

Para Gradle <2.2

Para usar Dagger 2 es necesario agregar el complemento `android-apt` , agregar esto a la raíz build.gradle:

```
buildscript {
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.0'
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}
```

Entonces el build.gradle del módulo de la aplicación debe contener:

```
apply plugin: 'com.android.application'
apply plugin: 'com.neenbedankt.android-apt'


android {
    …
}

final DAGGER_VERSION = '2.0.2'
dependencies {
    …

    compile "com.google.dagger:dagger:${DAGGER_VERSION}"
    apt "com.google.dagger:dagger-compiler:${DAGGER_VERSION}"
}
```

Referencia: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Creando un componente a partir de múltiples módulos.

Dagger 2 admite la creación de un componente a partir de múltiples módulos. Puedes crear tu componente de esta manera:

```
@Singleton
```

```
@Component(modules = {GeneralPurposeModule.class, SpecificModule.class})
public interface MyMultipleModuleComponent {
    void inject(MyFragment myFragment);
    void inject(MyService myService);
    void inject(MyController myController);
    void inject(MyActivity myActivity);
}
```

Los dos módulos de referencia `GeneralPurposeModule` y `SpecificModule` se pueden implementar de la siguiente manera:

GeneralPurposeModule.java

```
@Module
public class GeneralPurposeModule {
    @Provides
    @Singleton
    public Retrofit getRetrofit(PropertiesReader propertiesReader, RetrofitHeaderInterceptor
headerInterceptor){
        // Logic here...
        return retrofit;
    }

    @Provides
    @Singleton
    public PropertiesReader getPropertiesReader(){
        return new PropertiesReader();
    }

    @Provides
    @Singleton
    public RetrofitHeaderInterceptor getRetrofitHeaderInterceptor(){
         return new RetrofitHeaderInterceptor();
    }
}
```

SpecificModule.java

```
@Singleton
@Module
public class SpecificModule {
    @Provides @Singleton
    public RetrofitController getRetrofitController(Retrofit retrofit){
        RetrofitController retrofitController = new RetrofitController();
        retrofitController.setRetrofit(retrofit);
        return retrofitController;
    }

    @Provides @Singleton
    public MyService getMyService(RetrofitController retrofitController){
        MyService myService = new MyService();
        myService.setRetrofitController(retrofitController);
        return myService;
    }
}
```

Durante la fase de inyección de dependencia, el componente tomará objetos de ambos módulos

según las necesidades.

Este enfoque es muy útil en términos de *modularidad* . En el ejemplo, hay un módulo de propósito general que se utiliza para crear una instancia de componentes como el objeto `Retrofit` (usado para manejar la comunicación de la red) y un `PropertiesReader` (encargado de manejar los archivos de configuración). También hay un módulo específico que maneja la creación de instancias de controladores y clases de servicio específicos en relación con ese componente de aplicación específico.

Lea Daga 2 en línea: https://riptutorial.com/es/android/topic/3088/daga-2

Capítulo 81: Defina el valor del paso (incremento) para la barra de barras personalizada

Introducción

Una personalización de la RangeSeekBar de Android propuesta por Alex Florescu en https://github.com/anothem/android-range-seek-bar

Permite definir un valor de paso (incremento), al mover la barra de búsqueda.

Observaciones

1- Añadir el atributo de incremento en attrs.xml

```
<attr name="increment" format="integer|float"/>
```

2- Defina un valor predeterminado en RangeSeekBar.java y cree el atributo también

```
private static final int DEFAULT_INCREMENT = 1;
private int increment;
```

3- Iniciar el valor de incremento en init vacío privado (contexto de contexto, atributos AttributeSet)

```
if (attrs == null)
    increment = DEFAULT_INCREMENT;
else
    increment = a.getInt(R.styleable.RangeSeekBar_increment, DEFAULT_INCREMENT);
```

4- Defina el valor de incremento en onidraw vacío sincronizado protegido (lienzo de lienzo No @ nulo)

Tendrá que reemplazar los valores minText y maxText. Así que en lugar de:

- minText = valueToString (getSelectedMinValue ());
- maxText = valueToString (getSelectedMaxValue ());

Tendrás: int x;

```
        x = (int) ((getSelectedMinValue().intValue()+increment)/increment);
        x = x*increment;
        if (x<absoluteMaxValue.intValue())
            minText = ""+x;
        else
            minText=""+(absoluteMaxValue.intValue()-increment);
```

```
        x = (int) ((getSelectedMaxValue().intValue()+increment)/increment);
        x = x*increment;
        maxText = ""+x;
```

5 - Ahora solo tienes que usarlo. Espero eso ayude

Examples

Definir un valor de paso de 7.

```
<RangeSeekBar
        android:id="@+id/barPrice"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        app:barHeight="0.2dp"
        app:barHeight2="4dp"
        app:increment="7"
        app:showLabels="false" />
```

Lea Defina el valor del paso (incremento) para la barra de barras personalizada en línea: https://riptutorial.com/es/android/topic/8627/defina-el-valor-del-paso--incremento--para-la-barra-de-barras-personalizada

Capítulo 82: Desarrollo de juegos para Android

Introducción

Una breve introducción a la creación de un juego en la plataforma Android utilizando Java.

Observaciones

- El primer ejemplo cubre los conceptos básicos: no hay objetivos, pero te muestra cómo crear una parte básica de un juego 2D utilizando SurfaceView.
- Asegúrate de guardar cualquier dato importante cuando crees un juego; todo lo demás se perderá

Examples

Juego usando Canvas y SurfaceView

Esto cubre cómo puedes crear un juego 2D básico usando SurfaceView.

Primero, necesitamos una actividad:

```
public class GameLauncher extends AppCompatActivity {

    private Game game;
    @Override
    public void onCreate(Bundle sis){
        super.onCreate(sis);
        game = new Game(GameLauncher.this);//Initialize the game instance
        setContentView(game);//setContentView to the game surfaceview
        //Custom XML files can also be used, and then retrieve the game instance using
findViewById.
    }

}
```

La actividad también tiene que ser declarada en el manifiesto de Android.

Ahora para el juego en sí. Primero, comenzamos implementando un hilo de juego:

```
public class Game extends SurfaceView implements SurfaceHolder.Callback, Runnable{

    /**
     * Holds the surface frame
     */
    private SurfaceHolder holder;
```

```
    /**
     * Draw thread
     */
    private Thread drawThread;

    /**
     * True when the surface is ready to draw
     */
    private boolean surfaceReady = false;


    /**
     * Drawing thread flag
     */

    private boolean drawingActive = false;

    /**
     * Time per frame for 60 FPS
     */
    private static final int MAX_FRAME_TIME = (int) (1000.0 / 60.0);

    private static final String LOGTAG = "surface";

    /*
     * All the constructors are overridden to ensure functionality if one of the different
constructors are used through an XML file or programmatically
     */
    public Game(Context context) {
        super(context);
        init();
    }
    public Game(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }
    public Game(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
    }
    @TargetApi(21)
    public Game(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
        init();
    }

    public void init(Context c) {
        this.c = c;

        SurfaceHolder holder = getHolder();
        holder.addCallback(this);
        setFocusable(true);
        //Initialize other stuff here later
    }

    public void render(Canvas c){
        //Game rendering here
    }

    public void tick(){
```

```
        //Game logic here
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
    {
        if (width == 0 || height == 0){
            return;
        }

        // resize your UI
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder){
        this.holder = holder;

        if (drawThread != null){
            Log.d(LOGTAG, "draw thread still active..");
            drawingActive = false;
            try{
                drawThread.join();
            } catch (InterruptedException e){}
        }

        surfaceReady = true;
        startDrawThread();
        Log.d(LOGTAG, "Created");
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder){
        // Surface is not used anymore - stop the drawing thread
        stopDrawThread();
        // and release the surface
        holder.getSurface().release();

        this.holder = null;
        surfaceReady = false;
        Log.d(LOGTAG, "Destroyed");
    }

    @Override
    public boolean onTouchEvent(MotionEvent event){
        // Handle touch events
        return true;
    }

    /**
     * Stops the drawing thread
     */
    public void stopDrawThread(){
        if (drawThread == null){
            Log.d(LOGTAG, "DrawThread is null");
            return;
        }
        drawingActive = false;
        while (true){
            try{
                Log.d(LOGTAG, "Request last frame");
                drawThread.join(5000);
```

```
                break;
            } catch (Exception e) {
                Log.e(LOGTAG, "Could not join with draw thread");
            }
        }
        drawThread = null;
    }

    /**
     * Creates a new draw thread and starts it.
     */
    public void startDrawThread(){
        if (surfaceReady && drawThread == null){
            drawThread = new Thread(this, "Draw thread");
            drawingActive = true;
            drawThread.start();
        }
    }

    @Override
    public void run() {
        Log.d(LOGTAG, "Draw thread started");
        long frameStartTime;
        long frameTime;

        /*
         * In order to work reliable on Nexus 7, we place ~500ms delay at the start of drawing
thread
         * (AOSP - Issue 58385)
         */
        if (android.os.Build.BRAND.equalsIgnoreCase("google") &&
android.os.Build.MANUFACTURER.equalsIgnoreCase("asus") &&
android.os.Build.MODEL.equalsIgnoreCase("Nexus 7")) {
            Log.w(LOGTAG, "Sleep 500ms (Device: Asus Nexus 7)");
            try {
                Thread.sleep(500);
            } catch (InterruptedException ignored) {}
        }

        while (drawing) {
            if (sf == null) {
                return;
            }

            frameStartTime = System.nanoTime();
            Canvas canvas = sf.lockCanvas();
            if (canvas != null) {
                try {
                    synchronized (sf) {
                        tick();
                        render(canvas);
                    }
                } finally {

                    sf.unlockCanvasAndPost(canvas);
                }
            }

            // calculate the time required to draw the frame in ms
            frameTime = (System.nanoTime() - frameStartTime) / 1000000;
```

```
            if (frameTime < MAX_FRAME_TIME){
                try {
                    Thread.sleep(MAX_FRAME_TIME - frameTime);
                } catch (InterruptedException e) {
                    // ignore
                }
            }

        }
        Log.d(LOGTAG, "Draw thread finished");
    }
}
```

Esa es la parte básica. Ahora tienes la habilidad de dibujar en la pantalla.

Ahora, comencemos agregando números enteros:

```
public final int x = 100;//The reason for this being static will be shown when the game is
runnable
public int y;
public int velY;
```

Para esta próxima parte, vas a necesitar una imagen. Debería ser de unos 100x100 pero puede ser más grande o más pequeño. Para el aprendizaje, también se puede usar un Rect (pero eso requiere un cambio en el código un poco hacia abajo)

Ahora, declaramos un Bitmap:

```
private Bitmap PLAYER_BMP = BitmapFactory.decodeResource(getResources(),
R.drawable.my_player_drawable);
```

En render, necesitamos dibujar este bitmap.

```
...
c.drawBitmap(PLAYER_BMP, x, y, null);
...
```

ANTES DE LANZAR todavía hay algunas cosas por hacer

Necesitamos un booleano primero:

```
boolean up = false;
```

en onTouchEvent, agregamos:

```
if(ev.getAction() == MotionEvent.ACTION_DOWN){
    up = true;
}else if(ev.getAction() == MotionEvent.ACTION_UP){
    up = false;
}
```

Y en tick necesitamos esto para mover al jugador:

```
if(up){
    velY -=1;
}
else{
    velY +=1;
}
if(velY >14)velY = 14;
if(velY <-14)velY = -14;
y += velY *2;
```

Y ahora necesitamos esto en init:

```
WindowManager wm = (WindowManager) c.getSystemService(Context.WINDOW_SERVICE);
Display display = wm.getDefaultDisplay();
Point size = new Point();
display.getSize(size);
WIDTH = size.x;
HEIGHT = size.y;
y = HEIGHT/ 2 - PLAYER_BMP.getHeight();
```

Y necesitamos estas variables:

```
public static int WIDTH, HEIGHT;
```

En este punto, el juego es ejecutable. Lo que significa que puedes lanzarlo y probarlo.

Ahora deberías tener una imagen de jugador o rect subiendo y bajando la pantalla. El jugador puede ser creado como una clase personalizada si es necesario. Luego, todas las cosas relacionadas con el jugador se pueden mover a esa clase y usar una instancia de esa clase para mover, renderizar y hacer otra lógica.

Ahora, como probablemente viste bajo prueba, se sale de la pantalla. Así que tenemos que limitarlo.

Primero, necesitamos declarar el Rect:

```
private Rect screen;
```

En init, después de inicializar ancho y alto, creamos un nuevo rect que es la pantalla.

```
screen = new Rect(0,0,WIDTH,HEIGHT);
```

Ahora necesitamos otro rect en la forma de un método:

```
private Rect getPlayerBound(){
    return new Rect(x, y, x + PLAYER_BMP.getWidth(), y + PLAYER_BMP.getHeight();
}
```

y en tic

```
if(!getPlayerBound().intersects(screen){
    gameOver = true;
}
```

La implementación de gameOVer también se puede utilizar para mostrar el inicio de un juego.

Otros aspectos de un juego digno de mención:

Guardando (actualmente falta en la documentación)

Lea Desarrollo de juegos para Android en línea:
https://riptutorial.com/es/android/topic/10011/desarrollo-de-juegos-para-android

Capítulo 83: Descomprimir archivo en Android

Examples

Descomprimir archivo

```
private boolean unpackZip(String path, String zipname){
 InputStream is;
 ZipInputStream zis;
 try
 {
     String filename;
     is = new FileInputStream(path + zipname);
     zis = new ZipInputStream(new BufferedInputStream(is));
     ZipEntry ze;
     byte[] buffer = new byte[1024];
     int count;

     while ((ze = zis.getNextEntry()) != null){
         // zapis do souboru
         filename = ze.getName();

         // Need to create directories if not exists, or
         // it will generate an Exception...
         if (ze.isDirectory()) {
            File fmd = new File(path + filename);
            fmd.mkdirs();
            continue;
         }

         FileOutputStream fout = new FileOutputStream(path + filename);

         // cteni zipu a zapis
         while ((count = zis.read(buffer)) != -1){
             fout.write(buffer, 0, count);
         }

         fout.close();
         zis.closeEntry();
     }

     zis.close();
 }
 catch(IOException e){
     e.printStackTrace();
     return false;
 }

return true;}
```

Lea Descomprimir archivo en Android en línea:
https://riptutorial.com/es/android/topic/3927/descomprimir-archivo-en-android

Capítulo 84: Deslizamiento

Introducción

**** ADVERTENCIA Esta documentación no se mantiene y con frecuencia es inexacta ****

La documentación oficial de Glide es una fuente mucho mejor:

Para Glide v4, consulte http://bumptech.github.io/glide/ . Para Glide v3, consulte https://github.com/bumptech/glide/wiki .

Observaciones

Glide es un marco de administración de medios y carga de imágenes de código abierto rápido y eficiente para Android que envuelve la decodificación de medios, la memoria y el almacenamiento en caché de discos, y la agrupación de recursos en una interfaz simple y fácil de usar.

Glide admite la captura, decodificación y visualización de imágenes fijas de video, imágenes y GIF animados. Glide incluye una API flexible que permite a los desarrolladores conectarse a casi cualquier pila de red.
De manera predeterminada, Glide usa una pila personalizada basada en HttpUrlConnection , pero también incluye bibliotecas de utilidades conectadas al proyecto Volley de Google o a la biblioteca OkHttp de Square .

El objetivo principal de Glide es hacer que el desplazamiento de cualquier tipo de lista de imágenes sea lo más suave y rápido posible, pero Glide también es efectivo para casi cualquier caso en el que necesite buscar, cambiar el tamaño y mostrar una imagen remota.

El código fuente y la documentación adicional están disponibles en GitHub: https://github.com/bumptech/glide

Examples

Agrega Glide a tu proyecto

De la documentación oficial :

Con Gradle:

```
repositories {
  mavenCentral() // jcenter() works as well because it pulls from Maven Central
}

dependencies {
  compile 'com.github.bumptech.glide:glide:4.0.0'
  compile 'com.android.support:support-v4:25.3.1'
  annotationProcessor 'com.github.bumptech.glide:compiler:4.0.0'
```

```
}
```

Con Maven:

```
<dependency>
  <groupId>com.github.bumptech.glide</groupId>
  <artifactId>glide</artifactId>
  <version>4.0.0</version>
</dependency>
<dependency>
  <groupId>com.google.android</groupId>
  <artifactId>support-v4</artifactId>
  <version>r7</version>
</dependency>
<dependency>
  <groupId>com.github.bumptech.glide</groupId>
  <artifactId>compiler</artifactId>
  <version>4.0.0</version>
  <optional>true</optional>
</dependency>
```

Dependiendo de su configuración y uso de ProGuard (DexGuard), es posible que también deba incluir las siguientes líneas en su proguard.cfg (consulte la wiki de Glide para obtener más información):

```
-keep public class * implements com.bumptech.glide.module.GlideModule
-keep public class * extends com.bumptech.glide.AppGlideModule
-keep public enum com.bumptech.glide.load.resource.bitmap.ImageHeaderParser$**     {
  **[] $VALUES;
  public *;
}

# for DexGuard only
-keepresourcexmlelements manifest/application/meta-data@value=GlideModule
```

Cargando una imagen

ImageView

Para cargar una imagen desde una URL específica, Uri, ID de recurso o cualquier otro modelo en un `ImageView` :

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
String yourUrl = "http://www.yoururl.com/image.png";

Glide.with(context)
    .load(yourUrl)
    .into(imageView);
```

Para Uris, reemplace `yourUrl` con su Uri (`content://media/external/images/1`). Para Drawables, reemplace `yourUrl` con su ID de recurso (`R.drawable.image`).

RecyclerView y ListView

En ListView o RecyclerView, puede usar exactamente las mismas líneas:

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    MyViewHolder myViewHolder = (MyViewHolder) viewHolder;
    String currentUrl = myUrls.get(position);

    Glide.with(context)
        .load(currentUrl)
        .into(myViewHolder.imageView);
}
```

Si no desea iniciar una carga en `onBindViewHolder` , asegúrese de `clear()` cualquier `ImageView` Glide que esté administrando antes de modificar `ImageView` :

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    MyViewHolder myViewHolder = (MyViewHolder) viewHolder;
    String currentUrl = myUrls.get(position);

    if (TextUtils.isEmpty(currentUrl)) {
        Glide.clear(viewHolder.imageView);
        // Now that the view has been cleared, you can safely set your own resource
        viewHolder.imageView.setImageResource(R.drawable.missing_image);
    } else {
        Glide.with(context)
            .load(currentUrl)
            .into(myViewHolder.imageView);
    }
}
```

Transformación del círculo deslizante (Cargar imagen en una vista de imagen circular)

Crea una imagen de círculo con deslizamiento.

```
public class CircleTransform extends BitmapTransformation {

    public CircleTransform(Context context) {
        super(context);
    }

    @Override protected Bitmap transform(BitmapPool pool, Bitmap toTransform, int outWidth,
int outHeight) {
        return circleCrop(pool, toTransform);
    }

    private static Bitmap circleCrop(BitmapPool pool, Bitmap source) {
        if (source == null) return null;

        int size = Math.min(source.getWidth(), source.getHeight());
        int x = (source.getWidth() - size) / 2;
        int y = (source.getHeight() - size) / 2;
```

```
        Bitmap squared = Bitmap.createBitmap(source, x, y, size, size);

        Bitmap result = pool.get(size, size, Bitmap.Config.ARGB_8888);
        if (result == null) {
            result = Bitmap.createBitmap(size, size, Bitmap.Config.ARGB_8888);
        }

        Canvas canvas = new Canvas(result);
        Paint paint = new Paint();
        paint.setShader(new BitmapShader(squared, BitmapShader.TileMode.CLAMP,
BitmapShader.TileMode.CLAMP));
        paint.setAntiAlias(true);
        float r = size / 2f;
        canvas.drawCircle(r, r, r, paint);
        return result;
    }

    @Override public String getId() {
        return getClass().getName();
    }
}
```

Uso:

```
Glide.with(context)
    .load(yourimageurl)
    .transform(new CircleTransform(context))
    .into(userImageView);
```

Transformaciones por defecto

Glide incluye dos transformaciones predeterminadas, ajuste centro y centro de recorte.

Centro de ajuste:

```
Glide.with(context)
    .load(yourUrl)
    .fitCenter()
    .into(yourView);
```

El centro de ajuste realiza la misma transformación que ScaleType.FIT_CENTER de Android.

Cultivo central:

```
Glide.with(context)
    .load(yourUrl)
    .centerCrop()
    .into(yourView);
```

El recorte central realiza la misma transformación que el ScaleType.CENTER_CROP de Android.

Para más información, vea la wiki de Glide .

Imagen de esquinas redondeadas con objetivo Glide personalizado

Primero haga la clase de utilidad o use este método en la clase necesaria

```
public class UIUtils {
public static BitmapImageViewTarget getRoundedImageTarget(@NonNull final Context context,
@NonNull final ImageView imageView,
                                                          final float radius) {
    return new BitmapImageViewTarget(imageView) {
        @Override
        protected void setResource(final Bitmap resource) {
            RoundedBitmapDrawable circularBitmapDrawable =
                    RoundedBitmapDrawableFactory.create(context.getResources(), resource);
            circularBitmapDrawable.setCornerRadius(radius);
            imageView.setImageDrawable(circularBitmapDrawable);
        }
    };
}
```

Cargando imagen:

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Aunque usas asBitmap () las animaciones se eliminarán. Puedes usar tu propia animación en este lugar usando el método animate ().

Ejemplo con fundido similar a la animación predeterminada de deslizamiento.

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .animate(R.anim.abc_fade_in)
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Tenga en cuenta que esta animación es un recurso privado de la biblioteca de soporte; no se recomienda su uso, ya que puede cambiarse o incluso eliminarse.

Tenga en cuenta que también necesita tener una biblioteca de soporte para usar RoundedBitmapDrawableFactory

Precarga de imagenes

Para precargar imágenes remotas y asegurarse de que la imagen solo se descarga una vez:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE)
    .preload();
```

Entonces:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE) // ALL works here too
    .into(imageView);
```

Para precargar imágenes locales y asegurarse de que haya una copia transformada en la memoria caché del disco (y quizás en la memoria caché):

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // Or whatever transformation you want
    .preload(200, 200); // Or whatever width and height you want
```

Entonces:

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // You must use the same transformation as above
    .override(200, 200) // You must use the same width and height as above
    .into(imageView);
```

Marcador de posición y manejo de errores

Si desea agregar un Drawable que se muestra durante la carga, puede agregar un marcador de posición:

```
Glide.with(context)
    .load(yourUrl)
    .placeholder(R.drawable.placeholder)
    .into(imageView);
```

Si desea que se muestre un Drawable si la carga falla por algún motivo:

```
Glide.with(context)
    .load(yourUrl)
    .error(R.drawable.error)
    .into(imageView);
```

Si desea que se muestre un Drawable si proporciona un modelo nulo (URL, Uri, ruta de archivo, etc.):

```
Glide.with(context)
    .load(maybeNullUrl)
    .fallback(R.drawable.fallback)
    .into(imageView);
```

Cargar imagen en un ImageView circular sin transformaciones personalizadas.

Cree un BitmapImageViewTarget personalizado para cargar la imagen en:

```
public class CircularBitmapImageViewTarget extends BitmapImageViewTarget
{
    private Context context;
    private ImageView imageView;

    public CircularBitmapImageViewTarget(Context context, ImageView imageView)
    {
        super(imageView);
        this.context = context;
        this.imageView = imageView;
    }

    @Override
    protected void setResource(Bitmap resource)
    {
        RoundedBitmapDrawable bitmapDrawable =
RoundedBitmapDrawableFactory.create(context.getResources(), resource);
        bitmapDrawable.setCircular(true);
        imageView.setImageDrawable(bitmapDrawable);
    }
}
```

Uso:

```
Glide
    .with(context)
    .load(yourimageidentifier)
    .asBitmap()
    .into(new CircularBitmapImageViewTarget(context, imageView));
```

Falló la carga de la imagen de Glide

```
Glide
        .with(context)
        .load(currentUrl)
        .into(new BitmapImageViewTarget(profilePicture) {
    @Override
    protected void setResource(Bitmap resource) {
        RoundedBitmapDrawable circularBitmapDrawable =
                RoundedBitmapDrawableFactory.create(context.getResources(), resource);
        circularBitmapDrawable.setCornerRadius(radius);
        imageView.setImageDrawable(circularBitmapDrawable);
    }

    @Override
    public void onLoadFailed(@NonNull Exception e, Drawable errorDrawable) {
        super.onLoadFailed(e, SET_YOUR_DEFAULT_IMAGE);
        Log.e(TAG, e.getMessage(), e);
    }
});
```

Aquí, en `SET_YOUR_DEFAULT_IMAGE` , puede establecer cualquier `Drawable` predeterminado. Esta imagen se mostrará si falla la carga de la imagen.

Lea Deslizamiento en línea: https://riptutorial.com/es/android/topic/1091/deslizamiento

Capítulo 85: Deslizar para actualizar

Sintaxis

1. **setColorSchemeResources** establece los colores del **indicador** SwipeToRefreshLayout
2. **setOnRefreshListener** establece qué hacer cuando se **desliza el** diseño
3. **app: layout_behavior = "@ string / appbar_scrolling_view_behavior"** si tiene una barra de herramientas con su diseño, agregue esto con scrollflags en la barra de herramientas y la barra de herramientas se deslizará hacia arriba mientras se desplaza hacia abajo y se desliza hacia arriba mientras se desplaza hacia arriba.

Examples

Deslizar para actualizar con RecyclerView

Para agregar un diseño de **Swipe To Refresh** con un **RecyclerView,** agregue lo siguiente a su archivo de diseño de Actividad / Fragmento:

```
<android.support.v4.widget.SwipeRefreshLayout
    android:id="@+id/refresh_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior">

    <android.support.v7.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:scrollbars="vertical" />

</android.support.v4.widget.SwipeRefreshLayout>
```

En su Actividad / Fragmento, agregue lo siguiente para inicializar **SwipeToRefreshLayout** :

```
    SwipeRefreshLayout mSwipeRefreshLayout = (SwipeRefreshLayout)
findViewById(R.id.refresh_layout);
    mSwipeRefreshLayout.setColorSchemeResources(R.color.green_bg,
            android.R.color.holo_green_light,
            android.R.color.holo_orange_light,
            android.R.color.holo_red_light);

    mSwipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
        @Override
        public void onRefresh() {
            // Execute code when refresh layout swiped
        }
    });
```

Cómo agregar Swipe-to-Refresh a tu aplicación

Asegúrese de que la siguiente dependencia se agregue al archivo `build.gradle` su aplicación en las dependencias:

```
compile 'com.android.support:support-core-ui:24.2.0'
```

Luego agrega el `SwipeRefreshLayout` en tu diseño:

```
<android.support.v4.widget.SwipeRefreshLayout
        android:id="@+id/swipe_refresh_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

     <!-- place your view here -->

</android.support.v4.widget.SwipeRefreshLayout>
```

Finalmente, implemente el oyente `SwipeRefreshLayout.OnRefreshListener` .

```
mSwipeRefreshLayout = (SwipeRefreshLayout) findViewById(R.id.swipe_refresh_layout);
mSwipeRefreshLayout.setOnRefreshListener(new OnRefreshListener() {
    @Override
    public void onRefresh() {
         // your code
    }
});
```

Lea Deslizar para actualizar en línea: https://riptutorial.com/es/android/topic/5241/deslizar-para-actualizar

Capítulo 86: Detección de gestos

Observaciones

Documentación oficial: detección de gestos comunes

Examples

Detección de deslizamiento

```
public class OnSwipeListener implements View.OnTouchListener {

    private final GestureDetector gestureDetector;

    public OnSwipeListener(Context context) {
        gestureDetector = new GestureDetector(context, new GestureListener());
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        return gestureDetector.onTouchEvent(event);
    }

    private final class GestureListener extends GestureDetector.SimpleOnGestureListener {

        private static final int SWIPE_VELOCITY_THRESHOLD = 100;
        private static final int SWIPE_THRESHOLD = 100;

        @Override
        public boolean onDown(MotionEvent e) {
            return true;
        }

        @Override
        public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float
velocityY) {
            float diffY = e2.getY() - e1.getY();
            float diffX = e2.getX() - e1.getX();
            if (Math.abs(diffX) > Math.abs(diffY)) {
                if (Math.abs(diffX) > SWIPE_THRESHOLD && Math.abs(velocityX) >
SWIPE_VELOCITY_THRESHOLD) {
                    if (diffX > 0) {
                        onSwipeRight();
                    } else {
                        onSwipeLeft();
                    }
                }
            } else if (Math.abs(diffY) > SWIPE_THRESHOLD && Math.abs(velocityY) >
SWIPE_VELOCITY_THRESHOLD) {
                if (diffY > 0) {
                    onSwipeBottom();
                } else {
                    onSwipeTop();
                }
            }
```

```
            return true;
        }
    }

    public void onSwipeRight() {
    }

    public void onSwipeLeft() {
    }

    public void onSwipeTop() {
    }

    public void onSwipeBottom() {
    }

}
```

Aplicado a una vista ...

```
view.setOnTouchListener(new OnSwipeListener(context) {
    public void onSwipeTop() {
        Log.d("OnSwipeListener", "onSwipeTop");
    }
    public void onSwipeRight() {
        Log.d("OnSwipeListener", "onSwipeRight");
    }
    public void onSwipeLeft() {
        Log.d("OnSwipeListener", "onSwipeLeft");
    }
    public void onSwipeBottom() {
        Log.d("OnSwipeListener", "onSwipeBottom");
    }

});
```

Detección de gestos básicos

```
public class GestureActivity extends Activity implements
        GestureDetector.OnDoubleTapListener,
        GestureDetector.OnGestureListener {

    private GestureDetector mGestureDetector;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mGestureDetector = new GestureDetector(this, this);
        mGestureDetector.setOnDoubleTapListener(this);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event){
        mGestureDetector.onTouchEvent(event);
        return super.onTouchEvent(event);
    }
```

```
    @Override
    public boolean onDown(MotionEvent event) {
        Log.d("GestureDetector","onDown");
        return true;
    }

    @Override
    public boolean onFling(MotionEvent event1, MotionEvent event2, float velocityX, float
velocityY) {
        Log.d("GestureDetector","onFling");
        return true;
    }

    @Override
    public void onLongPress(MotionEvent event) {
        Log.d("GestureDetector","onLongPress");
    }

    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY)
{
        Log.d("GestureDetector","onScroll");
        return true;
    }

    @Override
    public void onShowPress(MotionEvent event) {
        Log.d("GestureDetector","onShowPress");
    }

    @Override
    public boolean onSingleTapUp(MotionEvent event) {
        Log.d("GestureDetector","onSingleTapUp");
        return true;
    }

    @Override
    public boolean onDoubleTap(MotionEvent event) {
        Log.d("GestureDetector","onDoubleTap");
        return true;
    }

    @Override
    public boolean onDoubleTapEvent(MotionEvent event) {
        Log.d("GestureDetector","onDoubleTapEvent");
        return true;
    }

    @Override
    public boolean onSingleTapConfirmed(MotionEvent event) {
        Log.d("GestureDetector","onSingleTapConfirmed");
        return true;
    }

}
```

Lea Detección de gestos en línea: https://riptutorial.com/es/android/topic/4711/deteccion-de-gestos

Capítulo 87: Detect Shake Event en Android

Examples

Shake Detector en el ejemplo de Android

```
public class ShakeDetector implements SensorEventListener {


    private static final float SHAKE_THRESHOLD_GRAVITY = 2.7F;
    private static final int SHAKE_SLOP_TIME_MS = 500;
    private static final int SHAKE_COUNT_RESET_TIME_MS = 3000;

    private OnShakeListener mListener;
    private long mShakeTimestamp;
    private int mShakeCount;

    public void setOnShakeListener(OnShakeListener listener) {
        this.mListener = listener;
    }

    public interface OnShakeListener {
        public void onShake(int count);
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // ignore
    }

    @Override
    public void onSensorChanged(SensorEvent event) {

        if (mListener != null) {
            float x = event.values[0];
            float y = event.values[1];
            float z = event.values[2];

            float gX = x / SensorManager.GRAVITY_EARTH;
            float gY = y / SensorManager.GRAVITY_EARTH;
            float gZ = z / SensorManager.GRAVITY_EARTH;

            // gForce will be close to 1 when there is no movement.
            float gForce = FloatMath.sqrt(gX * gX + gY * gY + gZ * gZ);

            if (gForce > SHAKE_THRESHOLD_GRAVITY) {
                final long now = System.currentTimeMillis();
                // ignore shake events too close to each other (500ms)
                if (mShakeTimestamp + SHAKE_SLOP_TIME_MS > now) {
                    return;
                }

                // reset the shake count after 3 seconds of no shakes
                if (mShakeTimestamp + SHAKE_COUNT_RESET_TIME_MS < now) {
                    mShakeCount = 0;
                }
```

```
                mShakeTimestamp = now;
                mShakeCount++;

                mListener.onShake(mShakeCount);
            }
        }
    }
}
```

Usando detección de sacudidas sísmicas

Seismic es una biblioteca de detección de sacudidas de dispositivos Android de Square. Para usarlo solo comienza a escuchar los eventos de shake que emite.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    sm = (SensorManager) getSystemService(SENSOR_SERVICE);
    sd = new ShakeDetector(() -> { /* react to detected shake */ });
}

@Override
protected void onResume() {
    sd.start(sm);
}

@Override
protected void onPause() {
    sd.stop();
}
```

Para definir un umbral de aceleración diferente, use `sd.setSensitivity(sensitivity)` con una `sensitivity` de `SENSITIVITY_LIGHT` , `SENSITIVITY_MEDIUM` , `SENSITIVITY_HARD` o cualquier otro valor entero razonable. Los valores predeterminados dados van desde *11* a *15* .

Instalación

```
compile 'com.squareup:seismic:1.0.2'
```

Lea Detect Shake Event en Android en línea: https://riptutorial.com/es/android/topic/4501/detect-shake-event-en-android

Capítulo 88: Diálogo

Parámetros

Línea	Descripción
espectáculo();	Muestra el dialogo
setContentView (R.layout.yourlayout);	establece el ContentView del diálogo a su diseño personalizado.
despedir()	Cierra el dialogo

Observaciones

- El diálogo en el primer ejemplo (Diálogo) no necesita llamar a `show()` cuando se crea como se maneja en el constructor
- Los diálogos de alerta deben construirse a través de una nueva instancia de la clase `AlertDialog.Builder()` . Siguiendo el patrón del generador, todos los miembros de AlertDialog.Builder pueden ser encadenados en un método para "construir" la instancia de diálogo.
- El constructor del cuadro de diálogo de alerta puede `show()` directamente `show()` el cuadro de diálogo; no es necesario llamar a `create()` luego a `show()` en la instancia de AlertDialog

Examples

Diálogo de alerta

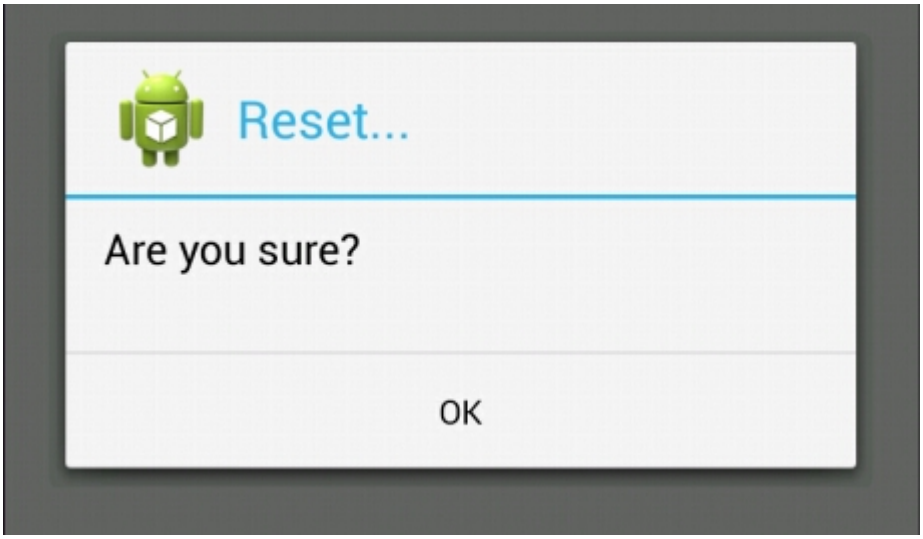
```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
                MainActivity.this);

        alertDialogBuilder.setTitle("Title Dialog");
        alertDialogBuilder
                .setMessage("Message Dialog")
                .setCancelable(true)
                .setPositiveButton("Yes",
                        new DialogInterface.OnClickListener() {

                            public void onClick(DialogInterface dialog, int arg1) {
                                // Handle Positive Button

                            }
                        })
                .setNegativeButton("No",
                        new DialogInterface.OnClickListener() {
```

```
                            public void onClick(DialogInterface dialog, int arg1) {
                                // Handle Negative Button
                                dialog.cancel();
                            }
                        });

        AlertDialog alertDialog = alertDialogBuilder.create();
        alertDialog.show();
```

Un diálogo de alerta básica

```
    AlertDialog.Builder builder = new AlertDialog.Builder(context);
    //Set Title
    builder.setTitle("Reset...")
            //Set Message
            .setMessage("Are you sure?")
            //Set the icon of the dialog
            .setIcon(drawable)
            //Set the positive button, in this case, OK, which will dismiss the dialog and do
everything in the onClick method
            .setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    // Reset
                }
            });
    AlertDialog dialog = builder.create();
    //Now, any time you can call on:
    dialog.show();
    //So you can show the dialog.
```

Ahora este código logrará esto:



(*Fuente de la imagen: WikiHow*)

Selector de fecha dentro de DialogFragment

xml del diálogo:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <DatePicker
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/datePicker"
        android:layout_gravity="center_horizontal"
        android:calendarViewShown="false"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="ACCEPT"
        android:id="@+id/buttonAccept" />

</LinearLayout>
```

Clase de diálogo:

```
public class ChooseDate extends DialogFragment implements View.OnClickListener {

    private DatePicker datePicker;
    private Button acceptButton;

    private boolean isDateSetted = false;
    private int year;
    private int month;
    private int day;

    private DateListener listener;

    public interface DateListener {
        onDateSelected(int year, int month, int day);
    }

    public ChooseDate(){}

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.dialog_year_picker, container);

        getDialog().setTitle(getResources().getString("TITLE"));

        datePicker = (DatePicker) rootView.findViewById(R.id.datePicker);
        acceptButton = (Button) rootView.findViewById(R.id.buttonAccept);
        acceptButton.setOnClickListener(this);

        if (isDateSetted) {
            datePicker.updateDate(year, month, day);
        }

        return rootView;
    }

    @Override
    public void onClick(View v) {
        switch(v.getId()){
```

```
            case R.id.buttonAccept:
                int year = datePicker.getYear();
                int month = datePicker.getMonth() + 1; // months start in 0
                int day = datePicker.getDayOfMonth();

                listener.onDateSelected(year, month, day);
                break;
        }
        this.dismiss();
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        listener = (DateListener) context;
    }

    public void setDate(int year, int month, int day) {

        this.year = year;
        this.month = month;
        this.day = day;
        this.isDateSetted = true;
    }

}
```

Actividad llamando al diálogo:

```
public class MainActivity extends AppCompatActivity implements ChooseDate.DateListener{

    private int year;
    private int month;
    private int day;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        private void showDateDialog();
    }

    private void showDateDialog(){
        ChooseDate pickDialog = new ChooseDate();
        // We could set a date
        //  pickDialog.setDate(23, 10, 2016);
        pickDialog.show(getFragmentManager(), "");
    }

    @Override
    onDateSelected(int year, int month, int day){
        this.day = day;
        this.month = month;
        this.year = year;
    }
}
```

DatePickerDialog

`DatePickerDialog` es la forma más sencilla de usar `DatePicker` , ya que puede mostrar el diálogo en cualquier lugar de su aplicación. No tienes que implementar tu propio diseño con el widget `DatePicker` .

Cómo mostrar el diálogo:

```
DatePickerDialog datePickerDialog = new DatePickerDialog(context, listener, year, month, day);
datePickerDialog.show();
```

Puede obtener el widget `DataPicker` desde el cuadro de diálogo de arriba, para obtener acceso a más funciones y, por ejemplo, establecer la fecha mínima en milisegundos:

```
DatePicker datePicker = datePickerDialog.getDatePicker();
datePicker.setMinDate(System.currentTimeMillis());
```

Selector de fechas

`DatePicker` permite al usuario elegir la fecha. Cuando creamos una nueva instancia de `DatePicker` , podemos establecer la fecha inicial. Si no establecemos la fecha inicial, la fecha actual se establecerá de forma predeterminada.

Podemos mostrar `DatePicker` al usuario utilizando `DatePickerDialog` o creando nuestro propio diseño con el widget `DatePicker` .

También podemos limitar el rango de fechas, que el usuario puede elegir.

Estableciendo la fecha mínima en milisegundos.

```
//In this case user can pick date only from future
datePicker.setMinDate(System.currentTimeMillis());
```

Al establecer la fecha máxima en milisegundos

```
//In this case user can pick date only, before following week.
datePicker.setMaxDate(System.currentTimeMillis() + TimeUnit.DAYS.toMillis(7));
```

Para recibir información, sobre qué fecha fue seleccionada por el usuario, tenemos que usar `Listener` .

Si estamos utilizando `DatePickerDialog` , podemos establecer `OnDateSetListener` en el constructor cuando estamos creando una nueva instancia de `DatePickerDialog` :

Ejemplo de uso de `DatePickerDialog`

```
public class SampleActivity extends AppCompatActivity implements
DatePickerDialog.OnDateSetListener {
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ...
    }

    private void showDatePicker() {
        //We need calendar to set current date as initial date in DatePickerDialog.
        Calendar calendar = new GregorianCalendar(Locale.getDefault());
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH);
        int day = calendar.get(Calendar.DAY_OF_MONTH);

        DatePickerDialog datePickerDialog = new DatePickerDialog(this, this, year, month,
day);
        datePickerDialog.show();
    }

    @Override
    public void onDateSet(DatePicker datePicker, int year, int month, int day) {

    }
}
```

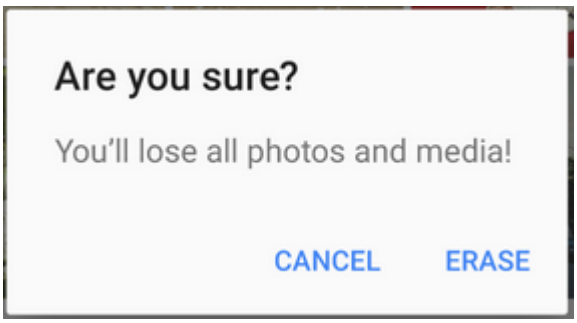
De lo contrario, si estamos creando nuestro propio diseño con el widget `DatePicker` , también tenemos que crear nuestro propio oyente como se muestra en otro ejemplo.

Adición de Material Design AlertDialog a su aplicación usando Appcompat

`AlertDialog` es una subclase de `Dialog` que puede mostrar uno, dos o tres botones. Si solo desea mostrar una cadena en este cuadro de diálogo, use el método `setMessage()` .

El paquete `AlertDialog` de `android.app` muestra de manera diferente en diferentes versiones del sistema operativo Android.

La biblioteca de aplicaciones de Android V7 proporciona una implementación de `AlertDialog` que se mostrará con Material Design en todas las versiones compatibles del sistema operativo Android, como se muestra a continuación:



Primero necesita agregar la biblioteca V7 Appcompat a su proyecto. Puedes hacer esto en el archivo build.gradle de nivel de aplicación:

```
dependencies {
```

```
    compile 'com.android.support:appcompat-v7:24.2.1'
    //........
}
```

Asegúrese de importar la clase correcta:

```
import android.support.v7.app.AlertDialog;
```

Luego crea AlertDialog como este:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Are you sure?");
builder.setMessage("You'll lose all photos and media!");
builder.setPositiveButton("ERASE", null);
builder.setNegativeButton("CANCEL", null);
builder.show();
```

ListView en AlertDialog

Siempre podemos usar `ListView` o `RecyclerView` para seleccionar de la lista de elementos, pero si tenemos una pequeña cantidad de opciones y entre esas opciones queremos que el usuario seleccione una, podemos usar `AlertDialog.Builder setAdapter` .

```
    private void showDialog()
    {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Choose any item");

        final List<String> lables = new ArrayList<>();
        lables.add("Item 1");
        lables.add("Item 2");
        lables.add("Item 3");
        lables.add("Item 4");

        ArrayAdapter<String> dataAdapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line, lables);
        builder.setAdapter(dataAdapter, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(MainActivity.this,"You have selected " +
lables.get(which),Toast.LENGTH_LONG).show();
            }
        });
        AlertDialog dialog = builder.create();
        dialog.show();
    }
```

Quizás, si no necesitamos un `ListView` particular, podemos usar una forma básica:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Select an item")
       .setItems(R.array.your_array, new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int which) {
               // The 'which' argument contains the index position of the selected item
```

```
                Log.v(TAG, "Selected item on position " + which);
            }
        });
builder.create().show();
```

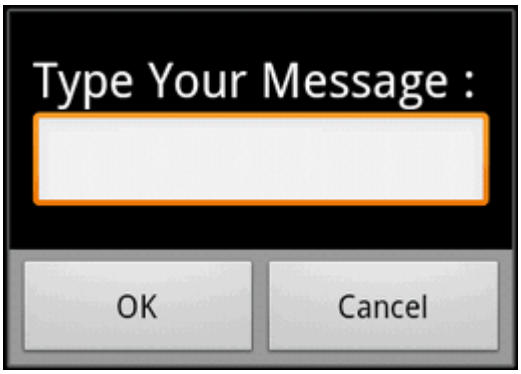
Cuadro de diálogo de alerta personalizada con EditText

```
void alertDialogDemo() {
        // get alert_dialog.xml view
        LayoutInflater li = LayoutInflater.from(getApplicationContext());
        View promptsView = li.inflate(R.layout.alert_dialog, null);

        AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
                getApplicationContext());

        // set alert_dialog.xml to alertdialog builder
        alertDialogBuilder.setView(promptsView);

        final EditText userInput = (EditText) promptsView.findViewById(R.id.etUserInput);

        // set dialog message
        alertDialogBuilder
                .setCancelable(false)
                .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        // get user input and set it to result
                        // edit text
                        Toast.makeText(getApplicationContext(), "Entered:
"+userInput.getText().toString(), Toast.LENGTH_LONG).show();
                    }
                })
                .setNegativeButton("Cancel",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                dialog.cancel();
                            }
                        });

        // create alert dialog
        AlertDialog alertDialog = alertDialogBuilder.create();

        // show it
        alertDialog.show();
    }
```

Archivo Xml: res / layout / alert_dialog.xml

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Type Your Message : "
    android:textAppearance="?android:attr/textAppearanceLarge" />

<EditText
    android:id="@+id/etUserInput"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
```

```
    <requestFocus />

</EditText>
```



Cuadro de diálogo personalizado a pantalla completa sin fondo y sin título

en `styles.xml` agrega tu estilo personalizado:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppBaseTheme" parent="@android:style/Theme.Light.NoTitleBar.Fullscreen">
    </style>
</resources>
```

Cree su diseño personalizado para el diálogo: `fullscreen.xml` :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

</RelativeLayout>
```

Luego, en el archivo java puede usarlo para una Actividad o un Diálogo, etc.

```
import android.app.Activity;
import android.app.Dialog;
import android.os.Bundle;

public class FullscreenActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //You can set no content for the activity.
        Dialog mDialog = new Dialog(this, R.style.AppBaseTheme);
        mDialog.setContentView(R.layout.fullscreen);
        mDialog.show();
    }
}
```

Cuadro de diálogo de alerta con título de multilínea

El método setCustomTitle () de AlertDialog.Builder le permite especificar una vista arbitraria que se usará para el título del diálogo. Un uso común de este método es crear un diálogo de alerta que tenga un título largo.

```
AlertDialog.Builder builder = new AlertDialog.Builder(context, Theme_Material_Light_Dialog);
builder.setCustomTitle(inflate(context, R.layout.my_dialog_title, null))
       .setView(inflate(context, R.layout.my_dialog, null))
       .setPositiveButton("OK", null);

Dialog dialog = builder.create();
dialog.show();
```

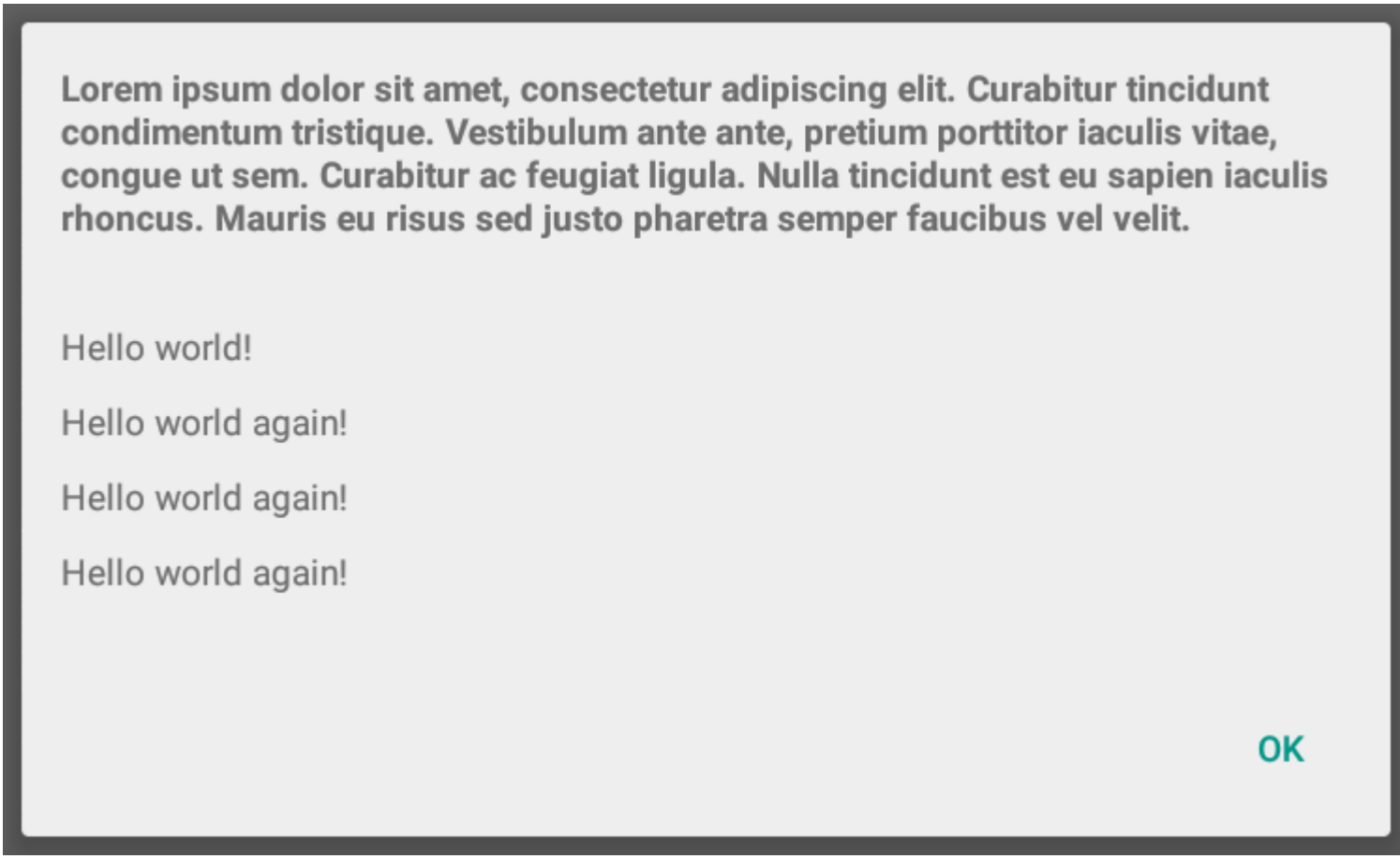
my_dialog_title.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:padding="16dp">

    <TextView
        style="@android:style/TextAppearance.Small"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur
tincidunt condimentum tristique. Vestibulum ante ante, pretium porttitor
iaculis vitae, congue ut sem. Curabitur ac feugiat ligula. Nulla
tincidunt est eu sapien iaculis rhoncus. Mauris eu risus sed justo
pharetra semper faucibus vel velit."
        android:textStyle="bold"/>

</LinearLayout>
```

my_dialog.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp"
        android:scrollbars="vertical">

        <TextView
            style="@android:style/TextAppearance.Small"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:text="Hello world!"/>

        <TextView
            style="@android:style/TextAppearance.Small"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:text="Hello world again!"/>

        <TextView
            style="@android:style/TextAppearance.Small"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:text="Hello world again!"/>

        <TextView

            style="@android:style/TextAppearance.Small"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:text="Hello world again!"/>

    </LinearLayout>
</ScrollView>
```



Lea Diálogo en línea: https://riptutorial.com/es/android/topic/1225/dialogo

Capítulo 89: Dibujables

Examples

Tintar un dibujo

Un dibujo puede ser teñido de un color determinado. Esto es útil para admitir diferentes temas dentro de su aplicación y para reducir la cantidad de archivos de recursos dibujables.

Usando APIs de framework en SDK 21+:

```
Drawable d = context.getDrawable(R.drawable.ic_launcher);
d.setTint(Color.WHITE);
```

Usando la biblioteca android.support.v4 en SDK 4+:

```
//Load the untinted resource
final Drawable drawableRes = ContextCompat.getDrawable(context, R.drawable.ic_launcher);
//Wrap it with the compatibility library so it can be altered
Drawable tintedDrawable = DrawableCompat.wrap(drawableRes);
//Apply a coloured tint
DrawableCompat.setTint(tintedDrawable, Color.WHITE);
//At this point you may use the tintedDrawable just as you usually would
//(and drawableRes can be discarded)

//NOTE: If your original drawableRes was in use somewhere (i.e. it was the result of
//a call to a `getBackground()` method then at this point you still need to replace
//the background. setTint does *not* alter the instance that drawableRes points to,
//but instead creates a new drawable instance
```

Tenga en cuenta que `int color` **no se** refiere a un recurso de color, sin embargo, no está limitado a los colores definidos en la clase 'Color'. Cuando tenga un color definido en su XML que desee utilizar, primero debe obtener su valor.

Puede reemplazar los usos de `Color.WHITE` utilizando los métodos a continuación

Al apuntar a las API más antiguas:

```
getResources().getColor(R.color.your_color);
```

O en nuevos objetivos:

```
ContextCompat.getColor(context, R.color.your_color);
```

Hacer ver con esquinas redondeadas

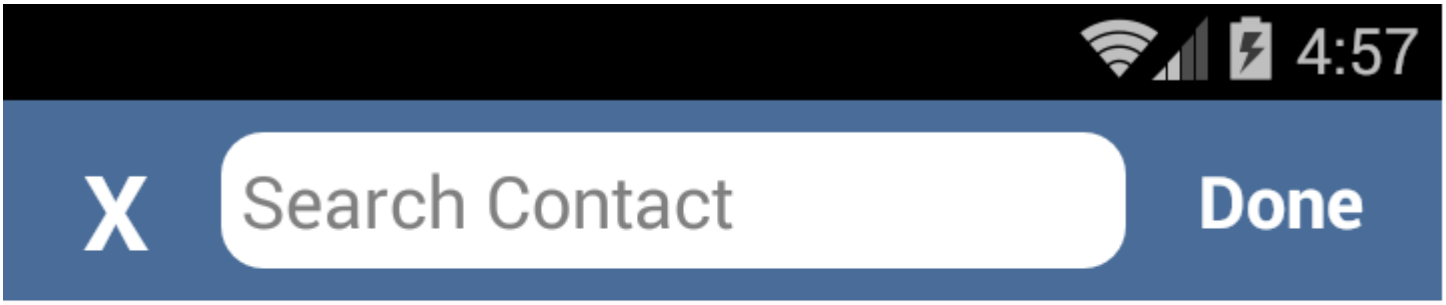
Crear el archivo llamado **estirable** con **custom_rectangle.xml** en la carpeta **dibujable:**

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <solid android:color="@android:color/white" />

    <corners android:radius="10dip" />

    <stroke
        android:width="1dp"
        android:color="@android:color/white" />

</shape>
```

Ahora aplique el **fondo del rectángulo** en la **vista** :

```
mView.setBackGround(R.drawlable.custom_rectangle);
```

Captura de pantalla de referencia:



Vista circular

Para una Vista circular (en este caso, `TextView`) cree un round_view.xml **drawble** en la carpeta **drawble** :

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="#FAA23C" />
    <stroke android:color="#FFF" android:width="2dp" />
</shape>
```

Asigna el dibujo a la vista:

```
    <TextView
        android:id="@+id/game_score"
```

```
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="@drawable/round_score"
        android:padding="6dp"
        android:text="100"
        android:textColor="#fff"
        android:textSize="20sp"
        android:textStyle="bold"
        android:gravity="center" />
```

Ahora debería verse como el círculo naranja:



Dibujo personalizable

Amplíe su clase con Drawable y anule estos métodos

```
public class IconDrawable extends Drawable {
    /**
     * Paint for drawing the shape
     */
    private Paint paint;
    /**
     * Icon drawable to be drawn to the center of the shape
     */
    private Drawable icon;
    /**
     * Desired width and height of icon
     */
    private int desiredIconHeight, desiredIconWidth;

    /**
     * Public constructor for the Icon drawable
     *
     * @param icon            pass the drawable of the icon to be drawn at the center
     * @param backgroundColor background color of the shape
     */
    public IconDrawable(Drawable icon, int backgroundColor) {
        this.icon = icon;
        paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColor(backgroundColor);
        desiredIconWidth = 50;
        desiredIconHeight = 50;
    }

    @Override
```

```
    public void draw(Canvas canvas) {
        //if we are setting this drawable to a 80dpX80dp imageview
        //getBounds will return that measurements,we can draw according to that width.
        Rect bounds = getBounds();
        //drawing the circle with center as origin and center distance as radius
        canvas.drawCircle(bounds.centerX(), bounds.centerY(), bounds.centerX(), paint);
        //set the icon drawable's bounds to the center of the shape
        icon.setBounds(bounds.centerX() - (desiredIconWidth / 2), bounds.centerY() -
(desiredIconHeight / 2), (bounds.centerX() - (desiredIconWidth / 2)) + desiredIconWidth,
(bounds.centerY() - (desiredIconHeight / 2)) + desiredIconHeight);
        //draw the icon to the bounds
        icon.draw(canvas);

    }

    @Override
    public void setAlpha(int alpha) {
        //sets alpha to your whole shape
        paint.setAlpha(alpha);
    }

    @Override
    public void setColorFilter(ColorFilter colorFilter) {
       //sets color filter to your whole shape
        paint.setColorFilter(colorFilter);
    }

    @Override
    public int getOpacity() {
        //give the desired opacity of the shape
        return PixelFormat.TRANSLUCENT;
    }
}
```

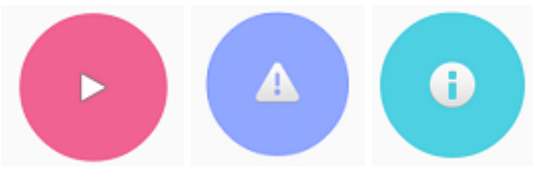
Declara un ImageView en tu diseño

```
<ImageView
   android:layout_width="80dp"
   android:id="@+id/imageView"
   android:layout_height="80dp" />
```

Establezca su dibujo personalizable para el ImageView

```
 IconDrawable iconDrawable=new
IconDrawable(ContextCompat.getDrawable(this,android.R.drawable.ic_media_play),ContextCompat.getColor(th

 imageView.setImageDrawable(iconDrawable);
```

Captura de pantalla



Lea Dibujables en línea: https://riptutorial.com/es/android/topic/4841/dibujables

Capítulo 90: Dibujos vectoriales

Introducción

Como su nombre lo indica, los dibujos vectoriales se basan en gráficos vectoriales. Los gráficos vectoriales son una forma de describir elementos gráficos utilizando formas geométricas. Esto le permite crear un dibujo basado en un gráfico vectorial XML. Ahora no hay necesidad de diseñar imágenes de diferentes tamaños para mdpi, hdpi, xhdpi y etc. Con Vector Drawable, necesita crear la imagen solo una vez como un archivo xml y puede escalarla para todas las ppp y para diferentes dispositivos. Esto tampoco ahorra espacio sino que también simplifica el mantenimiento.

Parámetros

Parámetro	Detalles
`<vector>`	Utilizado para definir un vector dibujable.
`<group>`	Define un grupo de rutas o subgrupos, más información de transformación. Las transformaciones se definen en las mismas coordenadas que la ventana gráfica. Y las transformaciones se aplican en el orden de escala, rotar y luego traducir.
`<path>`	Define trayectos a dibujar.
`<clip-path>`	Define la ruta para ser el clip actual. Tenga en cuenta que la ruta del clip solo se aplica al grupo actual y sus hijos.

Observaciones

Actualizar el archivo **build.gradle** .

```
dependencies {
    ...
   compile 'com.android.support:appcompat-v7:23.2.1'
}
```

Si está utilizando la **versión 2.0 o superior** del **complemento Gradle** , agregue el siguiente código.

```
// Gradle Plugin 2.0+
 android {
   defaultConfig {
     vectorDrawables.useSupportLibrary = true
    }
 }
```

Si está utilizando **v1.5 o inferior** del **complemento Gradle** , agregue el siguiente código.

```
// Gradle Plugin 1.5
 android {
   defaultConfig {
     generatedDensities = []
  }

  // This is handled for you by the 2.0+ Gradle Plugin
  aaptOptions {
    additionalParameters "--no-version-vectors"
  }
 }
```

Lea las notas de la versión 23.2 de la biblioteca de soporte de Android para obtener más información.

NOTA: Incluso con *AppCompat* , Vector Drawables no funciona fuera de su aplicación en versiones anteriores de Android. Por ejemplo, no puede pasar vectores dibujables como iconos de notificación, ya que son manejados por el sistema y no por la aplicación. Ver esta respuesta para una solución.

Examples

Ejemplo de uso de VectorDrawable

Aquí hay un ejemplo de vector activo que realmente estamos usando en AppCompat:

res / drawable / ic_search.xml

```
<vector xmlns:android="..."
        android:width="24dp"
        android:height="24dp"
        android:viewportWidth="24.0"
        android:viewportHeight="24.0"
        android:tint="?attr/colorControlNormal">

    <path
        android:pathData="..."
        android:fillColor="@android:color/white"/>

</vector>
```

Usando este dibujable, un ejemplo de declaración de `ImageView` sería:

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@drawable/ic_search"/>
```

También puedes configurarlo en tiempo de ejecución:

```
ImageView iv = (ImageView) findViewById(...);
iv.setImageResource(R.drawable.ic_search);
```

El mismo atributo y las llamadas también funcionan para `ImageButton` .

Ejemplo de VectorDrawable xml

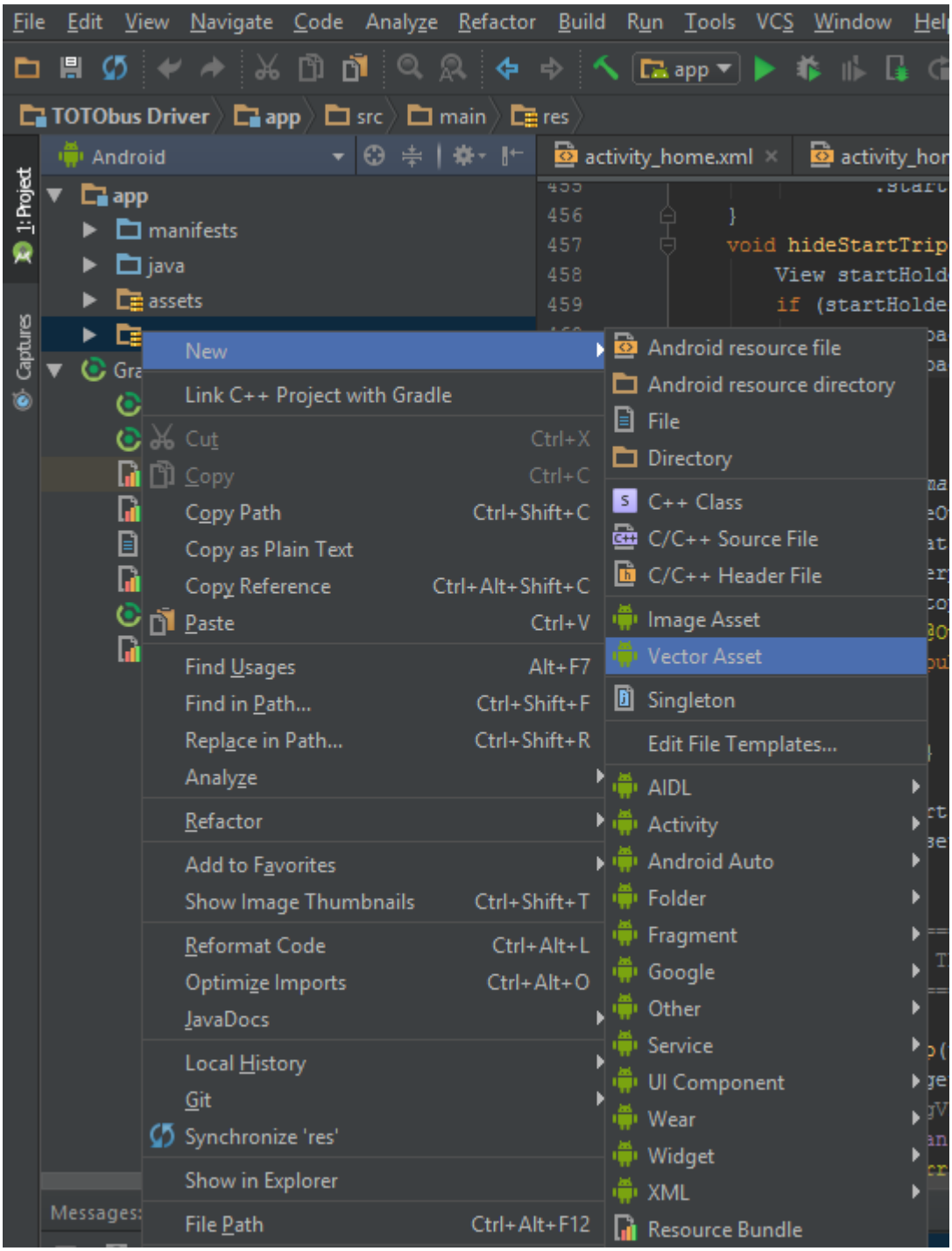
Aquí hay un `VectorDrawable` simple en este archivo **vectordrawable.xml** .

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="64dp"
    android:width="64dp"
    android:viewportHeight="600"
    android:viewportWidth="600" >
    <group
        android:name="rotationGroup"
        android:pivotX="300.0"
        android:pivotY="300.0"
        android:rotation="45.0" >
        <path
            android:name="v"
            android:fillColor="#000000"
            android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
    </group>
</vector>
```

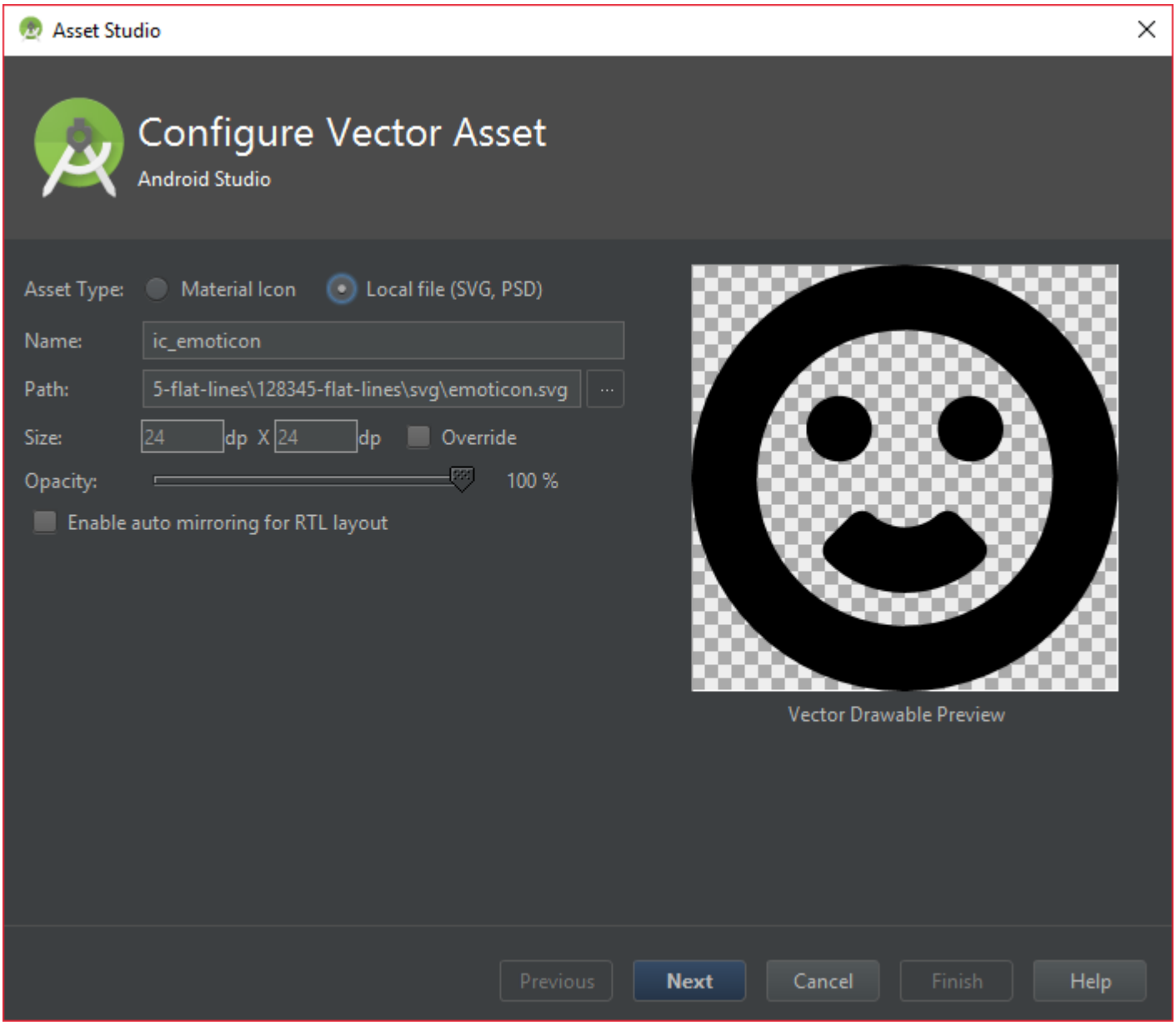
Importando archivo SVG como VectorDrawable

Puede importar un archivo *SVG* como `VectorDrawable` en Android Studio, siga estos pasos:

Haga clic con el botón derecho en la carpeta `res` y seleccione **nuevo** > **Vector Asset** .



Seleccione la opción **Archivo local** y busque su archivo .svg. Cambia las opciones a tu gusto y pulsa siguiente. Hecho.



Lea Dibujos vectoriales en línea: https://riptutorial.com/es/android/topic/8194/dibujos-vectoriales

Capítulo 91: Diseño de materiales

Introducción

Material Design es una guía completa para el diseño visual, de movimiento e interacción en plataformas y dispositivos.

Observaciones

También vea la publicación original del blog de Android que presenta la Biblioteca de soporte de diseño

Documentacion oficial

https://developer.android.com/design/material/index.html

Pautas para el diseño de materiales

https://material.io/guidelines

Otros recursos de diseño y bibliotecas.

https://design.google.com/resources/

Examples

Aplicar un tema de AppCompat

La biblioteca de soporte de AppCompat proporciona temas para crear aplicaciones con la especificación de Diseño de materiales . También se requiere un tema con un padre de `Theme.AppCompat` para que una Actividad extienda `AppCompatActivity` .

El primer paso es personalizar la paleta de colores de tu tema para colorear automáticamente tu aplicación.
En la aplicación `res/styles.xml` puede definir:

```
<!-- inherit from the AppCompat theme -->
<style name="AppTheme" parent="Theme.AppCompat">

    <!-- your app branding color for the app bar -->
    <item name="colorPrimary">#2196f3</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="colorPrimaryDark">#1976d2</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="colorAccent">#f44336</item>
</style>
```

En lugar de `Theme.AppCompat` , que tiene un fondo oscuro, también puede usar `Theme.AppCompat.Light` o `Theme.AppCompat.Light.DarkActionBar` .

Puedes personalizar el tema con tus propios colores. Las buenas elecciones se encuentran en la tabla de colores de especificación de diseño del material y en la paleta de materiales Los colores "500" son buenas opciones para el primario (azul 500 en este ejemplo); Elija "700" del mismo tono para el oscuro; y un tono de un tono diferente como el color de acento. El color primario se usa para la barra de herramientas de su aplicación y su entrada en la pantalla de información general (aplicaciones recientes), la variante más oscura para teñir la barra de estado y el color de acento para resaltar algunos controles.

Después de crear este tema, aplíquelo a su aplicación en `AndroidManifest.xml` y también aplique el tema a cualquier actividad en particular. Esto es útil para aplicar un tema `AppTheme.NoActionBar` , que le permite implementar configuraciones de barra de herramientas no predeterminadas.

```
<application android:theme="@style/AppTheme"
    ...>
    <activity
        android:name=".MainActivity"
        android:theme="@style/AppTheme" />
</application>
```

También puede aplicar temas a vistas individuales usando `android:theme` y un tema `ThemeOverlay` . Por ejemplo con una `Toolbar` :

```
<android.support.v7.widget.Toolbar
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:background="?attr/colorPrimary"
  android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />
```

o un `Button` :

```
<Button
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:theme="@style/MyButtonTheme"/>

<!-- res/values/themes.xml -->
<style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
    <item name="colorAccent">@color/my_color</item>
</style>
```

Agregar una barra de herramientas

Una `Toolbar` es una generalización de `ActionBar` para uso dentro de diseños de aplicaciones. Mientras que una `ActionBar` es tradicionalmente parte de `Activity's` decoración de la ventana opaca de una `Activity's` controlada por el marco, una `Toolbar` se puede colocar en cualquier nivel arbitrario de anidación dentro de una jerarquía de vistas. Se puede agregar realizando los siguientes pasos:

1. Asegúrese de que la siguiente dependencia se agregue al archivo **build.gradle de** su módulo (por ejemplo, la aplicación) en las dependencias:

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

2. Establezca el tema de su aplicación en uno que **no** tenga una `ActionBar` . Para hacerlo, edite su archivo **styles.xml** en `res/values` y configure un tema `Theme.AppCompat` .
   En este ejemplo, estamos usando `Theme.AppCompat.NoActionBar` como elemento principal de su `AppTheme` :

```
<style name="AppTheme" parent="Theme.AppCompat.NoActionBar">
    <item name="colorPrimary">@color/primary</item>
    <item name="colorPrimaryDark">@color/primaryDark</item>
    <item name="colorAccent">@color/accent</item>
</style>
```

   También puede usar `Theme.AppCompat.Light.NoActionBar` o `Theme.AppCompat.DayNight.NoActionBar` , o cualquier otro tema que no tenga inherentemente una `ActionBar`

3. Agregue la `Toolbar` de `Toolbar` a su diseño de actividad:

```
<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    android:elevation="4dp"/>
```

   Debajo de la `Toolbar` puede agregar el resto de su diseño.

4. En su `Activity` , configure la `Toolbar` como la `ActionBar` de `ActionBar` para esta `Activity` .
   Siempre que estés usando la biblioteca *appcompat* y una `AppCompatActivity` , `setSupportActionBar()` método `setSupportActionBar()` :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    final Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    //...
}
```

Después de realizar los pasos anteriores, puede utilizar el método `getSupportActionBar()` para manipular la `Toolbar` que se establece como la `ActionBar` .

Por ejemplo, puede establecer el título como se muestra a continuación:

```
getSupportActionBar().setTitle("Activity Title");
```

Por ejemplo, también puede configurar el título y el color de fondo como se muestra a continuación:

```
CharSequence title = "Your App Name";
SpannableString s = new SpannableString(title);
s.setSpan(new ForegroundColorSpan(Color.RED), 0, title.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
getSupportActionBar().setTitle(s);
getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.argb(128, 0, 0, 0)));
```

Agregando un FloatingActionButton (FAB)

En el diseño del material, un botón de acción flotante representa la acción principal en una actividad.
Se distinguen por un ícono en forma de círculo que flota sobre la interfaz de usuario y tienen comportamientos de movimiento que incluyen transformación, lanzamiento y un punto de anclaje de transferencia.

Asegúrese de que la siguiente dependencia se agregue al archivo build.gradle de su aplicación en las dependencias:

```
compile 'com.android.support:design:25.3.1'
```

Ahora agregue el `FloatingActionButton` a su archivo de diseño:

```
<android.support.design.widget.FloatingActionButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_margin="16dp"
      android:src="@drawable/some_icon"/>
```

donde el atributo `src` referencia al icono que se debe utilizar para la acción flotante.
El resultado debería ser similar a este (asumiendo que su color de acento es Material Pink):



De forma predeterminada, el color de fondo de su `FloatingActionButton` se establecerá en el color de acento de su tema. Además, tenga en cuenta que un `FloatingActionButton` requiere un margen alrededor de él para que funcione correctamente. El margen recomendado para la parte inferior es `16dp` para teléfonos y `24dp` para tabletas.

Aquí hay propiedades que puede usar para personalizar aún más el `FloatingActionButton` (asumiendo que `xmlns:app="http://schemas.android.com/apk/res-auto` se declara como espacio de nombres en la parte superior de su diseño):

- `app:fabSize` : se puede configurar en `normal` o `mini` para cambiar entre una versión de tamaño normal o una versión más pequeña.
- `app:rippleColor` : establece el color del efecto de onda de su `FloatingActionButton` . Puede ser un recurso de color o una cadena hexadecimal.
- `app:elevation` : puede ser una cadena, entero, booleano, valor de color, punto flotante, valor de dimensión.
- `app:useCompatPadding` : habilita el relleno de compatibilidad. Tal vez un valor booleano, como `true` o `false` . Establézcalo en `true` para usar el relleno de compatibilidad en api-21 y versiones posteriores, a fin de mantener un aspecto coherente con los niveles de API más antiguos.

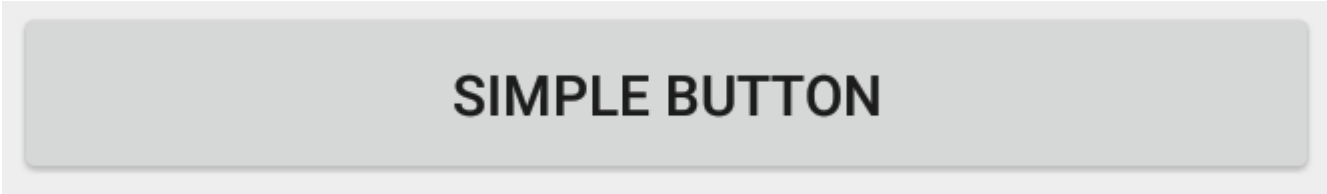
Puedes encontrar más ejemplos sobre FAB aquí .

Botones de estilo con Material Design.

La biblioteca de soporte de AppCompat define varios estilos útiles para los botones , cada uno de los cuales extiende un estilo `Widget.AppCompat.Button` base que se aplica a todos los botones de forma predeterminada si está utilizando un tema `AppCompat` . Este estilo ayuda a garantizar que todos los botones tengan el mismo aspecto por defecto siguiendo la especificación de Diseño de materiales .

En este caso el color de acento es rosa.

1. Botón simple: `@style/Widget.AppCompat.Button`




```
<Button
    style="@style/Widget.AppCompat.Button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/simple_button"/>
```

2. Botón de color: `@style/Widget.AppCompat.Button.Colored`
   El estilo `Widget.AppCompat.Button.Colored` extiende el estilo `Widget.AppCompat.Button` y aplica automáticamente el **color de acento** que seleccionó en el tema de su aplicación.




```
<Button
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/colored_button"/>
```

Si desea personalizar el color de fondo sin cambiar el color de acento en su *tema principal* , puede crear un *tema personalizado* (extendiendo el tema `ThemeOverlay`) para su `Button` y asignarlo al atributo `android:theme` del botón:

```
<Button
     style="@style/Widget.AppCompat.Button.Colored"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_margin="16dp"
     android:theme="@style/MyButtonTheme"/>
```

Defina el tema en `res/values/themes.xml` :

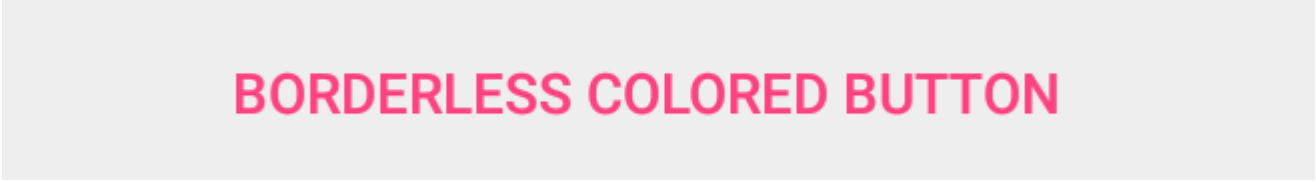
```
 <style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
      <item name="colorAccent">@color/my_color</item>
 </style>
```

3. Botón sin bordes: `@style/Widget.AppCompat.Button.Borderless`




```
<Button
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/borderless_button"/>
```

4. Botón de color sin bordes: `@style/Widget.AppCompat.Button.Borderless.Colored`




```
<Button
    style="@style/Widget.AppCompat.Button.Borderless.Colored"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/borderless_colored_button"/>
```

Cómo utilizar TextInputLayout

Asegúrese de que la siguiente dependencia se agregue al archivo `build.gradle` su aplicación en las dependencias:

```
compile 'com.android.support:design:25.3.1'
```

Muestra la sugerencia de un EditText como una etiqueta flotante cuando se ingresa un valor.

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

     <android.support.design.widget.TextInputEditText
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:hint="@string/form_username"/>

</android.support.design.widget.TextInputLayout>
```

Para mostrar el ícono del ojo de visualización de contraseña con TextInputLayout, podemos hacer uso del siguiente código:

```
<android.support.design.widget.TextInputLayout
    android:id="@+id/input_layout_current_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:passwordToggleEnabled="true">

    <android.support.design.widget.TextInputEditText

        android:id="@+id/current_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/current_password"
        android:inputType="textPassword" />

</android.support.design.widget.TextInputLayout>
```

donde se requieren los parámetros `app:passwordToggleEnabled="true"` & `android:inputType="textPassword"` .

`app` debe usar el espacio de nombres `xmlns:app="http://schemas.android.com/apk/res-auto"`

Puede encontrar más detalles y ejemplos en el tema dedicado.

Añadiendo un TabLayout

TabLayout proporciona un diseño horizontal para mostrar pestañas, y se usa comúnmente junto con un ViewPager .

Asegúrese de que la siguiente dependencia se agregue al archivo `build.gradle` su aplicación en las dependencias:

```
compile 'com.android.support:design:25.3.1'
```

Ahora puede agregar elementos a un TabLayout en su diseño usando la clase TabItem .

Por ejemplo:

```
<android.support.design.widget.TabLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:id="@+id/tabLayout">

    <android.support.design.widget.TabItem
        android:text="@string/tab_text_1"
        android:icon="@drawable/ic_tab_1"/>

    <android.support.design.widget.TabItem
        android:text="@string/tab_text_2"
        android:icon="@drawable/ic_tab_2"/>

</android.support.design.widget.TabLayout>
```

Agregue un `OnTabSelectedListener` para recibir una notificación cuando una pestaña en el `TabLayout` esté seleccionada / no seleccionada / reseleccionada:

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.tabLayout);
tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
    @Override
    public void onTabSelected(TabLayout.Tab tab) {
        int position = tab.getPosition();
        // Switch to view for this tab
    }

    @Override
    public void onTabUnselected(TabLayout.Tab tab) {

    }

    @Override
    public void onTabReselected(TabLayout.Tab tab) {

    }
});
```

Las pestañas también se pueden agregar / eliminar del `TabLayout` programación.

```
TabLayout.Tab tab = tabLayout.newTab();
tab.setText(R.string.tab_text_1);
tab.setIcon(R.drawable.ic_tab_1);
tabLayout.addTab(tab);

tabLayout.removeTab(tab);
tabLayout.removeTabAt(0);
tabLayout.removeAllTabs();
```

`TabLayout` tiene dos modos, fijo y desplazable.

```
tabLayout.setTabMode(TabLayout.MODE_FIXED);
tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);
```

Estos también se pueden aplicar en XML:

```
<android.support.design.widget.TabLayout
    android:id="@+id/tabLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:tabMode="fixed|scrollable" />
```

Nota: los modos `TabLayout` son mutuamente exclusivos, lo que significa que solo uno puede estar activo a la vez.

El color del indicador de tabulación es el color de acento definido para su tema de Diseño de materiales.
Puede anular este color definiendo un estilo personalizado en `styles.xml` y luego aplicando el estilo a su TabLayout:

```
<style name="MyCustomTabLayoutStyle" parent="Widget.Design.TabLayout">
    <item name="tabIndicatorColor">@color/your_color</item>
</style>
```

Luego puede aplicar el estilo a la vista usando:

```
<android.support.design.widget.TabLayout
        android:id="@+id/tabs"
        style="@style/MyCustomTabLayoutStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
</android.support.design.widget.TabLayout>
```

RippleDrawable

El efecto táctil de Ripple se introdujo con el diseño del material en Android 5.0 (nivel de API 21) y la animación se implementa mediante la nueva clase RippleDrawable .

> Dibujable que muestra un efecto dominó en respuesta a los cambios de estado. La posición de anclaje de la ondulación para un estado dado puede especificarse llamando a `setHotspot(float x, float y)` con el identificador de atributo de estado correspondiente.

5.0

En general, el efecto de onda para **los botones normales funciona de manera predeterminada** en API 21 y superior, y para otras vistas que se pueden tocar, se puede lograr especificando:

```
android:background="?android:attr/selectableItemBackground">
```

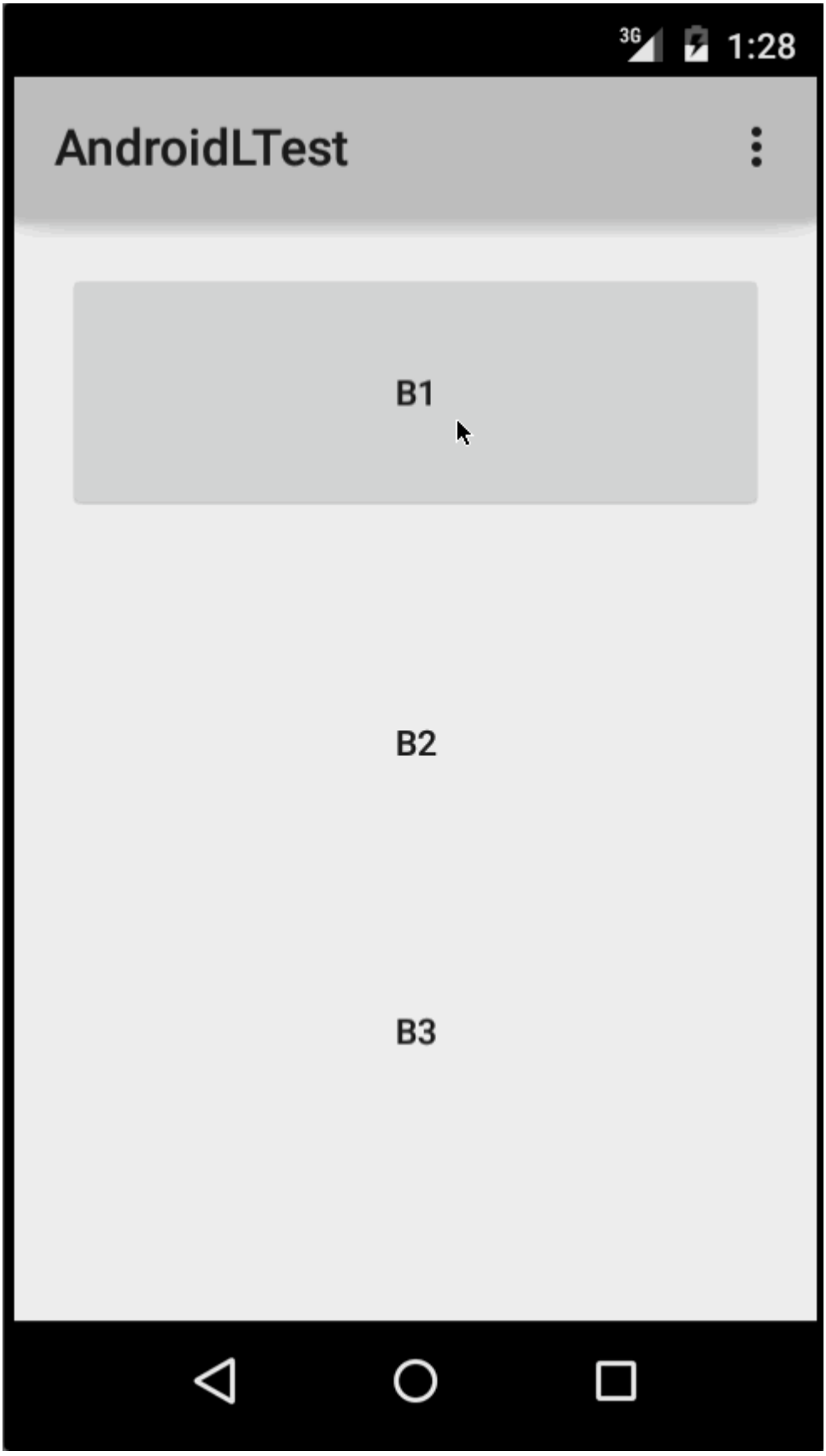
Para las ondulaciones contenidas dentro de la vista o:

```
android:background="?android:attr/selectableItemBackgroundBorderless"
```

para ondulaciones que se extienden más allá de los límites de la vista.

Por ejemplo, en la imagen de abajo,

- B1 es un botón que no tiene ningún fondo,
- B2 está configurado con `android:background="android:attr/selectableItemBackground"`
- B3 está configurado con `android:background="android:attr/selectableItemBackgroundBorderless"`



(Imagen cortesía: http://blog.csdn.net/a396901990/article/details/40187203)

Puedes lograr lo mismo en el código usando:

```
int[] attrs = new int[]{R.attr.selectableItemBackground};
TypedArray typedArray = getActivity().obtainStyledAttributes(attrs);
```

```
int backgroundResource = typedArray.getResourceId(0, 0);
myView.setBackgroundResource(backgroundResource);
```

Las ondulaciones también se pueden agregar a una vista usando el atributo `android:foreground` la misma manera que arriba. Como sugiere su nombre, en caso de que la onda se agregue al primer plano, se mostrará encima de cualquier vista a la que se agregue (por ejemplo, `ImageView` , un `LinearLayout` contenga varias vistas, etc.).

Si desea personalizar el efecto de rizado en una vista, debe crear un nuevo archivo `XML` , dentro del directorio dibujable.

Aquí hay algunos ejemplos:

Ejemplo 1 : Una ondulación sin límites

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#ffff0000" />
```

Ejemplo 2 : Ondulación con máscara y color de fondo.

```
<ripple android:color="#7777777"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/mask"
        android:drawable="#ffff00" />
    <item android:drawable="@android:color/white"/>
</ripple>
```

Si hay una `view` con un fondo *ya especificado* con una `shape` , `corners` y cualquier otra etiqueta, para agregar una ondulación a esa vista, use una `mask layer` y establezca la ondulación como el fondo de la vista.

Ejemplo :

```
 <?xml version="1.0" encoding="utf-8"?>
    <ripple xmlns:android="http://schemas.android.com/apk/res/android"
        android:color="?android:attr/colorControlHighlight">
        <item android:id="@android:id/mask">
           <shape
              android:shape="rectangle">
              solid android:color="#000000"/>
               <corners
                 android:radius="25dp"/>
            </shape>
        </item>
        <item android:drawable="@drawable/rounded_corners" />
    </ripple>
```

Ejemplo 3 : Ondulación sobre un recurso dibujable

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#ff0000ff">
    <item android:drawable="@drawable/my_drawable" />
</ripple>
```

Uso: Para adjuntar su archivo xpl ripple a cualquier vista, `my_ripple.xml` como fondo como sigue (asumiendo que su archivo ripple se llame `my_ripple.xml`):

```
<View
    android:id="@+id/myViewId"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/my_ripple" />
```

Selector:

El ripple drawable también se puede usar en lugar de los selectores de lista de estados de color si su versión de destino es v21 o superior (también puede colocar el selector de ripple en la carpeta `drawable-v21`):

```
<!-- /drawable/button.xml: -->
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true" android:drawable="@drawable/button_pressed"/>
    <item android:drawable="@drawable/button_normal"/>
</selector>

<!--/drawable-v21/button.xml:-->
<?xml version="1.0" encoding="utf-8"?>
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="?android:colorControlHighlight">
   <item android:drawable="@drawable/button_normal" />
</ripple>
```

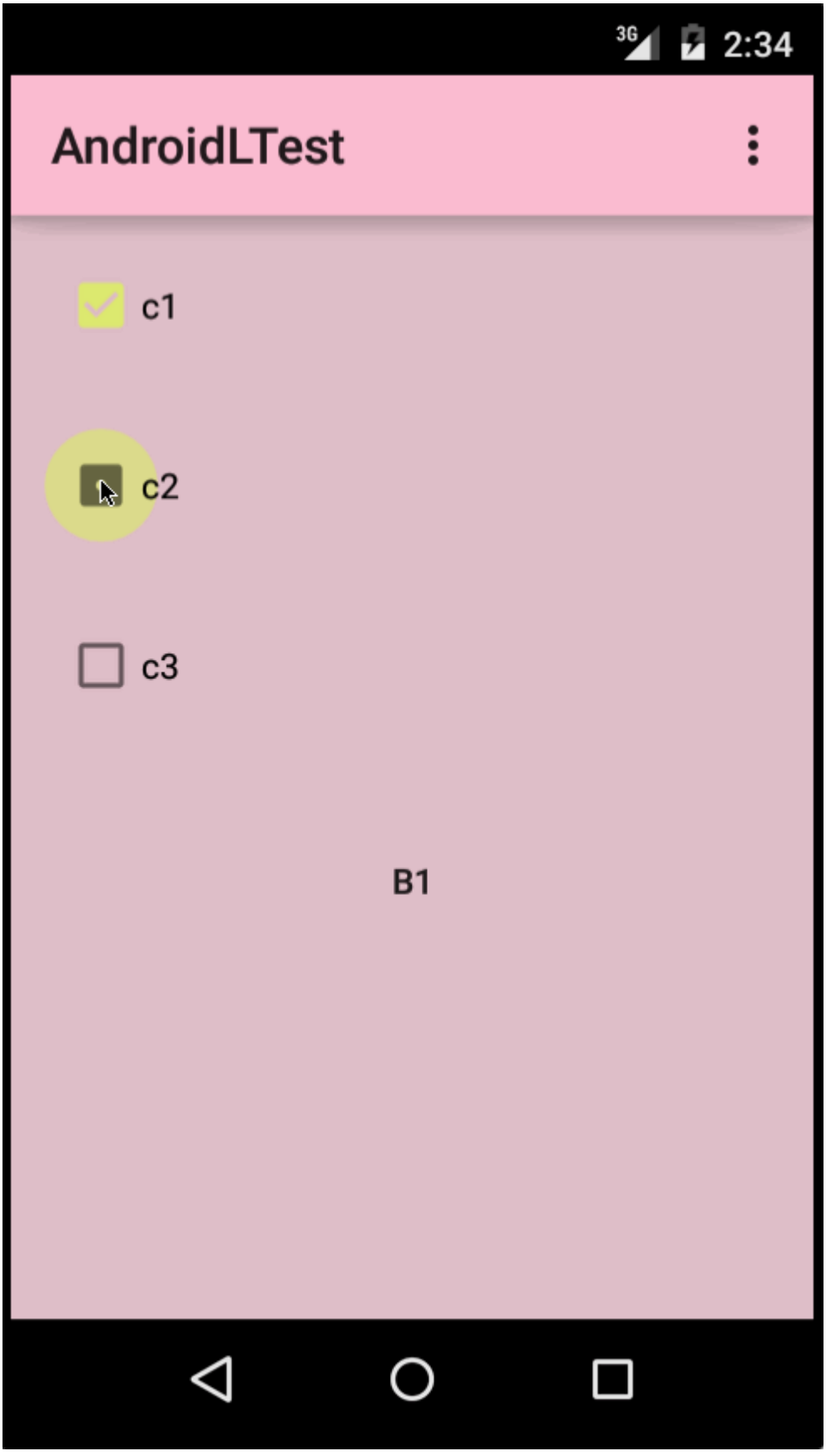
En este caso, el color del estado predeterminado de su vista sería blanco y el estado presionado mostraría la ondulación dibujable.

Punto a tener en cuenta: el uso de `?android:colorControlHighlight` le dará a la onda el mismo color que las ondas incorporadas en su aplicación.

Para cambiar solo el color de la onda, puede personalizar el color de `android:colorControlHighlight` en su tema así:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

  <style name="AppTheme" parent="android:Theme.Material.Light.DarkActionBar">
    <item name="android:colorControlHighlight">@color/your_custom_color</item>
  </style>

</resources>
```

y luego use este tema en sus actividades, etc. El efecto sería como la imagen a continuación:



(Imagen cortesía: http://blog.csdn.net/a396901990/article/details/40187203)

Añadir un cajón de navegación

Los cajones de navegación se utilizan para navegar a destinos de nivel superior en una aplicación.

Asegúrese de haber agregado la biblioteca de soporte de diseño en su archivo `build.gradle` bajo las dependencias:

```
 dependencies {
    // ...
    compile 'com.android.support:design:25.3.1'
}
```

A continuación, agregue `DrawerLayout` y `NavigationView` en su archivo de recursos de diseño XML. `DrawerLayout` es solo un elegante contenedor que permite que `NavigationView` , el cajón de navegación real, se deslice hacia afuera desde la izquierda o la derecha de la pantalla. Nota: para dispositivos móviles, el tamaño estándar del cajón es 320dp.

```
<!-- res/layout/activity_main.xml -->
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/navigation_drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">
    <! -- You can use "end" to open drawer from the right side -->

    <android.support.design.widget.CoordinatorLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true">

        <android.support.design.widget.AppBarLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:theme="@style/AppTheme.AppBarOverlay">

            <android.support.v7.widget.Toolbar
                android:id="@+id/toolbar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                android:background="?attr/colorPrimary"
                app:popupTheme="@style/AppTheme.PopupOverlay" />

        </android.support.design.widget.AppBarLayout>

    </android.support.design.widget.CoordinatorLayout>

    <android.support.design.widget.NavigationView
        android:id="@+id/navigation_drawer"
        android:layout_width="320dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/drawer_header"
        app:menu="@menu/navigation_menu" />

</android.support.v4.widget.DrawerLayout>
```

Ahora, si lo desea, cree un **archivo de encabezado** que servirá como la parte superior de su cajón de navegación. Esto se utiliza para dar un aspecto mucho más elegante al cajón.

```
<!-- res/layout/drawer_header.xml -->
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="190dp">

    <ImageView
        android:id="@+id/header_image"
        android:layout_width="140dp"
        android:layout_height="120dp"
        android:layout_centerInParent="true"
        android:scaleType="centerCrop"
        android:src="@drawable/image" />

    <TextView
        android:id="@+id/header_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/header_image"
        android:text="User name"
        android:textSize="20sp" />

</RelativeLayout>
```

Se hace referencia en la etiqueta `NavigationView` en la `app:headerLayout="@layout/drawer_header"` . Esta `app:headerLayout` infla el diseño especificado en el encabezado automáticamente. Esto también puede hacerse en tiempo de ejecución con:

```
// Lookup navigation view
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation_drawer);
// Inflate the header view at runtime
View headerLayout = navigationView.inflateHeaderView(R.layout.drawer_header);
```

Para llenar automáticamente el cajón de navegación con elementos de navegación compatibles con el diseño de materiales, cree un archivo de menú y agregue elementos según sea necesario. Nota: aunque no se requieren iconos para los elementos, se sugieren en la especificación de Diseño de materiales .
Se menciona en la etiqueta `NavigationView` en el `app:menu="@menu/navigation_menu" attribute` .

```
<!-- res/menu/menu_drawer.xml -->
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/nav_item_1"
        android:title="Item #1"
        android:icon="@drawable/ic_nav_1" />
    <item
        android:id="@+id/nav_item_2"
        android:title="Item #2"
        android:icon="@drawable/ic_nav_2" />
    <item
        android:id="@+id/nav_item_3"
        android:title="Item #3"
        android:icon="@drawable/ic_nav_3" />
```

```
    <item
        android:id="@+id/nav_item_4"
        android:title="Item #4"
        android:icon="@drawable/ic_nav_4" />
</menu>
```

Para separar los elementos en grupos, colóquelos en un `<menu>` anidado en otro `<item>` con un atributo `android:title` o envuélvalos con la etiqueta `<group>` .

Ahora que el diseño está listo, pase al código de `Activity` :

```
  // Find the navigation view
  NavigationView navigationView = (NavigationView) findViewById(R.id.navigation_drawer);
  navigationView.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
      @Override
      public boolean onNavigationItemSelected(MenuItem item) {
          // Get item ID to determine what to do on user click
          int itemId = item.getItemId();
          // Respond to Navigation Drawer selections with a new Intent
          startActivity(new Intent(this, OtherActivity.class));
          return true;
      }
  });

  DrawerLayout drawer = (DrawerLayout) findViewById(R.id.navigation_drawer_layout);
  // Necessary for automatically animated navigation drawer upon open and close
  ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawer, "Open navigation
drawer", "Close navigation drawer");
  // The two Strings are not displayed to the user, but be sure to put them into a separate
strings.xml file.
  drawer.addDrawerListener(toggle);
  toogle.syncState();
```

Ahora puede hacer lo que quiera en la vista de encabezado de `NavigationView`

```
View headerView = navigationView.getHeaderView();
TextView headerTextView = (TextView) headerview.findViewById(R.id.header_text_view);
ImageView headerImageView = (ImageView) headerview.findViewById(R.id.header_image);
// Set navigation header text
headerTextView.setText("User name");
// Set navigation header image
headerImageView.setImageResource(R.drawable.header_image);
```

La vista de encabezado se comporta como cualquier otra `View` , por lo que una vez que use `findViewById()` y agregue algunas otras `View` a su archivo de diseño, puede establecer las propiedades de cualquier elemento en él.

Puede encontrar más detalles y ejemplos en el tema dedicado .

Hojas inferiores en la biblioteca de soporte de diseño

Las hojas inferiores se deslizan hacia arriba desde la parte inferior de la pantalla para revelar más contenido.

Se agregaron a la biblioteca de soporte de Android en la versión v25.1.0 y son compatibles con todas las versiones.

Asegúrese de que la siguiente dependencia se agregue al archivo build.gradle de su aplicación en las dependencias:

```
compile 'com.android.support:design:25.3.1'
```

Hojas inferiores persistentes

Puede lograr una Hoja de abajo persistente adjuntando una `BottomSheetBehavior` de `BottomSheetBehavior` a una vista de niño de un `CoordinatorLayout` :

```
<android.support.design.widget.CoordinatorLayout >

    <!-- .....   -->

    <LinearLayout
       android:id="@+id/bottom_sheet"
       android:elevation="4dp"
       android:minHeight="120dp"
       app:behavior_peekHeight="120dp"
       ...
       app:layout_behavior="android.support.design.widget.BottomSheetBehavior">

           <!-- .....   -->

       </LinearLayout>

</android.support.design.widget.CoordinatorLayout>
```

Luego, en tu código puedes crear una referencia usando:

```
// The View with the BottomSheetBehavior
View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

Puede establecer el estado de su BottomSheetBehavior utilizando el método setState () :

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

Puedes usar uno de estos estados:

- `STATE_COLLAPSED` : este estado colapsado es el predeterminado y muestra solo una parte del diseño en la parte inferior. La altura se puede controlar con el atributo `app:behavior_peekHeight` (predeterminado en 0)
- `STATE_EXPANDED` : el estado totalmente expandido de la hoja inferior, donde puede verse toda la hoja inferior (si su altura es menor que la que contiene el `CoordinatorLayout`) o la totalidad del `CoordinatorLayout` se llena

- `STATE_HIDDEN` : deshabilitado de forma predeterminada (y habilitado con la `app:behavior_hideable` atributo de `app:behavior_hideable` ocultable), lo que permite a los usuarios deslizarse hacia abajo en la hoja inferior para ocultar completamente la hoja inferior

Además de abrir o cerrar la Hoja de Fondo al hacer clic en una Vista de su elección, digamos A Button, aquí le indicamos cómo cambiar el comportamiento de la hoja y la vista de actualización.

```
mButton = (Button) findViewById(R.id.button_2);
    //On Button click we monitor the state of the sheet
    mButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (mBottomSheetBehavior.getState() == BottomSheetBehavior.STATE_EXPANDED) {
                //If expanded then collapse it (setting in Peek mode).
                mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_COLLAPSED);
                mButton.setText(R.string.button2_hide);
            } else if (mBottomSheetBehavior.getState() == BottomSheetBehavior.STATE_COLLAPSED)
{
                //If Collapsed then hide it completely.
                mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_HIDDEN);
                mButton.setText(R.string.button2);
            } else if (mBottomSheetBehavior.getState() == BottomSheetBehavior.STATE_HIDDEN) {
                //If hidden then Collapse or Expand, as the need be.
                mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
                mButton.setText(R.string.button2_peek);
            }
        }
    });
```

Pero el comportamiento de `BottomSheet` también tiene una característica en la que el usuario puede interactuar con el movimiento hacia arriba o hacia abajo con un movimiento DRAG. En tal caso, es posible que no podamos actualizar la Vista dependiente (como el botón de arriba) si el estado de la Hoja ha cambiado. En ese caso, le gustaría recibir devoluciones de llamadas de cambios de estado, por lo tanto, puede agregar `BottomSheetCallback` para escuchar los eventos de deslizamiento de usuarios:

```
mBottomSheetBehavior.setBottomSheetCallback(new BottomSheetCallback() {
    @Override
    public void onStateChanged(@NonNull View bottomSheet, int newState) {
      // React to state change and notify views of the current state
    }
      @Override
      public void onSlide(@NonNull View bottomSheet, float slideOffset) {
       // React to dragging events and animate views or transparency of dependent views
   }
 });
```

Y si solo quiere que su Hoja inferior sea visible solo en el modo COLLAPSED y EXPANDED, y nunca OCULTE el uso:

```
mBottomSheetBehavior2.setHideable(false);
```

Hoja inferior DialogFragment

También puede mostrar un BottomSheetDialogFragment en lugar de una vista en la hoja inferior. Para hacer esto, primero necesita crear una nueva clase que amplíe BottomSheetDialogFragment.

Dentro del método `setupDialog()` , puede inflar un nuevo archivo de diseño y recuperar el BottomSheetBehavior de la vista del contenedor en su Actividad. Una vez que tenga el comportamiento, puede crear y asociar BottomSheetCallback con él para descartar el Fragmento cuando la hoja está oculta.

```
public class BottomSheetDialogFragmentExample extends BottomSheetDialogFragment {

    private BottomSheetBehavior.BottomSheetCallback mBottomSheetBehaviorCallback = new
BottomSheetBehavior.BottomSheetCallback() {

        @Override
        public void onStateChanged(@NonNull View bottomSheet, int newState) {
            if (newState == BottomSheetBehavior.STATE_HIDDEN) {
                dismiss();
            }

        }

        @Override
        public void onSlide(@NonNull View bottomSheet, float slideOffset) {
        }
    };

    @Override
    public void setupDialog(Dialog dialog, int style) {
        super.setupDialog(dialog, style);
        View contentView = View.inflate(getContext(), R.layout.fragment_bottom_sheet, null);
        dialog.setContentView(contentView);

        CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams) ((View)
contentView.getParent()).getLayoutParams();
        CoordinatorLayout.Behavior behavior = params.getBehavior();

        if( behavior != null && behavior instanceof BottomSheetBehavior ) {
            ((BottomSheetBehavior)
behavior).setBottomSheetCallback(mBottomSheetBehaviorCallback);
        }
    }
}
```

Finalmente, puede llamar a show () en una instancia de su Fragmento para mostrarlo en la hoja inferior.

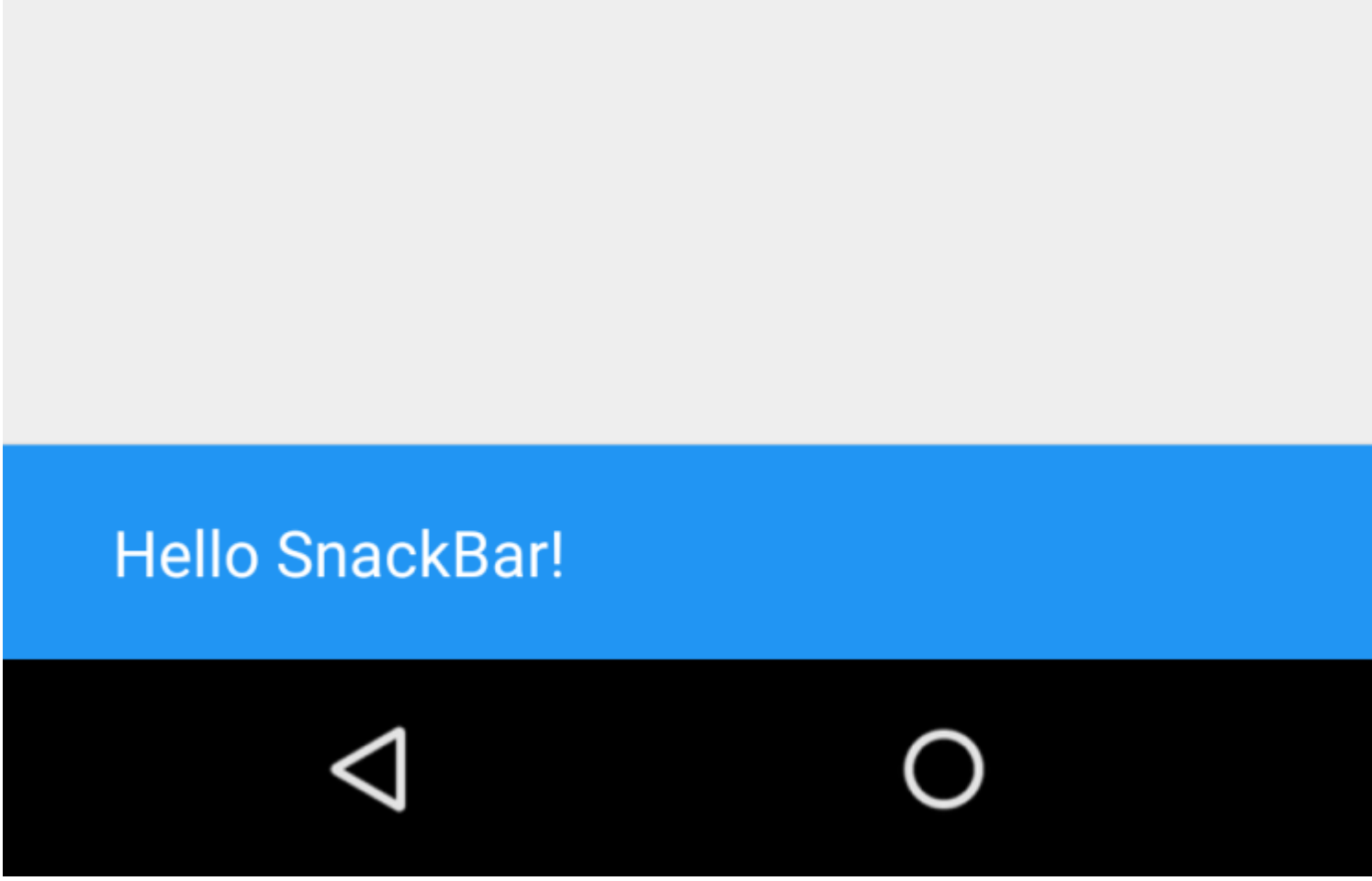
```
BottomSheetDialogFragment bottomSheetDialogFragment = new BottomSheetDialogFragmentExample();
bottomSheetDialogFragment.show(getSupportFragmentManager(),
bottomSheetDialogFragment.getTag());
```

Puedes encontrar más detalles en el tema dedicado.

Añadir un Snackbar

Una de las características principales en Material Design es la adición de un `Snackbar` , que en teoría reemplaza al `Toast` anterior. Según la documentación de Android:

> Snackbars contienen una sola línea de texto directamente relacionada con la operación realizada. Pueden contener una acción de texto, pero no iconos. Los brindis se utilizan principalmente para la mensajería del sistema. También se muestran en la parte inferior de la pantalla, pero no se pueden deslizar fuera de la pantalla.



Las tostadas aún se pueden usar en Android para mostrar mensajes a los usuarios, sin embargo, si ha decidido optar por el uso del diseño de material en su aplicación, se recomienda que use una barra de refrigerios. En lugar de mostrarse como una superposición en su pantalla, aparece un `Snackbar` desde la parte inferior.

Así es como se hace:

```
Snackbar snackbar = Snackbar
        .make(coordinatorLayout, "Here is your new Snackbar", Snackbar.LENGTH_LONG);
snackbar.show();
```

En cuanto a la cantidad de tiempo para mostrar el `Snackbar` , tenemos las opciones similares a las ofrecidas por un `Toast` o podríamos establecer una duración personalizada en milisegundos:

- `LENGTH_SHORT`

- `LENGTH_LONG`
- `LENGTH_INDEFINITE`
- `setDuration()` (desde la versión `22.2.1`)

También puede agregar funciones dinámicas a su `Snackbar` , como `ActionCallback` o color personalizado. Sin embargo, preste atención a la guía de diseño ofrecida por Android al personalizar un `Snackbar` .

Implementar el `Snackbar` tiene una limitación sin embargo. El diseño principal de la vista en la que va a implementar un `Snackbar` debe ser un `CoordinatorLayout` . Esto es para que se pueda hacer la ventana emergente real desde la parte inferior.

Así es como se define un `CoordinatorLayout` en su archivo xml de diseño:

```
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/coordinatorLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    //any other widgets in your layout go here.

</android.support.design.widget.CoordinatorLayout>
```

El `CoordinatorLayout` luego debe definirse en el método `onCreate` su Actividad, y luego usarse cuando se crea el `Snackbar` .

Para obtener más información sobre `Snackbar` , consulte la documentación oficial o el tema dedicado en la documentación.

Lea Diseño de materiales en línea: https://riptutorial.com/es/android/topic/124/diseno-de-materiales

Capítulo 92: Diseños

Introducción

Un diseño define la estructura visual de una interfaz de usuario, como una actividad o un widget.

Se declara un diseño en XML, incluidos los elementos de pantalla que aparecerán en él. Se puede agregar código a la aplicación para modificar el estado de los objetos de pantalla en tiempo de ejecución, incluidos los declarados en XML.

Sintaxis

- android: gravity = "arriba | abajo | izquierda | derecha | center_vertical | fill_vertical | center_horizontal | fill_horizontal | center | fill | clip_vertical | clip_horizontal | start | end"
- android: layout_gravity = "arriba | abajo | izquierda | derecha | center_vertical | fill_vertical | center_horizontal | fill_horizontal | center | fill | clip_vertical | clip_horizontal | start | end"

Observaciones

LayoutParams y Layout_ Attributes

Layout Attributes

+CoordinatorLayout

layout_behavior

+FrameLayout

layout_gravity

+LinearLayout

layout_weight

+RelativeLayout

layout_above layout_below

layout_alignLeft/Top/Right/Bottom

layout_alignParentLeft/etc

layout_toLeftOf/etc

layout_alignBaseline

layout_centerInParent

+AbsoluteLayout NO

Please don't

Linear



orientation="horiz

vs

orientation="vert



requiere dos pases de diseño para representarse correctamente. Para jerarquías de vista complejas, esto puede tener un impacto significativo en el rendimiento. Anidar `RelativeLayouts` hace que este problema sea aún peor, porque cada `RelativeLayout` hace que `RelativeLayout` el número de pases de diseño.

Examples

LinearLayout

El LinearLayout es un `ViewGroup` que organiza sus hijos en una sola columna o una sola fila. La orientación se puede establecer llamando al método `setOrientation()` o usando el atributo xml `android:orientation` .

1. **Orientación vertical** : `android:orientation="vertical"`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/app_name" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@android:string/cancel" />

</LinearLayout>
```

Aquí hay una captura de pantalla de cómo se verá esto:



2. **Orientación horizontal** : `android:orientation="horizontal"`

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/app_name" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@android:string/cancel" />
```

`LinearLayout` también admite la asignación de un peso a niños individuales con el atributo `android:layout_weight` .

Disposición relativa

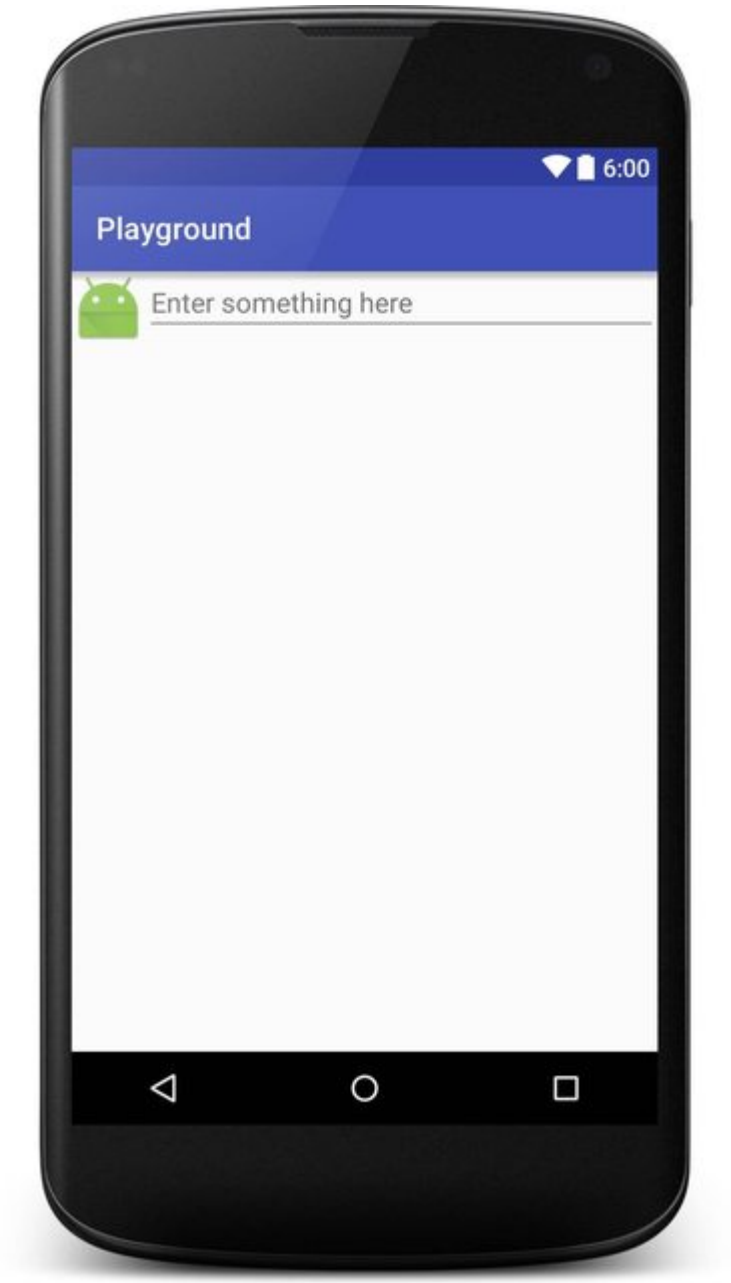
`RelativeLayout` es un `ViewGroup` que muestra vistas secundarias en posiciones relativas. De forma

predeterminada, todas las vistas secundarias se dibujan en la parte superior izquierda del diseño, por lo que debe definir la posición de cada vista utilizando las distintas propiedades de diseño disponibles en `RelativeLayout.LayoutParams` . El valor de cada propiedad de diseño es un valor booleano para habilitar una posición de diseño relativa al RelativeLayout principal o una ID que haga referencia a otra vista en el diseño en la que se debe colocar la vista.

Ejemplo:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@mipmap/ic_launcher" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:layout_toRightOf="@+id/imageView"
        android:layout_toEndOf="@+id/imageView"
        android:hint="@string/hint" />

</RelativeLayout>
```

Aquí hay una captura de pantalla de cómo se verá esto:



Gravedad y diseño de gravedad.

Android: layout_gravity

- `android:layout_gravity` se utiliza para establecer la posición de un elemento en su elemento principal (por ejemplo, una `View` secundaria dentro de un `Layout`).
- Compatible con LinearLayout y FrameLayout

android: gravedad

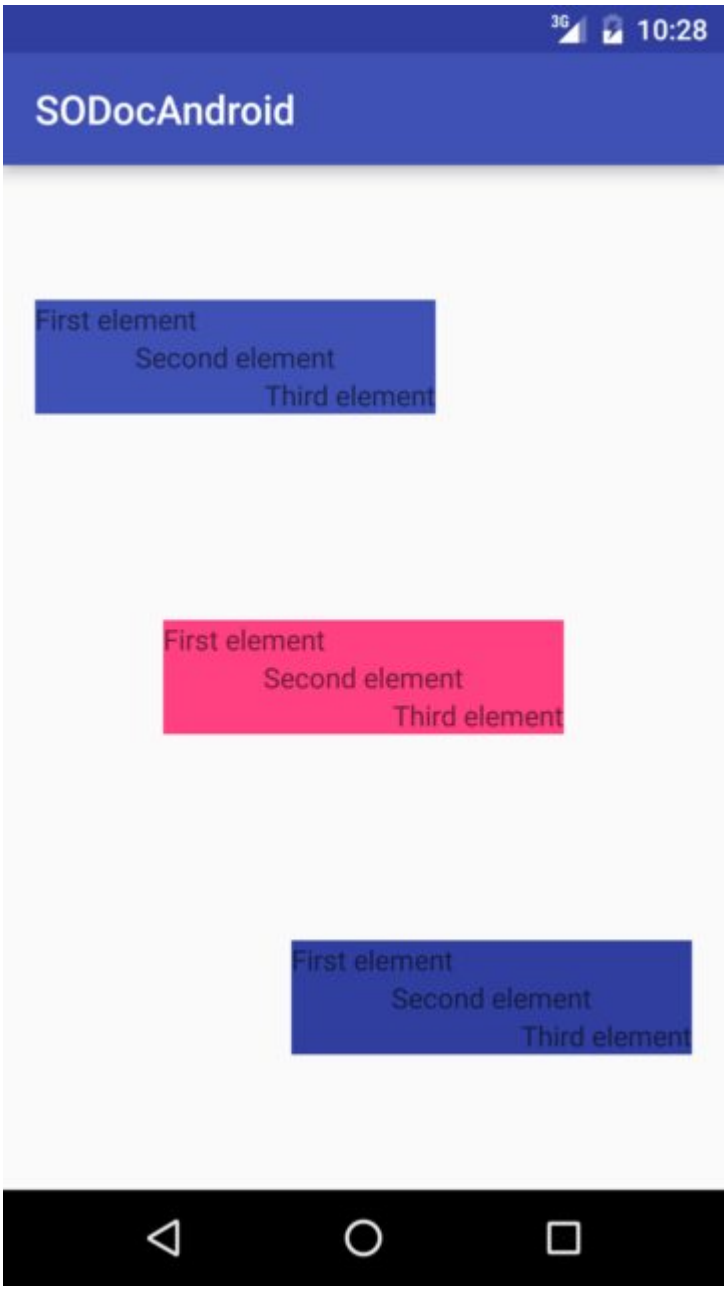
- `android:gravity` se utiliza para establecer la posición del contenido dentro de un elemento (por ejemplo, un texto dentro de un `TextView`).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        android:layout_gravity="left"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/first"
            android:background="@color/colorPrimary"
            android:gravity="left"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/second"
            android:background="@color/colorPrimary"
            android:gravity="center"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/third"
            android:background="@color/colorPrimary"
            android:gravity="right"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        android:layout_gravity="center"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/first"
            android:background="@color/colorAccent"
            android:gravity="left"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/second"
            android:background="@color/colorAccent"
            android:gravity="center"/>

        <TextView
```

```
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/third"
            android:background="@color/colorAccent"
            android:gravity="right"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        android:layout_gravity="right"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/first"
            android:background="@color/colorPrimaryDark"
            android:gravity="left"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/second"
            android:background="@color/colorPrimaryDark"
            android:gravity="center"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/third"
            android:background="@color/colorPrimaryDark"
            android:gravity="right"/>

    </LinearLayout>

</LinearLayout>
```

Que se renderiza de la siguiente manera:

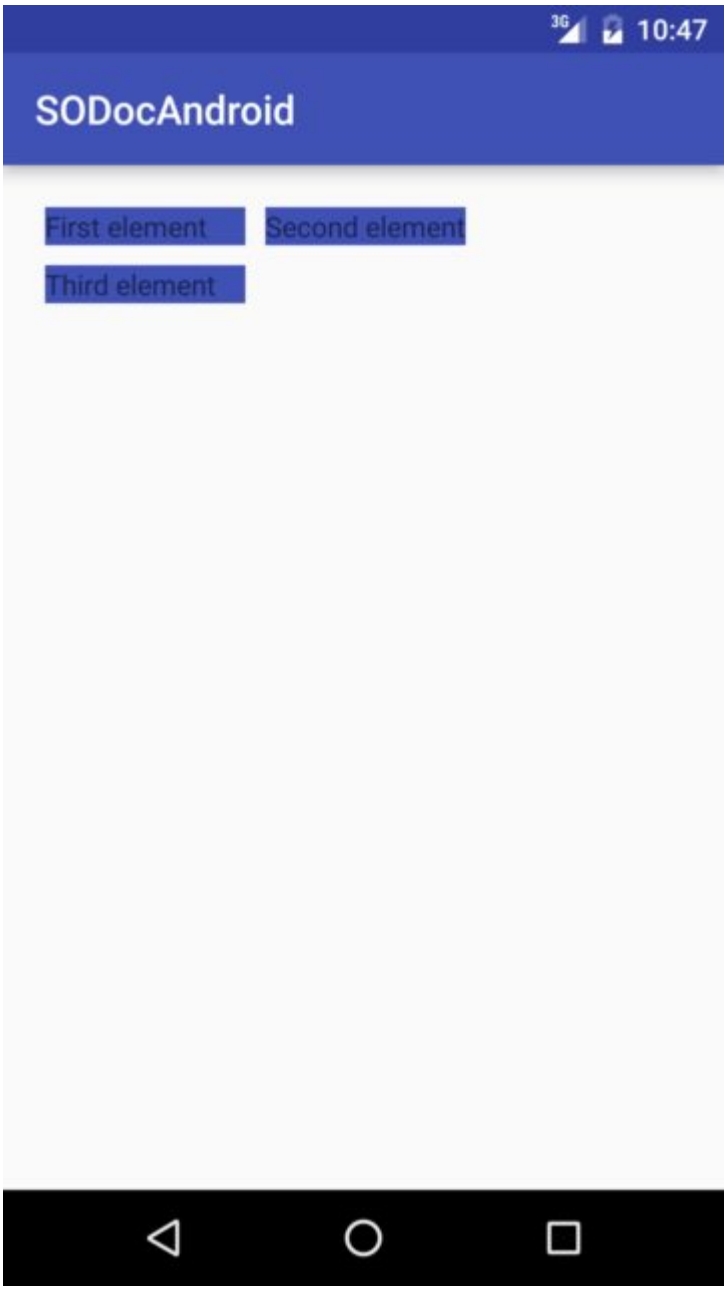


Diseño de cuadrícula

GridLayout, como su nombre indica, es un diseño utilizado para organizar las vistas en una cuadrícula. Un GridLayout se divide en columnas y filas. Como se puede ver en el siguiente ejemplo, la cantidad de columnas y / o filas se especifica por las propiedades `columnCount` y `rowCount` . Agregar vistas a este diseño agregará la primera vista a la primera columna, la segunda vista a la segunda columna y la tercera vista a la primera columna de la segunda fila.

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
```

```
    android:columnCount="2"
    android:rowCount="2">

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/first"
        android:background="@color/colorPrimary"
        android:layout_margin="@dimen/default_margin" />

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/second"
        android:background="@color/colorPrimary"
        android:layout_margin="@dimen/default_margin" />

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/third"
        android:background="@color/colorPrimary"
        android:layout_margin="@dimen/default_margin" />

</GridLayout>
```



Porcentaje de diseños

2.3

Percent Support Library proporciona `PercentFrameLayout` y `PercentRelativeLayout` , dos ViewGroups que proporcionan una manera fácil de especificar las **dimensiones y márgenes de la** Vista en términos de un **porcentaje** del tamaño general.

Puede usar la biblioteca de soporte de porcentaje agregando lo siguiente a sus dependencias.

```
compile 'com.android.support:percent:25.3.1'
```

Si quisiera mostrar una vista que llene la pantalla horizontalmente pero solo la mitad de la pantalla verticalmente, haría lo siguiente.

```
<android.support.percent.PercentFrameLayout
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        app:layout_widthPercent="100%"
        app:layout_heightPercent="50%"
        android:background="@android:color/black" />

<android.support.percent.PercentFrameLayout>
```

También puede definir los porcentajes en un archivo XML separado con código como:

```
<fraction name="margin_start_percent">25%</fraction>
```

Y consúltelos en sus diseños con `@fraction/margin_start_percent` .

También contienen la capacidad de establecer una **relación de aspecto** personalizada a través de la `app:layout_aspectRatio` .
Esto le permite establecer solo una dimensión, como solo el ancho, y la altura se determinará automáticamente en función de la relación de aspecto que haya definido, ya sea 4: 3 o 16: 9 o incluso un cuadrado 1: 1 relación de aspecto.

Por ejemplo:

```
 <ImageView
    app:layout_widthPercent="100%"
    app:layout_aspectRatio="178%"
    android:scaleType="centerCrop"
    android:src="@drawable/header_background"/>
```

FrameLayout

`FrameLayout` está diseñado para bloquear un área en la pantalla para mostrar un solo elemento. Sin embargo, puede agregar varios hijos a un FrameLayout y controlar su posición dentro del FrameLayout asignando la gravedad a cada niño, usando el atributo android: layout_gravity .

Generalmente, `FrameLayout` se usa para mantener una sola vista secundaria. Los casos de uso comunes son la creación de marcadores de posición para inflar `Fragments` en `Activity` , superponer vistas o aplicar primer plano a las vistas.

Ejemplo:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent">

    <ImageView
        android:src="@drawable/nougat"
        android:scaleType="fitCenter"
        android:layout_height="match_parent"
```

```
        android:layout_width="match_parent"/>

    <TextView
        android:text="FrameLayout Example"
        android:textSize="30sp"
        android:textStyle="bold"
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:gravity="center"/>

</FrameLayout>
```

Se verá así:



CoordinatorLayout

2.3

El `CoordinatorLayout` es un contenedor similar al `FrameLayout` pero con capacidades adicionales, se denomina `FrameLayout` en la documentación oficial.

Al adjuntar un comportamiento de `CoordinatorLayout.Behavior` a un hijo directo de CoordinatorLayout, podrá interceptar eventos táctiles, inserciones de ventanas, medidas, diseño y desplazamiento anidado.

Para usarlo, primero deberá agregar una dependencia para la biblioteca de soporte en su archivo de gradle:

```
compile 'com.android.support:design:25.3.1'
```

El número de la última versión de la biblioteca se puede encontrar aquí.

Un caso de uso práctico de `CoordinatorLayout` es crear una vista con un `FloatingActionButton` . En este caso específico, crearemos un `RecyclerView` con un `SwipeRefreshLayout` y un `FloatingActionButton` además de eso. Así es como puedes hacer eso:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/coord_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <android.support.v4.widget.SwipeRefreshLayout
        android:id="@+id/swipe_refresh_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <android.support.v7.widget.RecyclerView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/recycler_view"/>

    </android.support.v4.widget.SwipeRefreshLayout>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:clickable="true"
        android:color="@color/colorAccent"
        android:src="@mipmap/ic_add_white"
        android:layout_gravity="end|bottom"
        app:layout_anchorGravity="bottom|right|end"/>

</android.support.design.widget.CoordinatorLayout>
```

Observe cómo el FloatingActionButton está anclado al CoordinatorLayout con la `app:layout_anchor="@id/coord_layout"`

CoordinatorLayout Scrolling Behavior

2.3-2.3.2

Se puede usar un `CoordinatorLayout` adjunto para lograr efectos de desplazamiento de diseño de materiales cuando se usan diseños internos que admiten el desplazamiento anidado, como `NestedScrollView` o `RecyclerView` .

Para este ejemplo:

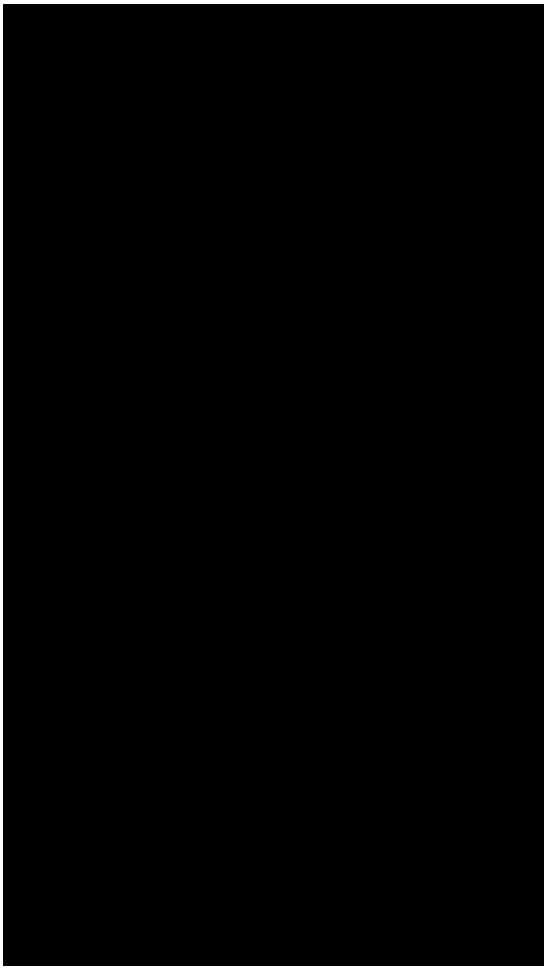
- `app:layout_scrollFlags="scroll|enterAlways"` se usa en las propiedades de la barra de herramientas
- `app:layout_behavior="@string/appbar_scrolling_view_behavior"` se usa en las propiedades de ViewPager
- Se utiliza un RecyclerView en los fragmentos de ViewPager

Aquí está el archivo XML de diseño utilizado en una actividad:

```
<android.support.design.widget.CoordinatorLayout
    android:id="@+id/main_layout"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <android.support.design.widget.AppBarLayout
        android:id="@+id/appBarLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:elevation="6dp">
    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:background="?attr/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
        app:elevation="0dp"
        app:layout_scrollFlags="scroll|enterAlways"
        />

    <android.support.design.widget.TabLayout
        android:id="@+id/tab_layout"
        app:tabMode="fixed"
        android:layout_below="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimary"
        app:elevation="0dp"
        app:tabTextColor="#d3d3d3"
        android:minHeight="?attr/actionBarSize"
        />

    </android.support.design.widget.AppBarLayout>

    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager"
```

```
        android:layout_below="@+id/tab_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        />

</android.support.design.widget.CoordinatorLayout>
```

Resultado:



Ver peso

Uno de los atributos más utilizados para LinearLayout es el peso de sus vistas secundarias. El peso define cuánto espacio consumirá una vista en comparación con otras vistas dentro de un LinearLayout.

El peso se usa cuando se quiere dar espacio de pantalla específico a un componente en comparación con otro.

Propiedades clave :

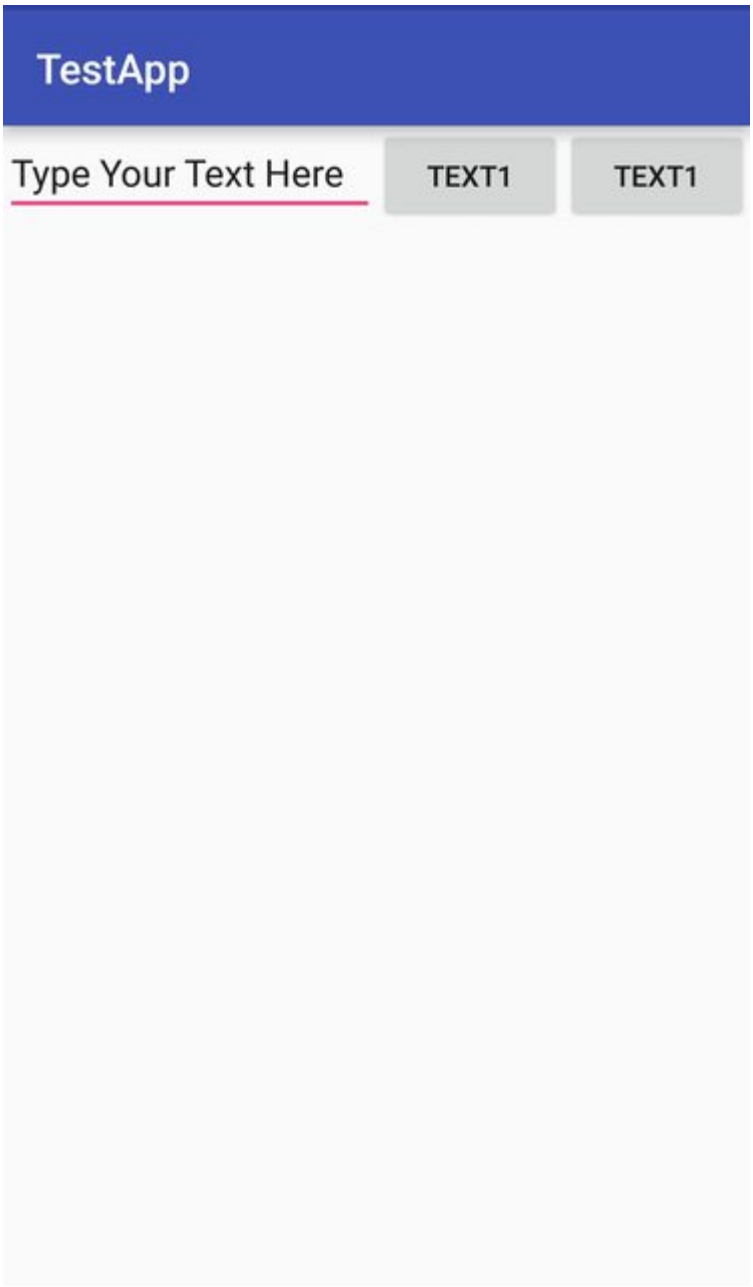
- `weightSum` es la suma total de pesos de todas las vistas de niños. Si no especifica el `weightSum` , el sistema calculará la suma de todos los pesos por su cuenta.
- `layout_weight` especifica la cantidad de espacio fuera de la suma de peso total que ocupará

el widget.

Código:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:weightSum="4">

    <EditText
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Type Your Text Here" />

    <Button
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Text1" />

    <Button
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Text1" />

</LinearLayout>
```

La salida es:



Ahora, incluso si el tamaño del dispositivo es mayor, el EditText ocupará 2/4 del espacio de la pantalla. Por lo tanto, el aspecto de su aplicación se ve consistente en todas las pantallas.

Nota: aquí el `layout_width` se mantiene `0dp` ya que el espacio del widget se divide horizontalmente. Si los widgets se alinean verticalmente, `layout_height` se establecerá en `0dp` . Esto se hace para aumentar la eficiencia del código porque en el tiempo de ejecución, el sistema no intentará calcular el ancho o la altura respectivamente, ya que esto se maneja con el peso. Si en su lugar usó `wrap_content` el sistema intentaría calcular el ancho / alto primero antes de aplicar el atributo de peso que causa otro ciclo de cálculo.

Creando LinearLayout programáticamente

Jerarquía

```
- LinearLayout(horizontal)
   - ImageView
```

```
  - LinearLayout(vertical)
    - TextView
    - TextView
```

Código

```
LinearLayout rootView = new LinearLayout(context);
rootView.setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.WRAP_CONTENT));
rootView.setOrientation(LinearLayout.HORIZONTAL);

// for imageview
ImageView imageView = new ImageView(context);
// for horizontal linearlayout
LinearLayout linearLayout2 = new LinearLayout(context);
linearLayout2.setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.WRAP_CONTENT));
linearLayout2.setOrientation(LinearLayout.VERTICAL);

TextView tv1 = new TextView(context);
TextView tv2 = new TextView(context);
// add 2 textview to horizontal linearlayout
linearLayout2.addView(tv1);
linearLayout2.addView(tv2);

// finally, add imageview and horizontal linearlayout to vertical linearlayout (rootView)
rootView.addView(imageView);
rootView.addView(linearLayout2);
```

LayoutParams

Cada `ViewGroup` (por ejemplo, `LinearLayout` , `RelativeLayout` , `CoordinatorLayout` , etc.) necesita almacenar información sobre las propiedades de sus hijos. Sobre la forma en que sus hijos están siendo presentados en el `ViewGroup` . Esta información se almacena en objetos de una clase contenedora `ViewGroup.LayoutParams` .

Para incluir parámetros específicos para un tipo de diseño particular, los `ViewGroups` usan subclases de la clase `ViewGroup.LayoutParams` .

Por ejemplo para

- `LinearLayout` es `LinearLayout.LayoutParams`
- `RelativeLayout` es `RelativeLayout.LayoutParams`
- `CoordinatorLayout` is `CoordinatorLayout.LayoutParams`
- ...

La mayoría de los `ViewGroups` reutilizan la capacidad de establecer `margins` para sus hijos, por lo que no subclase `ViewGroup.LayoutParams` directamente, sino que subclase `ViewGroup.MarginLayoutParams` (que es una subclase de `ViewGroup.LayoutParams`).

`LayoutParams` en xml

`LayoutParams`

objetos `LayoutParams` se crean en función del archivo `xml` diseño inflado.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:layout_gravity="right"
        android:gravity="bottom"
        android:text="Example text"
        android:textColor="@android:color/holo_green_dark"/>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@android:color/holo_green_dark"
        android:scaleType="centerInside"
        android:src="@drawable/example"/>

</LinearLayout>
```

Todos los parámetros que comienzan con `layout_` especifican cómo debe funcionar el layout **adjunto** . Cuando se infla el diseño, esos parámetros se envuelven en un objeto `LayoutParams` adecuado, que luego será utilizado por el `Layout` para posicionar correctamente una `View` particular dentro del grupo de `ViewGroup` . Otros atributos de una `View` están directamente relacionados con la `View` y son procesados por la propia `View` .

Para `TextView` :

- `layout_width` , `layout_height` y `layout_gravity` se almacenarán en un objeto `LinearLayout.LayoutParams` y `LinearLayout.LayoutParams` utilizados por `LinearLayout`
- `gravity` , `text` y `textColor` serán utilizados por `TextView`

Para `ImageView` :

- `layout_width` , `layout_height` y `layout_weight` se almacenarán en un objeto `LinearLayout.LayoutParams` y `LinearLayout.LayoutParams` utilizados por `LinearLayout`
- `background` , `scaleType` y `src` serán utilizados por el propio `ImageView`

Obtención del objeto `LayoutParams`

`getLayoutParams` es una `View's` método que permite recuperar una corriente `LayoutParams` objeto.

Debido a que el `LayoutParams` objeto se relaciona directamente con la **encerrando** `ViewGroup` , este método devolverá un valor no nulo sólo cuando `View` se une a la `ViewGroup` . Debe tener en cuenta que este objeto podría no estar presente en todo momento. Especialmente no debes depender de tenerlo dentro `View's` constructor `View's` .

```
public class ExampleView extends View {

    public ExampleView(Context context) {
        super(context);
        setupView(context);
    }

    public ExampleView(Context context, AttributeSet attrs) {
        super(context, attrs);
        setupView(context);
    }

    public ExampleView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        setupView(context);
    }

    private void setupView(Context context) {
        if (getLayoutParams().height == 50){  // DO NOT DO THIS!
                                              // This might produce NullPointerException
            doSomething();
        }
    }

    //...
}
```

Si desea depender de tener el objeto `LayoutParams` , debe usar el método `onAttachedToWindow` lugar.

```
public class ExampleView extends View {

    public ExampleView(Context context) {
        super(context);
    }

    public ExampleView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public ExampleView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    @Override
    protected void onAttachedToWindow() {
        super.onAttachedToWindow();
        if (getLayoutParams().height == 50) { // getLayoutParams() will NOT return null here
            doSomething();
        }
    }

    //...
}
```

Casting `LayoutParams` objeto

Es posible que deba usar características que son específicas de un `ViewGroup` particular (por ejemplo, es posible que desee cambiar las reglas de un `RelativeLayout`). Para ello, deberá saber

cómo convertir correctamente el objeto `ViewGroup.LayoutParams` .

Esto puede ser un poco confuso cuando se obtiene un objeto `LayoutParams` para una `View` secundaria que en realidad es otro `ViewGroup` .

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/outer_layout"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical">

    <FrameLayout
        android:id="@+id/inner_layout"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_gravity="right"/>

</LinearLayout>
```

IMPORTANTE: el tipo de objeto `LayoutParams` está directamente relacionado con el tipo del grupo de **vista** `ViewGroup` .

*Casting **incorrecto*** :

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
FrameLayout.LayoutParams par = (FrameLayout.LayoutParams) innerLayout.getLayoutParams();
                                        // INCORRECT! This will produce ClassCastException
```

*Casting **correcto*** :

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
LinearLayout.LayoutParams par = (LinearLayout.LayoutParams) innerLayout.getLayoutParams();
                                       // CORRECT! the enclosing layout is a LinearLayout
```

Lea Diseños en línea: https://riptutorial.com/es/android/topic/94/disenos

Capítulo 93: Editar texto

Examples

Trabajando con EditTexts

El EditText es el widget de entrada de texto estándar en aplicaciones de Android. Si el usuario necesita ingresar texto en una aplicación, esta es la forma principal de hacerlo.

Editar texto

Hay muchas propiedades importantes que se pueden configurar para personalizar el comportamiento de un EditText. Varios de estos se enumeran a continuación. Consulte la guía de campos de texto oficial para obtener más detalles del campo de entrada.

Uso

Se agrega un texto de edición a un diseño con todos los comportamientos predeterminados con el siguiente XML:

```
<EditText
    android:id="@+id/et_simple"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
</EditText>
```

Tenga en cuenta que un EditText es simplemente una extensión delgada de TextView y hereda todas las mismas propiedades.

Recuperando el valor

Obtener el valor del texto introducido en un EditText es el siguiente:

```
EditText simpleEditText = (EditText) findViewById(R.id.et_simple);
String strValue = simpleEditText.getText().toString();
```

Mayor personalización de entrada

Podríamos querer limitar la entrada a una sola línea de texto (evitar nuevas líneas):

```
<EditText
  android:singleLine="true"
  android:lines="1"
/>
```

Puede limitar los caracteres que pueden ingresarse en un campo usando el atributo de dígitos:

```
<EditText
  android:inputType="number"
  android:digits="01"
/>
```

Esto restringiría los dígitos ingresados a solo "0" y "1". Podríamos querer limitar el número total de caracteres con:

```
<EditText
  android:maxLength="5"
/>
```

Usando estas propiedades podemos definir el comportamiento de entrada esperado para los campos de texto.

Ajuste de colores

Puede ajustar el color de fondo de resaltado del texto seleccionado dentro de un texto de edición con la propiedad `android:textColorHighlight` :

```
<EditText
    android:textColorHighlight="#7cff88"
/>
```

Visualización de sugerencias de marcador de posición

Es posible que desee configurar la sugerencia para que el control EditText solicite a un usuario una entrada específica con:

```
<EditText
    ...
    android:hint="@string/my_hint">
</EditText>
```

Consejos

Cambiando el color de la línea de fondo

Suponiendo que está utilizando la biblioteca AppCompat, puede anular los estilos colorControlNormal, colorControlActivated y colorControlHighlight:

```
<style name="Theme.App.Base" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="colorControlNormal">#d32f2f</item>
    <item name="colorControlActivated">#ff5722</item>
    <item name="colorControlHighlight">#f44336</item>
</style>
```

Si no ve estos estilos aplicados dentro de un DialogFragment, hay un error conocido cuando se usa el LayoutInflater pasado en el método onCreateView ().

El problema ya se ha solucionado en la biblioteca AppCompat v23. Consulte esta guía sobre

cómo actualizar. Otra solución temporal es usar el diseño de la actividad en lugar del que se pasó al método onCreateView ():

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = getActivity().getLayoutInflater().inflate(R.layout.dialog_fragment,
container);
  }
```

Escuchando para la entrada de EditText

Echa un vistazo a las notas clásicas de los oyentes de eventos para ver cómo escuchar los cambios en un EditText y realizar una acción cuando se producen esos cambios.

Visualización de comentarios de etiquetas flotantes

Tradicionalmente, el EditText oculta el mensaje de sugerencia (explicado anteriormente) después de que el usuario comienza a escribir. Además, cualquier mensaje de error de validación tenía que ser administrado manualmente por el desarrollador.

Con `TextInputLayout` puede configurar una etiqueta flotante para mostrar sugerencias y mensajes de error. Puedes encontrar más detalles aquí .

Personalizando el tipo de entrada

Los campos de texto pueden tener diferentes tipos de entrada, como número, fecha, contraseña o dirección de correo electrónico. El tipo determina qué tipo de caracteres se permiten dentro del campo y puede solicitar al teclado virtual que optimice su diseño para los caracteres de uso frecuente.

De forma predeterminada, cualquier contenido de texto dentro de un control `EditText` se muestra como texto sin formato. Al establecer el atributo `inputType` , podemos facilitar la entrada de diferentes tipos de información, como números de teléfono y contraseñas:

```
<EditText
    ...
    android:inputType="phone">
</EditText>
```

Los tipos de entrada más comunes incluyen:

Tipo	Descripción
textoUri	Texto que se utilizará como URI
textoEmailDirección	Texto que se utilizará como dirección de correo electrónico.
textPersonName	Texto que es el nombre de una persona.
contraseña de texto	Texto que es una contraseña que debe ser ocultada

Tipo	Descripción
número	Un campo solo numérico
teléfono	Para ingresar un número de teléfono
fecha	Para ingresar una fecha
hora	Por entrar un tiempo
textMultiLine	Permitir múltiples líneas de texto en el campo.

El `android:inputType` también le permite especificar ciertos comportamientos de teclado, tales como si poner en mayúscula todas las palabras nuevas o usar características como autocompletar y sugerencias de ortografía.
Estos son algunos de los valores comunes de tipo de entrada que definen los comportamientos del teclado:

Tipo	Descripción
textCapSentences	Teclado de texto normal que pone en mayúscula la primera letra para cada nueva oración
textCapWords	Teclado de texto normal que pone en mayúscula cada palabra. Bueno para títulos o nombres de personas
textAutoCorrect	Teclado de texto normal que corrige las palabras mal escritas.

Puede configurar múltiples `inputType` atributos si es necesario (separados por '|').
Ejemplo:

```
<EditText
    android:id="@+id/postal_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/postal_address_hint"
    android:inputType="textPostalAddress|
                       textCapWords|
                       textNoSuggestions" />
```

Puede ver una lista de todos los tipos de entrada disponibles aquí .

atributo `inputype`

atributo `inputype` en el widget `EditText` : *(probado en Android 4.4.3 y 2.3.3)*

```
<EditText android:id="@+id/et_test" android:inputType="?????"/>
```

textLongMessage = Teclado: alfabeto / predeterminado. Botón de entrar: Enviar / Siguiente.

Emoción: si. Caso: minúscula. Sugerencia: si. Añadir. carboniza **a:, y.** y todo

textFilter = Teclado: alfabeto / predeterminado. Botón de entrar: Enviar / Siguiente. Emoción: si. Caso: minúscula. **Sugerencia: no** . Añadir. carboniza **a:, y.** y todo

textCapWords = Teclado: alfabeto / predeterminado. Botón de entrar: Enviar / Siguiente. Emoción: si. **Caso: Camel Case** . Sugerencia: si. Añadir. carboniza **a:, y.** y todo

textCapSentences = Teclado: alfabeto / predeterminado. Botón de entrar: Enviar / Siguiente. Emoción: si. **Caso: Caso de sentencia** . Sugerencia: si. Añadir. carboniza **a:, y.** y todo

tiempo = teclado: numérico. Botón de entrar: Enviar / Siguiente. Emoción: no. Caso: -. **Sugerencia: no** . Añadir. caracteres::

textMultiLine = Teclado: alfabeto / predeterminado. **Botón enter: siguiente línea** . Emoción: si. Caso: minúscula. Sugerencia: si. Añadir. carboniza **a:, y.** y todo

número = teclado: numérico . Botón de entrar: Enviar / Siguiente. Emoción: no. Caso: -. Sugerencia: no. **Añadir. caracteres: nada**

textEmailAddress = Teclado: alfabeto / predeterminado. Botón de entrar: Enviar / Siguiente. **Emoción: no** . Caso: minúscula. **Sugerencia: no** . Añadir. caracteres: @ y **.** y todo

(Sin tipo) = Teclado: alfabeto / predeterminado. **Botón enter: siguiente línea** . Emoción: si. Caso: minúscula. Sugerencia: si. Añadir. carboniza **a:, y.** y todo

textPassword = Teclado: alfabeto / predeterminado. Botón de entrar: Enviar / Siguiente. Emoción: no. Caso: minúscula. **Sugerencia: no** . Añadir. carboniza **a:, y.** y todo

texto = Teclado: Teclado: alfabeto / predeterminado. Botón de entrar: Enviar / Siguiente. Emoción: si. Caso: minúscula. Sugerencia: si. Añadir. carboniza **a:, y.** y todo

textShortMessage = Teclado: alfabeto / predeterminado. **Botón de entrar: emoción** . Emoción: si. Caso: minúscula. Sugerencia: si. Añadir. carboniza **a:, y.** y todo

textUri = Teclado: alfabeto / predeterminado. Botón de entrar: Enviar / Siguiente. Emoción: no. Caso: minúscula. **Sugerencia: no** . Añadir. caracteres: **/** y **.** y todo

textCapCharacters = Teclado: alfabeto / predeterminado. Botón de entrar: Enviar / Siguiente. Emoción: si. **Caso: MAYÚSCULAS** . Sugerencia: si. Añadir. carboniza **a:, y.** y todo

teléfono = teclado: numérico . Botón de entrar: Enviar / Siguiente. Emoción: no. Caso: -. **Sugerencia: no** . Añadir. caracteres: *** #. - / () WPN, + **

textPersonName = Teclado: alfabeto / predeterminado. Botón de entrar: Enviar / Siguiente. Emoción: si. Caso: minúscula. Sugerencia: si. Añadir. carboniza **a:, y.** y todo

Nota: `Auto-capitalization` configuración de `Auto-capitalization` cambiará el comportamiento predeterminado.

Nota 2: En el `Numeric keyboard` , TODOS los números son 1234567890 en inglés.

Nota 3: la configuración de `Correction/Suggestion` cambiará el comportamiento predeterminado.

Ocultar SoftKeyboard

Ocultar Softkeyboard es un **requisito básico por lo** general cuando se trabaja con EditText. El teclado de teclado por *defecto* solo puede cerrarse presionando el botón Atrás y, por lo tanto, la mayoría de los desarrolladores usan InputMethodManager para forzar a Android a ocultar el teclado virtual que llama a hideSoftInputFromWindow y pasa el token de la ventana que contiene su vista enfocada. El código para hacer lo siguiente:

```
public void hideSoftKeyboard()
{
        InputMethodManager inputMethodManager = (InputMethodManager)
getSystemService(Activity.INPUT_METHOD_SERVICE);
        inputMethodManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(), 0);
}
```

El código es directo, pero otro problema importante que surge es que se debe llamar a la función de ocultar cuando ocurre algún evento. ¿Qué hacer cuando necesita que el Softkeyboard esté oculto al presionar en otro lugar que no sea su EditText? El siguiente código proporciona una función clara que debe llamarse en su método onCreate () solo una vez.

```
public void setupUI(View view)
{
        String s = "inside";
        //Set up touch listener for non-text box views to hide keyboard.
        if (!(view instanceof EditText)) {

            view.setOnTouchListener(new View.OnTouchListener() {

                public boolean onTouch(View v, MotionEvent event) {
                    hideSoftKeyboard();
                    return false;
                }

            });
        }

        //If a layout container, iterate over children and seed recursion.
        if (view instanceof ViewGroup) {

            for (int i = 0; i < ((ViewGroup) view).getChildCount(); i++) {

                View innerView = ((ViewGroup) view).getChildAt(i);

                setupUI(innerView);
            }
        }
}
```

Icono o botón dentro de Texto de edición personalizado y su acción y haga clic en escuchas.

Este ejemplo ayudará a tener el texto de edición con el icono en el lado derecho.

> Nota: En esto solo estoy usando setCompoundDrawablesWithIntrinsicBounds, así que si desea cambiar la posición del ícono, puede lograrlo usando setCompoundDrawablesWithIntrinsicBounds en setIcon.

```
public class MKEditText extends AppCompatEditText {

    public interface IconClickListener {
        public void onClick();
    }

    private IconClickListener mIconClickListener;

    private static final String TAG = MKEditText.class.getSimpleName();

    private final int EXTRA_TOUCH_AREA = 50;
    private Drawable mDrawable;
    private boolean touchDown;

    public MKEditText(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    public MKEditText(Context context) {
        super(context);
    }

    public MKEditText(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public void showRightIcon() {
        mDrawable = ContextCompat.getDrawable(getContext(), R.drawable.ic_android_black_24dp);

        setIcon();
    }

    public void setIconClickListener(IconClickListener iconClickListener) {
        mIconClickListener = iconClickListener;
    }

    private void setIcon() {
        Drawable[] drawables = getCompoundDrawables();

        setCompoundDrawablesWithIntrinsicBounds(drawables[0], drawables[1], mDrawable,
drawables[3]);

        setInputType(InputType.TYPE_CLASS_TEXT | InputType.TYPE_TEXT_VARIATION_PASSWORD);
        setSelection(getText().length());
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        final int right = getRight();
        final int drawableSize = getCompoundPaddingRight();
        final int x = (int) event.getX();
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                if (x + EXTRA_TOUCH_AREA >= right - drawableSize && x <= right +
```

```
EXTRA_TOUCH_AREA) {
                    touchDown = true;
                    return true;
                }
                break;
            case MotionEvent.ACTION_UP:
                if (x + EXTRA_TOUCH_AREA >= right - drawableSize && x <= right +
EXTRA_TOUCH_AREA && touchDown) {
                    touchDown = false;
                    if (mIconClickListener != null) {
                        mIconClickListener.onClick();
                    }
                    return true;
                }
                touchDown = false;
                break;

        }
        return super.onTouchEvent(event);
    }
}
```

Si desea cambiar el área táctil, puede cambiar los valores predeterminados de EXTRA_TOUCH_AREA que di como 50.

Y para Habilitar el botón y hacer clic en el oyente, puede llamar desde su Actividad o Fragmento como este,

```
MKEditText mkEditText = (MKEditText) findViewById(R.id.password);
mkEditText.showRightIcon();
mkEditText.setIconClickListener(new MKEditText.IconClickListener() {
            @Override
            public void onClick() {
              // You can do action here for the icon.

            }
        });
```

Lea Editar texto en línea: https://riptutorial.com/es/android/topic/5843/editar-texto

Capítulo 94: Ejecución instantánea en Android Studio

Observaciones

La ejecución instantánea es un comportamiento extendido para los comandos de ejecución y depuración que permite una depuración más rápida al no requerir una compilación y reinstalación completas para el cambio de eevry realizado en el código de su aplicación.

> Introducido en Android Studio 2.0, Instant Run es un comportamiento de los comandos Ejecutar y Depurar que reduce significativamente el tiempo entre las actualizaciones de su aplicación. Aunque su primera compilación puede tardar más tiempo en completarse, Instant Run empuja las actualizaciones posteriores a su aplicación sin crear un nuevo APK, por lo que los cambios son visibles mucho más rápidamente.
>
> Instant Run solo se admite cuando implementa la variante de compilación de depuración, usa el complemento de Android para Gradle versión 2.0.0 o superior, y establece minSdkVersion a 15 o superior en el archivo build.gradle de nivel de módulo de tu aplicación. Para obtener el mejor rendimiento, establezca minSdkVersion en 21 o superior.
>
> Después de implementar una aplicación, aparece un pequeño icono amarillo de rayo dentro del botón Ejecutar (o botón Depurar), que indica que la Ejecución instantánea está lista para enviar actualizaciones la próxima vez que haga clic en el botón. En lugar de crear un nuevo APK, solo empuja esos cambios nuevos y, en algunos casos, la aplicación ni siquiera necesita reiniciarse, pero muestra inmediatamente el efecto de esos cambios de código.
>
> La ejecución instantánea envía el código y los recursos actualizados a su dispositivo o emulador conectado mediante un intercambio en caliente, un intercambio en caliente o un intercambio en frío. Determina automáticamente el tipo de swap a realizar en función del tipo de cambio realizado. El video anterior proporciona detalles interesantes sobre cómo funciona todo esto bajo el capó. Sin embargo, consulte la siguiente tabla para obtener un resumen rápido de cómo se comporta Instant Run cuando presiona ciertos cambios de código en un dispositivo de destino.

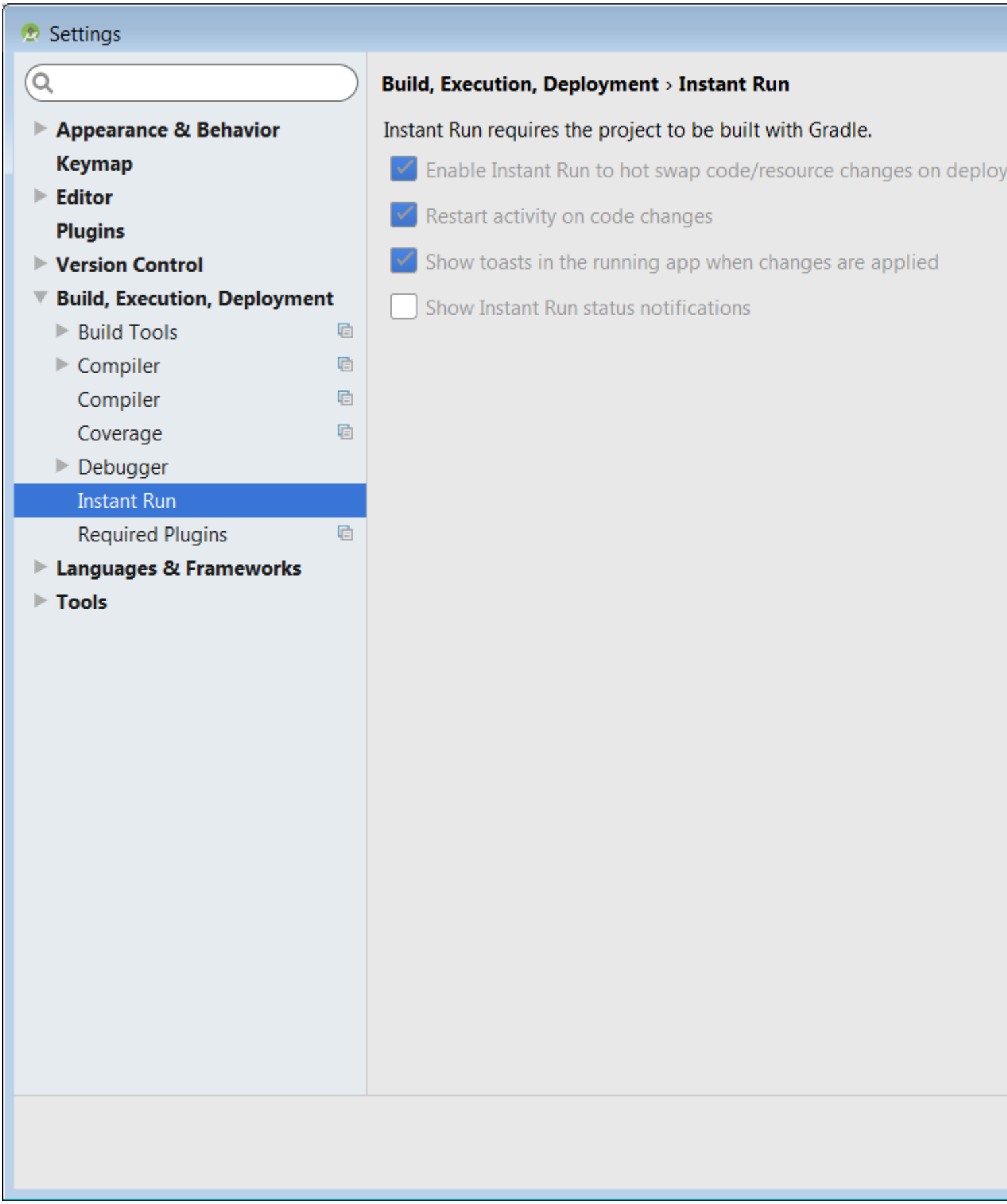
Documentación

Examples

Habilitar o deshabilitar la ejecución instantánea

1. Abra el cuadro de diálogo Configuración o Preferencias:
   - En Windows o Linux, seleccione `File` > `Settings` en el menú principal.

    - En Mac OSX, seleccione `Android Studio` > `Preferences` en el menú principal.
2. Navegue para `Build, Execution, Deployment` > `Compiler` .
3. En el campo de texto junto a Opciones de línea de comandos, ingrese sus opciones de línea de comandos.
4. Haga clic en Aceptar para guardar y salir.



La opción superior es la ejecución instantánea. Marque / desmarque esa casilla.

Documentación

Tipos de swaps de código en ejecución instantánea

Existen tres tipos de intercambios de código que la ejecución instantánea permite admitir una aplicación de depuración y ejecución más rápida desde su código en Android Studio.

- Intercambio en caliente
- Intercambio de calor
- Cambio en frío

¿Cuándo se activan cada uno de estos swaps?

HOT SWAP se activa cuando se cambia la implementación de un método existente.

WARM SWAP se activa cuando se modifica o elimina un recurso existente (cualquier elemento en la carpeta res)

CAMBIO EN FRÍO siempre que haya un cambio de código estructural en el código de su aplicación, por ejemplo

1. Añadir, eliminar o cambiar:

- una anotación
- un campo de instancia
- un campo estático
- una firma de método estático
- una firma de método de instancia

2. Cambiar de qué clase padre hereda la clase actual
3. Cambiar la lista de interfaces implementadas.
4. Cambiar el inicializador estático de una clase
5. Reordenar los elementos de diseño que utilizan ID de recursos dinámicos

¿Qué sucede cuando ocurre un intercambio de código?

Los cambios de **HOT SWAP** son visibles al instante, tan pronto como se realiza la próxima llamada al método cuya implementación se cambia.

WARM SWAP reinicia la actividad actual.

COLD SWAP reinicia toda la aplicación (sin reinstalar)

Cambios de código no admitidos al usar la ejecución instantánea

Hay algunos cambios en los que Instant no funcionará y una compilación y reinstalación completas de su aplicación sucederán como solía suceder antes de que naciera Instant Run.

1. Cambia el manifiesto de la aplicación.
2. Cambiar recursos referenciados por el manifiesto de la aplicación.
3. Cambiar un elemento de la interfaz de usuario de Android Widget (requiere un Limpiar y

volver a ejecutar)

Documentación

Lea Ejecución instantánea en Android Studio en línea:
https://riptutorial.com/es/android/topic/2119/ejecucion-instantanea-en-android-studio

Capítulo 95: El archivo de manifiesto

Introducción

El manifiesto es un archivo obligatorio llamado exactamente "AndroidManifest.xml" y se encuentra en el directorio raíz de la aplicación. Especifica el nombre de la aplicación, el icono, el nombre del paquete de Java, la versión, la declaración de Actividades, los Servicios, los permisos de la aplicación y otra información.

Examples

Declarando componentes

La tarea principal del manifiesto es informar al sistema sobre los componentes de la aplicación. Por ejemplo, un archivo de manifiesto puede declarar una actividad de la siguiente manera:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

En el elemento `<application>` , el atributo `android:icon` apunta a recursos para un icono que identifica la aplicación.

En el elemento, el atributo `android:name` especifica el nombre de clase completamente calificado de la subclase Activity y el atributo android: label especifica una cadena para usar como la etiqueta visible para el usuario para la actividad.

Debe declarar todos los componentes de la aplicación de esta manera:

- Elementos de `<activity>` para actividades.

- `<service>` elementos para servicios

- Elementos `<receiver>` para receptores de difusión.

- `<provider>` elementos para proveedores de contenido

Las actividades, servicios y proveedores de contenido que incluye en su fuente pero que no declara en el manifiesto no son visibles para el sistema y, por lo tanto, nunca pueden ejecutarse. Sin embargo, los receptores de difusión pueden declararse en el manifiesto o crearse dinámicamente en código (como objetos `BroadcastReceiver`) y registrarse en el sistema llamando a `registerReceiver()` .

Para obtener más información sobre cómo estructurar el archivo de manifiesto para su aplicación, consulte la documentación del archivo AndroidManifest.xml.

Declarando permisos en su archivo manifiesto

Cualquier permiso requerido por su aplicación para acceder a una parte protegida de la API o para interactuar con otras aplicaciones debe ser declarado en su archivo `AndroidManifest.xml` . Esto se hace usando la etiqueta `<uses-permission />` .

Sintaxis

```
<uses-permission android:name="string"
   android:maxSdkVersion="integer"/>
```

android: nombre: este es el nombre del permiso requerido

android: maxSdkVersion: el nivel de API más alto en el que se debe otorgar este permiso a su aplicación. La configuración de este permiso es opcional y solo debe establecerse si el permiso que requiere su aplicación ya no es necesario en un determinado nivel de API.

Muestra de AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.samplepackage">

    <!-- request internet permission -->
    <uses-permission android:name="android.permission.INTERNET" />

    <!-- request camera permission -->
    <uses-permission android:name="android.permission.CAMERA"/>

    <!-- request permission to write to external storage -->
    <uses-permission
         android:name="android.permission.WRITE_EXTERNAL_STORAGE"
         android:maxSdkVersion="18" />

    <application>....</application>
</manifest>
```

* También vea el tema Permisos .

Lea El archivo de manifiesto en línea: https://riptutorial.com/es/android/topic/1848/el-archivo-de-manifiesto

Capítulo 96: Emulador

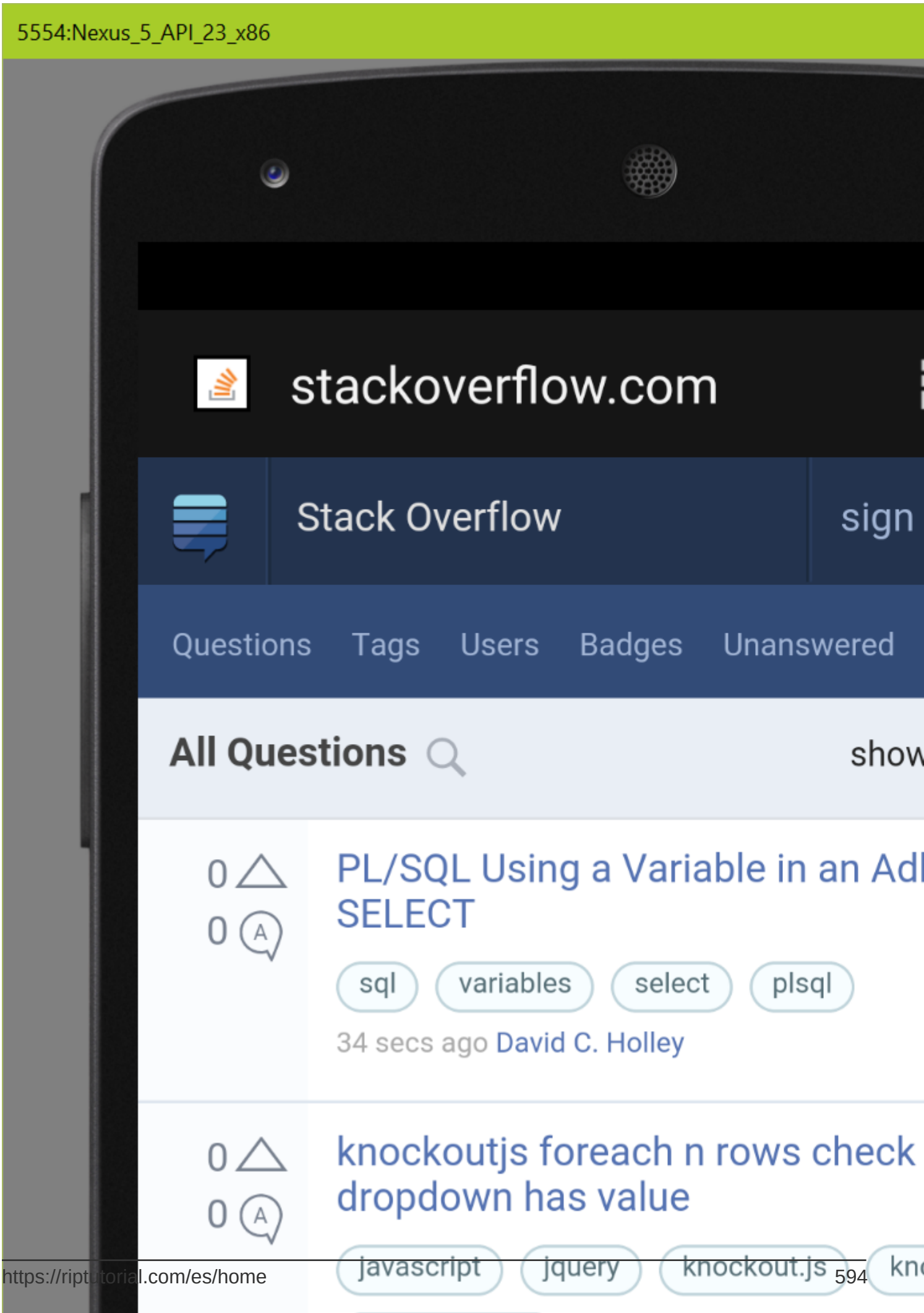
Observaciones

AVD significa *dispositivo virtual de Android*

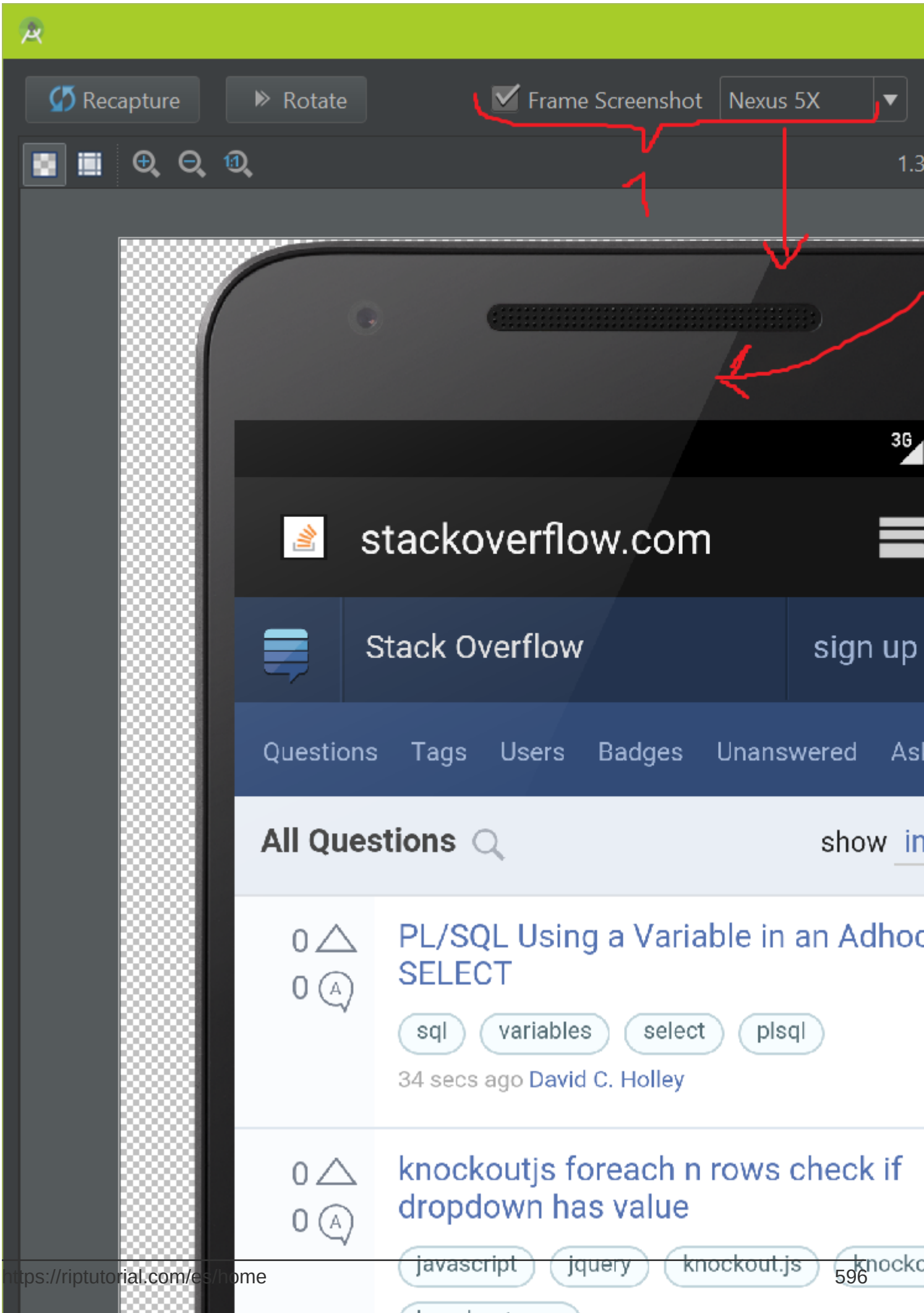
Examples

Tomando capturas de pantalla

Si desea tomar una captura de pantalla del Android Emulator (2.0), solo tiene que presionar `Ctrl` + `S` o hacer clic en el icono de la cámara en la barra lateral:


5554:Nexus_5_API_23_x86
stackoverflow.com
Stack Overflow
sign
Questions
Tags
Users
Badges
Unanswered
All Questions
show
0
0
PL/SQL Using a Variable in an Adl
SELECT
sql
variables
select
plsql
34 secs ago David C. Holley
0
0
knockoutjs foreach n rows check
dropdown has value
javascript
jquery
knockout.js

2. Una sombra debajo del marco del dispositivo.
3. Un brillo de pantalla a través del marco del dispositivo y captura de pantalla.

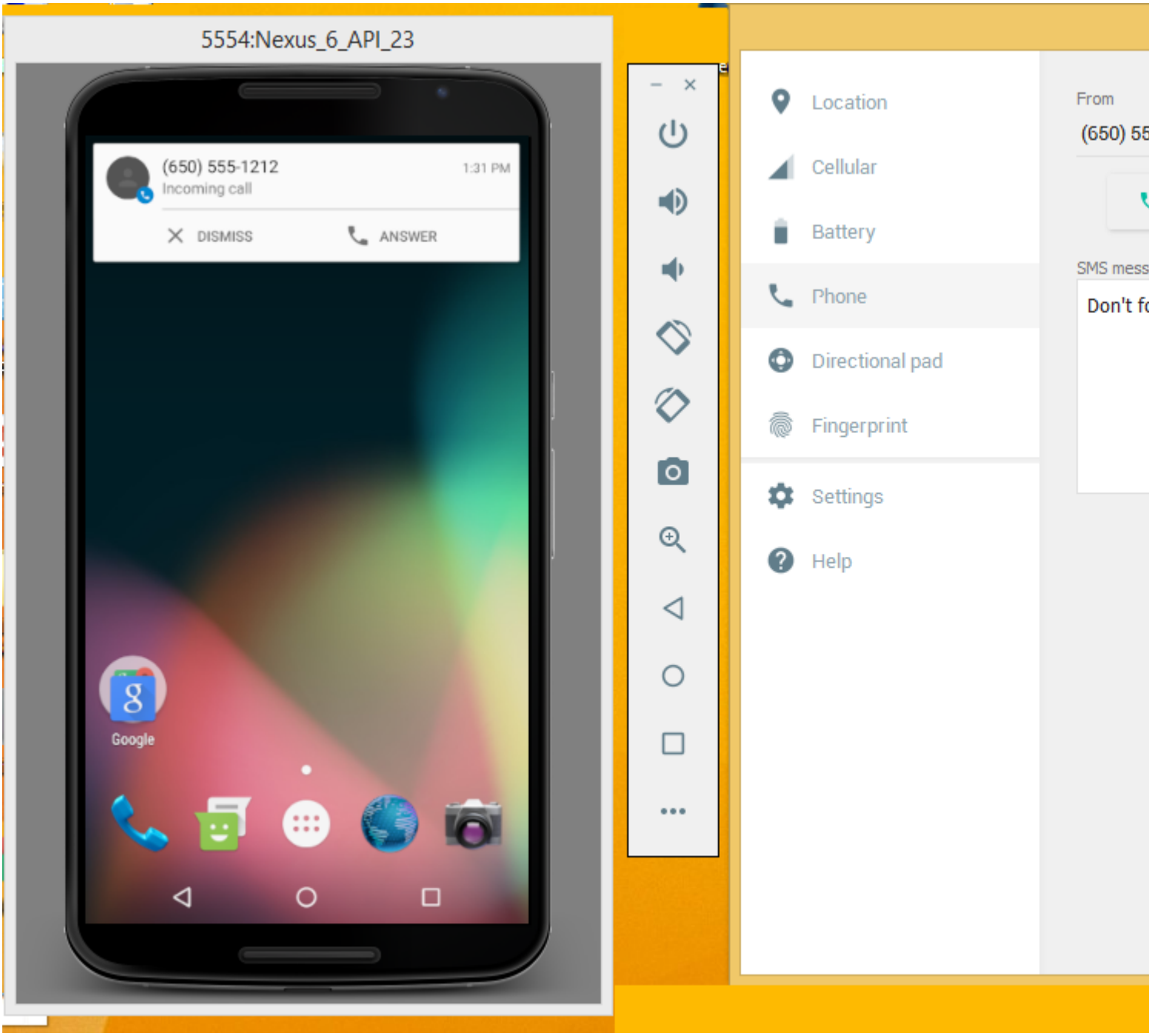

Recapture
Rotate
Frame Screenshot
Nexus 5X
3G
stackoverflow.com
Stack Overflow
sign up
Questions
Tags
Users
Badges
Unanswered
All Questions
show
0
0
PL/SQL Using a Variable in an Adhoc SELECT
sql
variables
select
plsql
34 secs ago David C. Holley
0
0
knockoutjs foreach n rows check if dropdown has value
javascript
jquery
knockout.js

`AVD Manager` o haciendo clic en el icono de AVD Manager en la barra de herramientas, que es la segunda en la captura de pantalla a continuación.



Simular llamada

Para simular una llamada telefónica, presione el botón 'Controles extendidos' indicado por tres puntos, elija 'Teléfono' y seleccione 'Llamar'. Opcionalmente también puede cambiar el número de teléfono.



Resolviendo errores al iniciar el emulador

En primer lugar, asegúrese de haber habilitado la ' **virtualización'** en la configuración de su

BIOS.

Inicie el **Android SDK Manager** , seleccione **Extras** y luego seleccione **Intel Hardware Accelerated Execution Manager** y espere hasta que se complete la descarga. Si aún no funciona, abra la carpeta de su SDK y ejecute
`/extras/intel/Hardware_Accelerated_Execution_Manager/IntelHAXM.exe` .

Siga las instrucciones en pantalla para completar la instalación.

O para OS X puede hacerlo sin indicaciones en pantalla como esta:
`/extras/intel/Hardware_Accelerated_Execution_Manager/HAXM\ installation`

> Si su CPU no es compatible con VT-x o SVM, no puede usar imágenes de Android basadas en x86. Por favor, use imágenes basadas en ARM en su lugar.

Una vez completada la instalación, confirme que el controlador de virtualización está funcionando correctamente abriendo una ventana del símbolo del sistema y ejecutando el siguiente comando:
`sc query intelhaxm`

Para ejecutar un emulador basado en x86 con aceleración de máquina virtual: si está ejecutando el emulador desde la línea de comandos, simplemente especifique un AVD: `emulator -avd <avd_name>` basado en x86

Si sigue correctamente todos los pasos mencionados anteriormente, entonces seguramente debería poder ver su AVD con HAXM normalmente.

Lea Emulador en línea: https://riptutorial.com/es/android/topic/122/emulador

Capítulo 97: Entrenador de animales

Observaciones

Un controlador se puede usar fácilmente para ejecutar código después de un período de tiempo retrasado. También es útil para ejecutar el código repetidamente después de un período de tiempo específico llamando nuevamente al método Handler.postDelayed () desde el método run () de Runnable.

Examples

Uso de un controlador para ejecutar código después de un período de tiempo retrasado

Ejecutando código después de 1.5 segundos:

```
Handler handler = new Handler();
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        //The code you want to run after the time is up
    }
}, 1500); //the time you want to delay in milliseconds
```

Ejecutando código repetidamente cada 1 segundo:

```
Handler handler = new Handler();
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        handler.postDelayed(this, 1000);
    }
}, 1000); //the time you want to delay in milliseconds
```

HandlerThreads y comunicación entre hilos.

Como los `Handler` se utilizan para enviar `Message` y `Runnable` a la cola de mensajes de un subproceso, es fácil implementar una comunicación basada en eventos entre varios subprocesos. Cada hilo que tiene un `Looper` puede recibir y procesar mensajes. Un `HandlerThread` es un subproceso que implementa tal `Looper` , por ejemplo, el subproceso principal (UI Thread) implementa las características de un `HandlerThread` .

Creación de un controlador para el hilo actual

```
Handler handler = new Handler();
```

Creación de un controlador para el subproceso principal (subproceso de la interfaz de usuario)

```
Handler handler = new Handler(Looper.getMainLooper());
```

Enviar un Runnable de otro hilo al hilo principal

```
new Thread(new Runnable() {
    public void run() {
        // this is executed on another Thread

        // create a Handler associated with the main Thread
        Handler handler = new Handler(Looper.getMainLooper());

        // post a Runnable to the main Thread
        handler.post(new Runnable() {
            public void run() {
                // this is executed on the main Thread
            }
        });
    }
}).start();
```

Creando un Handler para otro HandlerThread y enviándole eventos

```
// create another Thread
HandlerThread otherThread = new HandlerThread("name");

// create a Handler associated with the other Thread
Handler handler = new Handler(otherThread.getLooper());

// post an event to the other Thread
handler.post(new Runnable() {
    public void run() {
        // this is executed on the other Thread
    }
});
```

Detener el manejador de la ejecución

Para detener la ejecución del controlador, elimine la devolución de llamada adjunta utilizando el ejecutable ejecutable dentro de él:

```
Runnable my_runnable = new Runnable() {
    @Override
    public void run() {
        // your code here
    }
};
```

```
public Handler handler = new Handler(); // use 'new Handler(Looper.getMainLooper());' if you
want this handler to control something in the UI
// to start the handler
public void start() {
    handler.postDelayed(my_runnable, 10000);
}

// to stop the handler
public void stop() {
    handler.removeCallbacks(my_runnable);
}

// to reset the handler
public void restart() {
    handler.removeCallbacks(my_runnable);
    handler.postDelayed(my_runnable, 10000);
}
```

Use el controlador para crear un temporizador (similar a javax.swing.Timer)

Esto puede ser útil si estás escribiendo un juego o algo que necesita ejecutar un fragmento de código cada pocos segundos.

```
import android.os.Handler;

public class Timer {
    private Handler handler;
    private boolean paused;

    private int interval;

    private Runnable task = new Runnable () {
        @Override
        public void run() {
            if (!paused) {
                runnable.run ();
                Timer.this.handler.postDelayed (this, interval);
            }
        }
    };

    private Runnable runnable;

    public int getInterval() {
        return interval;
    }

    public void setInterval(int interval) {
        this.interval = interval;
    }

    public void startTimer () {
        paused = false;
        handler.postDelayed (task, interval);
    }

    public void stopTimer () {
        paused = true;
```

```
    }

    public Timer (Runnable runnable, int interval, boolean started) {
        handler = new Handler ();
        this.runnable = runnable;
        this.interval = interval;
        if (started)
            startTimer ();
    }
}
```

Ejemplo de uso:

```
Timer timer = new Timer(new Runnable() {
    public void run() {
        System.out.println("Hello");
    }
}, 1000, true)
```

Este código imprimirá "Hola" cada segundo.

Lea Entrenador de animales en línea: https://riptutorial.com/es/android/topic/1425/entrenador-de-animales

Capítulo 98: Escribir pruebas de interfaz de usuario - Android

Introducción

El enfoque de este documento es representar objetivos y formas de escribir la interfaz de usuario de Android y las pruebas de integración. Google proporciona el expreso y el UIAutomator, por lo que el enfoque debe estar en torno a estas herramientas y sus respectivos envoltorios, por ejemplo, Appium, Spoon, etc.

Sintaxis

- **Recurso inactivo**
- Cadena getName (): devuelve el nombre del recurso inactivo (utilizado para el registro y la idempotencia del registro).
- boolean isIdleNow (): devuelve true si el recurso está actualmente inactivo.
- void registerIdleTransitionCallback (IdlingResource.ResourceCallback callback): registra el IdlingResource.ResourceCallback dado con el recurso

Observaciones

Reglas de JUnit:

Como puede ver en el ejemplo de MockWebServer y en ActivityTestRule, todas se incluyen en la categoría de reglas JUnit que puede crear usted mismo, que luego deben ejecutarse para cada prueba que defina su comportamiento @see: https://github.com/junit-team/junit4/ wiki / rules

Apio

Parámetros

Dado que los parámetros tienen algunos problemas al colocarlos aquí hasta que se resuelva el error de documentación:

Parámetro	Detalles
Actividad de clase clase	que actividad comenzar
initialTouchMode	en caso de que la actividad se coloque en modo táctil al inicio: https://android-developers.blogspot.de/2008/12/touch-mode.html

Parámetro	Detalles
launchActivity	Es cierto si la Actividad debe iniciarse una vez por método de prueba. Se lanzará antes del primer método Antes, y terminará después del último método Después.

Examples

Ejemplo de MockWebServer

En caso de que sus actividades, fragmentos e IU requieran algo de procesamiento en segundo plano, una buena cosa para usar es un MockWebServer que se ejecuta localmente en un dispositivo Android que brinda un entorno cerrado y comprobable para su IU.

https://github.com/square/okhttp/tree/master/mockwebserver

El primer paso es incluir la dependencia de Gradle:

```
testCompile 'com.squareup.okhttp3:mockwebserver:(insert latest version)'
```

Ahora los pasos para ejecutar y usar el servidor simulado son:

- crear un objeto de servidor simulado
- inícielo en la dirección y el puerto específicos (generalmente localhost: número de puerto)
- Encolar respuestas para solicitudes específicas
- comienza la prueba

Esto está muy bien explicado en la página github de mockwebserver pero en nuestro caso queremos algo mejor y reutilizable para todas las pruebas, y las reglas de JUnit entrarán en juego aquí:

```
/**
 *JUnit  rule that starts and stops a mock web server for test runner
*/
 public class MockServerRule extends UiThreadTestRule {

 private MockWebServer mServer;

 public static final int MOCK_WEBSERVER_PORT = 8000;

    @Override
    public Statement apply(final Statement base, Description description) {
        return new Statement() {
            @Override
            public void evaluate() throws Throwable {
                startServer();
                try {
                    base.evaluate();
                } finally {
                    stopServer();
                }
            }
```

```
        };
    }

    /**
     * Returns the started web server instance
     *
     * @return mock server
     */
    public MockWebServer server() {
        return mServer;
    }

    public void startServer() throws IOException, NoSuchAlgorithmException {
        mServer = new MockWebServer();
        try {
            mServer(MOCK_WEBSERVER_PORT);
        } catch (IOException e) {
            throw new IllegalStateException(e,"mock server start issue");
        }
    }

    public void stopServer() {
        try {
            mServer.shutdown();
        } catch (IOException e) {
            Timber.e(e, "mock server shutdown error”);
        }
    }
}
```

Ahora asumamos que tenemos exactamente la misma actividad que en el ejemplo anterior, solo en este caso cuando presionamos el botón, la aplicación buscará algo de la red, por ejemplo: https://someapi.com/name

Esto devolvería algunas cadenas de texto que se concatenarían en el texto de la barra de snack, por ejemplo, el NOMBRE + texto que ingresaste.

```
/**
* Testing of the snackbar activity with networking.
**/
@RunWith(AndroidJUnit4.class)
@LargeTest
public class SnackbarActivityTest{
    //espresso rule which tells which activity to start
    @Rule
    public final ActivityTestRule<SnackbarActivity> mActivityRule =
        new ActivityTestRule<>(SnackbarActivity.class, true, false);

    //start mock web server
    @Rule
    public final MockServerRule mMockServerRule = new MockServerRule();

    @Override
    public void tearDown() throws Exception {
       //same as previous example
    }

    @Override
    public void setUp() throws Exception {
```

```
       //same as previous example

       **//IMPORTANT:** point your application to your mockwebserver endpoint e.g.
       MyAppConfig.setEndpointURL("http://localhost:8000");
    }

    /**
    *Test methods should always start with "testXYZ" and it is a good idea to
    *name them after the intent what you want to test
    **/
    @Test
    public void testSnackbarIsShown() {
        //setup mockweb server
        mMockServerRule.server().setDispatcher(getDispatcher());

        mActivityRule.launchActivity(null);
        //check is our text entry displayed and enter some text to it
        String textToType="new snackbar text";
        onView(withId(R.id.textEntry)).check(matches(isDisplayed()));
        //we check is our snackbar showing text from mock webserver plus the one we typed
        onView(withId(R.id.textEntry)).perform(typeText("JazzJackTheRabbit" + textToType));
        //click the button to show the snackbar
        onView(withId(R.id.shownSnackbarBtn)).perform(click());
        //assert that a view with snackbar_id with text which we typed and is displayed
        onView(allOf(withId(android.support.design.R.id.snackbar_text),
        withText(textToType))) .check(matches(isDisplayed()));
    }

     /**
     *creates a mock web server dispatcher with prerecorded requests and responses
     **/
    private Dispatcher getDispatcher() {
        final Dispatcher dispatcher = new Dispatcher() {
            @Override
            public MockResponse dispatch(RecordedRequest request) throws InterruptedException
{
                if (request.getPath().equals("/name")){
                    return new MockResponse().setResponseCode(200)
                            .setBody("JazzJackTheRabbit");
                }
                throw new IllegalStateException("no mock set up for " + request.getPath());
            }
        };
        return dispatcher;
    }
```

Yo sugeriría envolver al despachador en una especie de Builder para que pueda encadenar fácilmente y agregar nuevas respuestas para sus pantallas. p.ej

```
 return newDispatcherBuilder()
            .withSerializedJSONBody("/authenticate", Mocks.getAuthenticationResponse())
            .withSerializedJSONBody("/getUserInfo", Mocks.getUserInfo())
            .withSerializedJSONBody("/checkNotBot", Mocks.checkNotBot());
```

IdlingResource

El poder de los recursos de inactividad radica en no tener que esperar a que el procesamiento de algunas aplicaciones (redes, cálculos, animaciones, etc.) finalice con el modo de `sleep()` , lo que

trae la descamación y prolonga las pruebas. La documentación oficial se puede encontrar aquí .

Implementación

Hay tres cosas que debe hacer al implementar la interfaz `IdlingResource` :

- **`getName()`** : devuelve el nombre de su recurso inactivo.
- **`isIdleNow()`** : comprueba si su objeto xyz, operación, etc. está inactivo en este momento.
- **`registerIdleTransitionCallback`** (`IdlingResource.ResourceCallback` callback): proporciona una devolución de llamada que debe llamar cuando su objeto pase al estado inactivo.

Ahora debe crear su propia lógica y determinar cuándo su aplicación está inactiva y cuándo no, ya que esto depende de la aplicación. A continuación encontrará un ejemplo simple, solo para mostrar cómo funciona. Hay otros ejemplos en línea, pero la implementación de una aplicación específica lleva a implementaciones específicas de recursos inactivos.

NOTAS

- Ha habido algunos ejemplos de Google donde pusieron `IdlingResources` en el código de la aplicación. **No hagas esto.** Presumiblemente lo colocaron allí solo para mostrar cómo funcionan.
- ¡Mantener su código limpio y mantener un solo principio de responsabilidad depende de usted!

Ejemplo

Digamos que tiene una actividad que hace cosas extrañas y demora mucho tiempo en cargar el fragmento y, por lo tanto, hace que sus pruebas de Espresso fracasen al no poder encontrar recursos de su fragmento (debe cambiar cómo se crea su actividad y cuándo). para acelerarlo). Pero en cualquier caso, para mantenerlo simple, el siguiente ejemplo muestra cómo debería ser.

Nuestro ejemplo de recurso inactivo obtendría dos objetos:

- La **etiqueta** del fragmento que necesita encontrar y en espera de unirse a la actividad.
- Un objeto **FragmentManager** que se utiliza para encontrar el fragmento.

```
/**
 * FragmentIdlingResource - idling resource which waits while Fragment has not been loaded.
 */
public class FragmentIdlingResource implements IdlingResource {
    private final FragmentManager mFragmentManager;
    private final String mTag;
    //resource callback you use when your activity transitions to idle
    private volatile ResourceCallback resourceCallback;

    public FragmentIdlingResource(FragmentManager fragmentManager, String tag) {
        mFragmentManager = fragmentManager;
```

```
        mTag = tag;
    }

    @Override
    public String getName() {
        return FragmentIdlingResource.class.getName() + ":" + mTag;
    }

    @Override
    public boolean isIdleNow() {
        //simple check, if your fragment is added, then your app has became idle
        boolean idle = (mFragmentManager.findFragmentByTag(mTag) != null);
        if (idle) {
            //IMPORTANT: make sure you call onTransitionToIdle
            resourceCallback.onTransitionToIdle();
        }
        return idle;
    }

    @Override
    public void registerIdleTransitionCallback(ResourceCallback resourceCallback) {
        this.resourceCallback = resourceCallback;
    }
}
```

Ahora que tienes tu `IdlingResource` escrito, necesitas usarlo en algún lugar, ¿no?

Uso

Vamos a omitir la configuración completa de la clase de prueba y solo veremos cómo se vería un caso de prueba:

```
@Test
public void testSomeFragmentText() {
    mActivityTestRule.launchActivity(null);

    //creating the idling resource
    IdlingResource fragmentLoadedIdlingResource = new
FragmentIdlingResource(mActivityTestRule.getActivity().getSupportFragmentManager(),
SomeFragmentText.TAG);
    //registering the idling resource so espresso waits for it
    Espresso.registerIdlingResources(idlingResource1);
    onView(withId(R.id.txtHelloWorld)).check(matches(withText(helloWorldText)));

    //lets cleanup after ourselves
    Espresso.unregisterIdlingResources(fragmentLoadedIdlingResource);
}
```

Combinación con regla JUnit

Esto no es difícil; También puede aplicar el recurso inactivo en forma de una regla de prueba JUnit. Por ejemplo, digamos que tiene algún SDK que contiene Volley y quiere que Espresso lo espere. En lugar de pasar por cada caso de prueba o aplicarlo en la configuración, puede crear

una regla JUnit y simplemente escribir:

```
@Rule
public final SDKIdlingRule mSdkIdlingRule = new
SDKIdlingRule(SDKInstanceHolder.getInstance());
```

Ahora, ya que este es un ejemplo, no lo des por sentado; Todo el código aquí es imaginario y se usa solo para fines de demostración:

```
public class SDKIdlingRule implements TestRule {
    //idling resource you wrote to check is volley idle or not
    private VolleyIdlingResource mVolleyIdlingResource;
    //request queue that you need from volley to give it to idling resource
    private RequestQueue mRequestQueue;

    //when using the rule extract the request queue from your SDK
    public SDKIdlingRule(SDKClass sdkClass) {
        mRequestQueue = getVolleyRequestQueue(sdkClass);
    }

    private RequestQueue getVolleyRequestQueue(SDKClass sdkClass) {
        return sdkClass.getVolleyRequestQueue();
    }

    @Override
    public Statement apply(final Statement base, Description description) {
        return new Statement() {
            @Override
            public void evaluate() throws Throwable {
                //registering idling resource
                mVolleyIdlingResource = new VolleyIdlingResource(mRequestQueue);
                Espresso.registerIdlingResources(mVolleyIdlingResource);
                try {
                    base.evaluate();
                } finally {
                    if (mVolleyIdlingResource != null) {
                        //deregister the resource when test finishes
                        Espresso.unregisterIdlingResources(mVolleyIdlingResource);
                    }
                }
            }
        };
    }
}
```

Lea Escribir pruebas de interfaz de usuario - Android en línea:
https://riptutorial.com/es/android/topic/3530/escribir-pruebas-de-interfaz-de-usuario---android

Capítulo 99: Eventos / intenciones de botón de hardware (PTT, LWP, etc.)

Introducción

Varios dispositivos Android tienen botones personalizados añadidos por el fabricante. Esto abre nuevas posibilidades para que el desarrollador maneje esos botones, especialmente cuando hace aplicaciones dirigidas a dispositivos de hardware.

Este tema documenta los botones que tienen intenciones adjuntas, que puede escuchar a través de los receptores de intenciones.

Examples

Dispositivos Sonim

Los dispositivos de Sonim tienen diferentes modelos de diferentes botones personalizados:

PTT_KEY

```
com.sonim.intent.action.PTT_KEY_DOWN
com.sonim.intent.action.PTT_KEY_UP
```

YELLOW_KEY

```
com.sonim.intent.action.YELLOW_KEY_DOWN
com.sonim.intent.action.YELLOW_KEY_UP
```

SOS_KEY

```
com.sonim.intent.action.SOS_KEY_DOWN
com.sonim.intent.action.SOS_KEY_UP
```

GREEN_KEY

```
com.sonim.intent.action.GREEN_KEY_DOWN
com.sonim.intent.action.GREEN_KEY_UP
```

Registrando los botones

Para recibir esos intentos, deberá asignar los botones a su aplicación en la Configuración del teléfono. Sonim tiene la posibilidad de registrar automáticamente los botones en la aplicación cuando se instala. Para hacer eso, tendrá que contactarlos y obtener una clave específica del paquete para incluir en su Manifiesto de la siguiente manera:

```
<meta-data
    android:name="app_key_green_data"
    android:value="your-key-here" />
```

Dispositivos RugGear

Botón PTT

```
android.intent.action.PTT.down
android.intent.action.PTT.up
```

Confirmado en: RG730, RG740A

Lea Eventos / intenciones de botón de hardware (PTT, LWP, etc.) en línea: https://riptutorial.com/es/android/topic/10418/eventos---intenciones-de-boton-de-hardware--ptt--lwp--etc--

Capítulo 100: Eventos táctiles

Examples

Cómo variar entre los eventos táctiles de grupo de vista infantil y padre

1. `onTouchEvents()` para grupos de vistas anidadas se puede administrar mediante el `boolean` onInterceptTouchEvent .

El valor predeterminado para `OnInterceptTouchEvent` es falso.

`onTouchEvent` los `onTouchEvent` se recibe antes que el niño. Si `OnInterceptTouchEvent` devuelve false, envía el evento de movimiento a lo largo de la cadena al controlador `OnTouchEvent` del niño. Si retorna verdadero, los padres manejarán el evento táctil.

Sin embargo, puede haber casos en los que deseamos que algunos elementos secundarios gestionen los `OnTouchEvent` de `OnTouchEvent` y otros que sean gestionados por la vista principal (o posiblemente el principal del elemento primario).

Esto se puede gestionar de más de una manera.

2. Una forma en que un elemento secundario se puede proteger de `OnInterceptTouchEvent` del `OnInterceptTouchEvent` es mediante la implementación del requestDisallowInterceptTouchEvent .

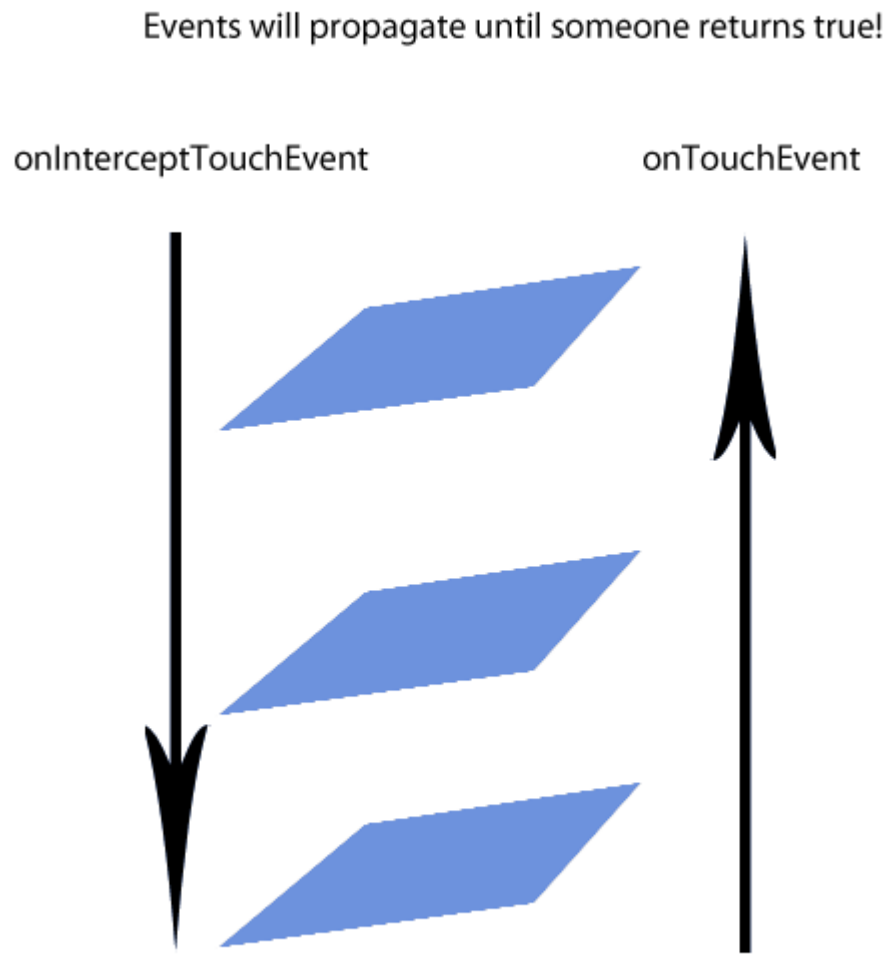
   public void requestDisallowInterceptTouchEvent (boolean disallowIntercept)

Esto evita que cualquiera de las vistas principales administre `OnTouchEvent` para este elemento, si el elemento tiene habilitados los controladores de eventos.

3.

Si `OnInterceptTouchEvent` es falso, se evaluará `OnTouchEvent` del elemento `OnTouchEvent` . Si tiene métodos dentro de los elementos secundarios que manejan los diversos eventos táctiles, cualquier controlador de eventos relacionado que esté deshabilitado devolverá el OnTouchEvent al padre.

Esta respuesta:
Una visualización de cómo la propagación de eventos táctiles pasa a través de:
`parent -> child|parent -> child|parent -> child views.`



Cortesía desde aquí

4. Otra forma es devolver valores variables de `OnInterceptTouchEvent` para el padre.

Este ejemplo, tomado de Managing Touch Events en un ViewGroup, demuestra cómo interceptar el `OnTouchEvent` del niño cuando el usuario se desplaza.

4a.

```
@Override
public boolean onInterceptTouchEvent(MotionEvent ev) {
    /*
     * This method JUST determines whether we want to intercept the motion.
     * If we return true, onTouchEvent will be called and we do the actual
     * scrolling there.
     */


    final int action = MotionEventCompat.getActionMasked(ev);

    // Always handle the case of the touch gesture being complete.
    if (action == MotionEvent.ACTION_CANCEL || action == MotionEvent.ACTION_UP) {
        // Release the scroll.
```

```
        mIsScrolling = false;
        return false; // Do not intercept touch event, let the child handle it
    }

    switch (action) {
        case MotionEvent.ACTION_MOVE: {
            if (mIsScrolling) {
                // We're currently scrolling, so yes, intercept the
                // touch event!
                return true;
            }

            // If the user has dragged her finger horizontally more than
            // the touch slop, start the scroll

            // left as an exercise for the reader
            final int xDiff = calculateDistanceX(ev);

            // Touch slop should be calculated using ViewConfiguration
            // constants.
            if (xDiff > mTouchSlop) {
                // Start scrolling!
                mIsScrolling = true;
                return true;
            }
            break;
        }
        ...
    }

    // In general, we don't want to intercept touch events. They should be
    // handled by the child view.
    return false;
}
```

Este es un código del mismo enlace que muestra cómo crear los parámetros del rectángulo alrededor de su elemento:

4b.

```
// The hit rectangle for the ImageButton
myButton.getHitRect(delegateArea);

// Extend the touch area of the ImageButton beyond its bounds
// on the right and bottom.
delegateArea.right += 100;
delegateArea.bottom += 100;

// Instantiate a TouchDelegate.
// "delegateArea" is the bounds in local coordinates of
// the containing view to be mapped to the delegate view.
// "myButton" is the child view that should receive motion
// events.
TouchDelegate touchDelegate = new TouchDelegate(delegateArea, myButton);

// Sets the TouchDelegate on the parent view, such that touches
// within the touch delegate bounds are routed to the child.
if (View.class.isInstance(myButton.getParent())) {
    ((View) myButton.getParent()).setTouchDelegate(touchDelegate);
```

```
}
```

Lea Eventos táctiles en línea: https://riptutorial.com/es/android/topic/7167/eventos-tactiles

Capítulo 101: Excepciones

Examples

NetworkOnMainThreadException

De la documentación :

> La excepción que se produce cuando una aplicación intenta realizar una operación de red en su hilo principal.
>
> Esto solo se lanza para aplicaciones dirigidas al Honeycomb SDK o superior. Las aplicaciones que apuntan a versiones anteriores del SDK pueden hacer redes en sus subprocesos de bucle de eventos principales, pero no se recomienda.

Aquí hay un ejemplo de un fragmento de código que puede causar esa excepción:

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Uri.Builder builder = new Uri.Builder().scheme("http").authority("www.google.com");
        HttpURLConnection urlConnection = null;
        BufferedReader reader = null;
        URL url;
        try {
            url = new URL(builder.build().toString());
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setRequestMethod("GET");
            urlConnection.connect();
        } catch (IOException e) {
            Log.e("TAG","Connection error", e);
        } finally{
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
            if (reader != null) {
                try {
                    reader.close();
                } catch (final IOException e) {
                    Log.e("TAG", "Error closing stream", e);
                }
            }
        }
    }
}
```

El código anterior lanzará la `NetworkOnMainThreadException` para aplicaciones dirigidas a Honeycomb SDK (Android v3.0) o superior, ya que la aplicación está intentando realizar una operación de red en el hilo principal.

Para evitar esta excepción, sus operaciones de red siempre deben ejecutarse en una tarea en segundo plano a través de una `AsyncTask` , `Thread` , `IntentService` , etc.

```
private class MyAsyncTask extends AsyncTask<String, Integer, Void> {

    @Override
    protected Void doInBackground(String[] params) {
        Uri.Builder builder = new Uri.Builder().scheme("http").authority("www.google.com");
        HttpURLConnection urlConnection = null;
        BufferedReader reader = null;
        URL url;
        try {
            url = new URL(builder.build().toString());
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setRequestMethod("GET");
            urlConnection.connect();
        } catch (IOException e) {
            Log.e("TAG","Connection error", e);
        } finally{
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
            if (reader != null) {
                try {
                    reader.close();
                } catch (final IOException e) {
                    Log.e("TAG", "Error closing stream", e);
                }
            }
        }

        return null;
    }
}
```

ActivityNotFoundException

Esta es una `Exception` muy común. Hace que la aplicación se detenga durante el inicio o la ejecución de la aplicación. En el `LogCat` ves el mensaje:

```
android.content.ActivityNotFoundException : Unable to find explicit activity class;
have you declared this activity in your AndroidManifest.xml?
```

En este caso, verifique si ha declarado su actividad en el archivo `AndroidManifest.xml` .

La forma más sencilla de declarar su `Activity` en `AndroidManifest.xml` es:

```
<activity  android:name="com.yourdomain.YourStoppedActivity" />
```

Error de memoria insuficiente

Este es un error de tiempo de ejecución que ocurre cuando solicita una gran cantidad de memoria en el montón. Esto es común cuando se carga un mapa de bits en un ImageView.

Tienes algunas opciones:

1. Utilice un montón de aplicaciones grandes

Agregue la opción "largeHeap" a la etiqueta de la aplicación en su AndroidManifest.xml. Esto hará que haya más memoria disponible para su aplicación, pero es probable que no solucione el problema de raíz.

```
<application largeHeap="true" ... >
```

2. Recicla tus bitmaps

Después de cargar un mapa de bits, asegúrese de reciclarlo y liberar memoria:

```
    if (bitmap != null && !bitmap.isRecycled())
       bitmap.recycle();
```

3. Cargar bitmaps muestreados en memoria

Evite cargar todo el mapa de bits en la memoria al mismo tiempo muestreando un tamaño reducido, utilizando BitmapOptions e inSampleSize.

Ver la documentación de Android por ejemplo

DexException

```
com.android.dex.DexException: Multiple dex files define Lcom/example/lib/Class;
```

Este error se produce porque la aplicación, al empaquetar, encuentra dos archivos `.dex` que definen el mismo conjunto de métodos.

Por lo general, esto sucede porque la aplicación ha adquirido accidentalmente 2 dependencias separadas en la misma biblioteca.

Por ejemplo, supongamos que tiene un proyecto y desea confiar en dos bibliotecas: `A` y `B` , cada una con sus propias dependencias. Si la biblioteca `B` ya tiene una dependencia de la biblioteca `A` , se lanzará este error si la biblioteca `A` se agrega al proyecto por sí misma. La compilación de la biblioteca `B` ya proporcionó el conjunto de códigos de `A` , de modo que cuando el compilador va a la biblioteca de paquetes `A` , encuentra los métodos de la biblioteca `A` ya empaquetados.

Para resolverlo, asegúrese de que ninguna de sus dependencias se pueda agregar accidentalmente dos veces de esa manera

UncaughtException

Si desea manejar excepciones no detectadas, intente capturarlas todas en el método onCreate de su clase de aplicación:

```
public class MyApp extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        try {
            Thread
                .setDefaultUncaughtExceptionHandler(
                            new Thread.UncaughtExceptionHandler() {

                @Override
                public void uncaughtException(Thread thread, Throwable e) {
                    Log.e(TAG,
                            "Uncaught Exception thread: "+thread.getName()+"
                             "+e.getStackTrace());
                    handleUncaughtException (thread, e);
                }
            });
        } catch (SecurityException e) {
            Log.e(TAG,
                    "Could not set the Default Uncaught Exception Handler:"
                    +e.getStackTrace());
        }
    }

    private void handleUncaughtException (Thread thread, Throwable e){
        Log.e(TAG, "uncaughtException:");
        e.printStackTrace();
    }
}
```

Registro de manejador propio para excepciones inesperadas.

Así es como puede reaccionar a las excepciones que no se han detectado, de forma similar al estándar del sistema "La aplicación XYZ se ha bloqueado"

```
import android.app.Application;
import android.util.Log;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

/**
 * Application class writing unexpected exceptions to a crash file before crashing.
 */
public class MyApplication extends Application {
    private static final String TAG = "ExceptionHandler";

    @Override
    public void onCreate() {
        super.onCreate();

        // Setup handler for uncaught exceptions.
        final Thread.UncaughtExceptionHandler defaultHandler =
Thread.getDefaultUncaughtExceptionHandler();
```

```
        Thread.setDefaultUncaughtExceptionHandler(new Thread.UncaughtExceptionHandler() {
            @Override
            public void uncaughtException(Thread thread, Throwable e) {
                try {
                    handleUncaughtException(e);
                    System.exit(1);
                } catch (Throwable e2) {
                    Log.e(TAG, "Exception in custom exception handler", e2);
                    defaultHandler.uncaughtException(thread, e);
                }
            }
        });
    }

    private void handleUncaughtException(Throwable e) throws IOException {
        Log.e(TAG, "Uncaught exception logged to local file", e);

        // Create a new unique file
        final DateFormat dateFormat =  new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss", Locale.US);
        String timestamp;
        File file = null;
        while (file == null || file.exists()) {
            timestamp = dateFormat.format(new Date());
            file = new File(getFilesDir(), "crashLog_" + timestamp + ".txt");
        }
        Log.i(TAG, "Trying to create log file " + file.getPath());
        file.createNewFile();

        // Write the stacktrace to the file
        FileWriter writer = null;
        try {
            writer = new FileWriter(file, true);
            for (StackTraceElement element : e.getStackTrace()) {
                writer.write(element.toString());
            }
        } finally {
            if (writer != null) writer.close();
        }

        // You can (and probably should) also display a dialog to notify the user
    }
}
```

Luego registre esta clase de aplicación en su AndroidManifest.xml:

```
<application android:name="de.ioxp.arkmobile.MyApplication" >
```

Lea Excepciones en línea: https://riptutorial.com/es/android/topic/112/excepciones

Capítulo 102: ExoPlayer

Examples

Agrega ExoPlayer al proyecto

A través de jCenter

Incluyendo lo siguiente en el archivo build.gradle de su proyecto:

```
compile 'com.google.android.exoplayer:exoplayer:rX.X.X'
```

donde rX.XX es la versión preferida. Para la última versión, ver los lanzamientos del proyecto. Para más detalles, vea el proyecto en Bintray .

Utilizando ExoPlayer

Crea una instancia de tu ExoPlayer:

```
exoPlayer = ExoPlayer.Factory.newInstance(RENDERER_COUNT, minBufferMs, minRebufferMs);
```

Para reproducir audio solo puedes usar estos valores:

```
RENDERER_COUNT = 1 //since you want to render simple audio
minBufferMs = 1000
minRebufferMs = 5000
```

Ambos valores de búfer pueden ser ajustados de acuerdo a sus requerimientos.

Ahora tienes que crear un DataSource. Cuando quiera transmitir mp3, puede usar DefaultUriDataSource. Tienes que pasar el contexto y un UserAgent. Para mantenerlo simple, reproduzca un archivo local y pase nulo como userAgent:

```
DataSource dataSource = new DefaultUriDataSource(context, null);
```

Luego crea el código fuente:

```
ExtractorSampleSource sampleSource = new ExtractorSampleSource(
                uri, dataSource, new Mp3Extractor(), RENDERER_COUNT, requestedBufferSize);
```

uri apunta a su archivo, como un Extractor puede usar un simple Mp3Extractor predeterminado si desea reproducir mp3. requiredBufferSize se puede modificar de nuevo según sus requisitos. Use 5000 por ejemplo.

Ahora puede crear su procesador de pistas de audio utilizando la fuente de muestra de la siguiente manera:

```
MediaCodecAudioTrackRenderer audioRenderer = new MediaCodecAudioTrackRenderer(sampleSource);
```

Finalmente llame a preparar en su instancia de exoPlayer:

```
exoPlayer.prepare(audioRenderer);
```

Para iniciar la reproducción de la llamada:

```
exoPlayer.setPlayWhenReady(true);
```

Pasos principales para reproducir video y audio usando las implementaciones estándar de TrackRenderer

```
// 1. Instantiate the player.
player = ExoPlayer.Factory.newInstance(RENDERER_COUNT);
// 2. Construct renderers.
MediaCodecVideoTrackRenderer videoRenderer = ...
MediaCodecAudioTrackRenderer audioRenderer = ...
// 3. Inject the renderers through prepare.
player.prepare(videoRenderer, audioRenderer);
// 4. Pass the surface to the video renderer.
player.sendMessage(videoRenderer, MediaCodecVideoTrackRenderer.MSG_SET_SURFACE,     surface);
// 5. Start playback.
player.setPlayWhenReady(true);
...
player.release(); // Don’t forget to release when done!
```

Lea ExoPlayer en línea: https://riptutorial.com/es/android/topic/6248/exoplayer

Capítulo 103: Facebook SDK para Android

Sintaxis

- **newInstance** : para crear una instancia única de la clase de ayuda de Facebook.
- **loginUser** : Para iniciar sesión usuario.
- **SignOut** : Para cerrar la sesión del usuario.
- **getCallbackManager** : para obtener devolución de llamada para Facebook.
- **getLoginCallback** : para obtener devolución de llamada para inicio de sesión.
- **getKeyHash** : Para generar Hash clave de Facebook.

Parámetros

Parámetro	Detalles
ETIQUETA	Una cadena utilizada durante el registro
FacebookSignInHelper	Una referencia estática a facebook helper
CallbackManager	Una devolución de llamada para las operaciones de facebook
Actividad	Un contexto
PERMISO_LOGIN	Una matriz que contiene todos los permisos requeridos de facebook para iniciar sesión.
loginCallback	Una devolución de llamada para el inicio de sesión de facebook

Examples

Cómo agregar Facebook Login en Android

Agregue debajo las dependencias a su `build.gradle`

```
  // Facebook login
    compile 'com.facebook.android:facebook-android-sdk:4.21.1'
```

Agregue la siguiente clase de ayuda a su paquete de utilidades:

```
/**
 * Created by Andy
 * An utility for Facebook
 */
public class FacebookSignInHelper {
    private static final String TAG = FacebookSignInHelper.class.getSimpleName();
    private static FacebookSignInHelper facebookSignInHelper = null;
```

```
    private CallbackManager callbackManager;
    private Activity mActivity;
    private static final Collection<String> PERMISSION_LOGIN = (Collection<String>)
Arrays.asList("public_profile", "user_friends","email");
    private FacebookCallback<LoginResult> loginCallback;



    public static FacebookSignInHelper newInstance(Activity context) {
        if (facebookSignInHelper == null)
            facebookSignInHelper = new FacebookSignInHelper(context);
        return facebookSignInHelper;
    }


    public FacebookSignInHelper(Activity mActivity) {
        try {
            this.mActivity = mActivity;
            // Initialize the SDK before executing any other operations,
            // especially, if you're using Facebook UI elements.
            FacebookSdk.sdkInitialize(this.mActivity);
            callbackManager = CallbackManager.Factory.create();
            loginCallback = new FacebookCallback<LoginResult>() {
                @Override
                public void onSuccess(LoginResult loginResult) {
                   // You are logged into Facebook
                }

                @Override
                public void onCancel() {
                    Log.d(TAG, "Facebook: Cancelled by user");
                }

                @Override
                public void onError(FacebookException error) {
                    Log.d(TAG, "FacebookException: " + error.getMessage());
                }
            };
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * To login user on facebook without default Facebook button
     */
    public void loginUser() {
        try {
            LoginManager.getInstance().registerCallback(callbackManager, loginCallback);
            LoginManager.getInstance().logInWithReadPermissions(this.mActivity,
PERMISSION_LOGIN);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    /**
     * To log out user from facebook
     */
    public void signOut() {
```

```
        // Facebook sign out
        LoginManager.getInstance().logOut();
    }

    public CallbackManager getCallbackManager() {
        return callbackManager;
    }

    public FacebookCallback<LoginResult> getLoginCallback() {
        return loginCallback;
    }

    /**
     * Attempts to log debug key hash for facebook
     *
     * @param context : A reference to context
     * @return : A facebook debug key hash
     */
    public static String getKeyHash(Context context) {
        String keyHash = null;
        try {
            PackageInfo info = context.getPackageManager().getPackageInfo(
                    context.getPackageName(),
                    PackageManager.GET_SIGNATURES);
            for (Signature signature : info.signatures) {
                MessageDigest md = MessageDigest.getInstance("SHA");
                md.update(signature.toByteArray());
                keyHash = Base64.encodeToString(md.digest(), Base64.DEFAULT);
                Log.d(TAG, "KeyHash:" + keyHash);
            }
        } catch (PackageManager.NameNotFoundException e) {
            e.printStackTrace();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return keyHash;
    }
}
```

Agregue el siguiente código en su actividad:

```
FacebookSignInHelper facebookSignInHelper =
FacebookSignInHelper.newInstance(LoginActivity.this, fireBaseAuthHelper);
facebookSignInHelper.loginUser();
```

Agregue el siguiente código a su `OnActivityResult` :

```
 facebookSignInHelper.getCallbackManager().onActivityResult(requestCode, resultCode, data);
```

Configuración de permisos para acceder a los datos desde el perfil de Facebook

Si desea recuperar los detalles del perfil de Facebook de un usuario, necesita establecer permisos para el mismo:

```
loginButton = (LoginButton)findViewById(R.id.login_button);

loginButton.setReadPermissions(Arrays.asList("email", "user_about_me"));
```

Puede seguir agregando más permisos, como listas de amigos, publicaciones, fotos, etc. Simplemente elija el permiso correcto y agréguelo a la lista anterior.

Nota: no es necesario establecer ningún permiso explícito para acceder al perfil público (nombre, apellido, ID, género, etc.).

Crea tu propio botón personalizado para iniciar sesión en Facebook

Una vez que agregas el inicio de sesión / registro en Facebook, el botón se ve algo así como:



La mayoría de las veces, no coincide con las especificaciones de diseño de su aplicación. Y así es como puedes personalizarlo:

```
<FrameLayout
    android:layout_below="@+id/no_network_bar"
    android:id="@+id/FrameLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <com.facebook.login.widget.LoginButton
        android:id="@+id/login_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone" />

    <Button
        android:background="#3B5998"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:id="@+id/fb"
        android:onClick="onClickFacebookButton"
        android:textAllCaps="false"
        android:text="Sign up with Facebook"
        android:textSize="22sp"
        android:textColor="#ffffff" />
</FrameLayout>
```

Simplemente envuelva el `com.facebook.login.widget.LoginButton` original en un `FrameLayout` y haga que su visibilidad desaparezca.

A continuación, agregue su botón personalizado en el mismo `FrameLayout` . He añadido algunas especificaciones de muestra. Siempre puedes crear tu propio fondo dibujable para el botón de Facebook y establecerlo como fondo del botón.

Lo último que hacemos es simplemente convertir el clic en mi botón personalizado en un clic en el

botón de Facebook:

```
//The original Facebook button
LoginButton loginButton = (LoginButton)findViewById(R.id.login_button);

//Our custom Facebook button
fb = (Button) findViewById(R.id.fb);

public void onClickFacebookButton(View view) {
    if (view == fb) {
        loginButton.performClick();
    }
}
```

¡Genial! Ahora el botón se ve algo así:



Una guía minimalista para la implementación de inicio de sesión / registro en Facebook.

1. Tienes que configurar los requisitos previos .
2. Agrega la actividad de Facebook al archivo *AndroidManifest.xml* :

   ```
   <activity
       android:name="com.facebook.FacebookActivity"
       android:configChanges= "keyboard|keyboardHidden|screenLayout|screenSize|orientation"
       android:theme="@android:style/Theme.Translucent.NoTitleBar"
       android:label="@string/app_name" />
   ```

3. Agregue el botón de inicio de sesión a su archivo XML de diseño:

   ```
   <com.facebook.login.widget.LoginButton
       android:id="@+id/login_button"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content" />
   ```

4. Ahora tienes el botón de Facebook. Si el usuario hace clic en él, el cuadro de diálogo de inicio de sesión de Facebook aparecerá en la parte superior de la pantalla de la aplicación. Aquí el usuario puede completar sus credenciales y presionar el botón *Iniciar sesión* . Si las credenciales son correctas, el cuadro de diálogo concede los permisos correspondientes y se envía una devolución de llamada a su actividad original que contiene el botón. El siguiente código muestra cómo puede recibir esa devolución de llamada:

   ```
   loginButton.registerCallback(callbackManager, new FacebookCallback<LoginResult>() {
       @Override
       public void onSuccess(LoginResult loginResult) {
           // Completed without error. You might want to use the retrieved data here.
       }
   ```

```
    @Override
    public void onCancel() {
        // The user either cancelled the Facebook login process or didn't authorize the
app.
    }

    @Override
    public void onError(FacebookException exception) {
        // The dialog was closed with an error. The exception will help you recognize
what exactly went wrong.
    }
});
```

Cerrar sesión de Facebook

Facebook SDK 4.0 en adelante, así es como cerramos la sesión:

```
com.facebook.login.LoginManager.getInstance().logOut();
```

Para las versiones anteriores a 4.0, el cierre de sesión se borra explícitamente el token de acceso:

```
Session session = Session.getActiveSession();
session.closeAndClearTokenInformation();
```

Lea Facebook SDK para Android en línea: https://riptutorial.com/es/android/topic/3919/facebook-sdk-para-android

Capítulo 104: Facturación en la aplicación

Examples

Compras consumibles en la aplicación

Los productos administrados consumibles son productos que se pueden comprar varias veces, como la moneda del juego, la vida del juego, los power-ups, etc.

En este ejemplo, vamos a implementar 4 productos diferentes consumibles **administrados** `"item1", "item2", "item3", "item4"` .

Pasos en resumen:

1. Agregue la biblioteca de facturación integrada en la aplicación a su proyecto (archivo AIDL).
2. Agregue el permiso requerido en el archivo `AndroidManifest.xml` .
3. Implementar un apk firmado a la consola de desarrolladores de Google.
4. Define tus productos.
5. Implementar el código.
6. Prueba de facturación en la aplicación (opcional).

Paso 1:

En primer lugar, deberemos agregar el archivo AIDL a su proyecto como se explica claramente en la documentación de Google aquí .

`IInAppBillingService.aidl` es un archivo de Lenguaje de definición de interfaz de Android (AIDL) que define la interfaz para el servicio de versión 3 de facturación integrada en la aplicación. Utilizará esta interfaz para realizar solicitudes de facturación invocando llamadas a métodos de IPC.

Paso 2:

Después de agregar el archivo AIDL, agregue el permiso BILLING en `AndroidManifest.xml` :

```
<!-- Required permission for implementing In-app Billing -->
<uses-permission android:name="com.android.vending.BILLING" />
```

Paso 3:

Genere un apk firmado y cárguelo en la Consola de desarrolladores de Google. Esto es necesario para que podamos comenzar a definir nuestros productos en la aplicación allí.

Etapa 4:

Defina todos sus productos con diferentes ID de producto y establezca un precio para cada uno de ellos. Existen 2 tipos de productos (productos gestionados y suscripciones). Como ya dijimos, vamos a implementar 4 productos diferentes consumibles **administrados** `"item1", "item2", "item3", "item4"`.

Paso 5:

Después de realizar todos los pasos anteriores, ahora está listo para comenzar a implementar el código en su propia actividad.

Actividad principal:

```
public class MainActivity extends Activity {

    IInAppBillingService inAppBillingService;
    ServiceConnection serviceConnection;

    // productID for each item. You should define them in the Google Developers Console.
    final String item1 = "item1";
    final String item2 = "item2";
    final String item3 = "item3";
    final String item4 = "item4";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Instantiate the views according to your layout file.
        final Button buy1 = (Button) findViewById(R.id.buy1);
        final Button buy2 = (Button) findViewById(R.id.buy2);
        final Button buy3 = (Button) findViewById(R.id.buy3);
        final Button buy4 = (Button) findViewById(R.id.buy4);

        // setOnClickListener() for each button.
        // buyItem() here is the method that we will implement to launch the PurchaseFlow.
        buy1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                buyItem(item1);
            }
        });

        buy2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                buyItem(item2);
            }
```

```
        });

        buy3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                buyItem(item3);
            }
        });

        buy4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                buyItem(item4);
            }
        });

        // Attach the service connection.
        serviceConnection = new ServiceConnection() {
            @Override
            public void onServiceDisconnected(ComponentName name) {
                inAppBillingService = null;
            }

            @Override
            public void onServiceConnected(ComponentName name, IBinder service) {
                inAppBillingService = IInAppBillingService.Stub.asInterface(service);
            }
        };

        // Bind the service.
        Intent serviceIntent = new
Intent("com.android.vending.billing.InAppBillingService.BIND");
        serviceIntent.setPackage("com.android.vending");
        bindService(serviceIntent, serviceConnection, BIND_AUTO_CREATE);

        // Get the price of each product, and set the price as text to
        // each button so that the user knows the price of each item.
        if (inAppBillingService != null) {
            // Attention: You need to create a new thread here because
            // getSkuDetails() triggers a network request, which can
            // cause lag to your app if it was called from the main thread.
            Thread thread = new Thread(new Runnable() {
                @Override
                public void run() {
                    ArrayList<String> skuList = new ArrayList<>();
                    skuList.add(item1);
                    skuList.add(item2);
                    skuList.add(item3);
                    skuList.add(item4);
                    Bundle querySkus = new Bundle();
                    querySkus.putStringArrayList("ITEM_ID_LIST", skuList);

                    try {
                        Bundle skuDetails = inAppBillingService.getSkuDetails(3,
getPackageName(), "inapp", querySkus);
                        int response = skuDetails.getInt("RESPONSE_CODE");

                        if (response == 0) {
                            ArrayList<String> responseList =
skuDetails.getStringArrayList("DETAILS_LIST");
```

```
                            for (String thisResponse : responseList) {
                                JSONObject object = new JSONObject(thisResponse);
                                String sku = object.getString("productId");
                                String price = object.getString("price");

                                switch (sku) {
                                    case item1:
                                        buy1.setText(price);
                                        break;
                                    case item2:
                                        buy2.setText(price);
                                        break;
                                    case item3:
                                        buy3.setText(price);
                                        break;
                                    case item4:
                                        buy4.setText(price);
                                        break;
                                }
                            }
                        }
                    } catch (RemoteException | JSONException e) {
                        e.printStackTrace();
                    }
                }
            });
            thread.start();
        }
    }

    // Launch the PurchaseFlow passing the productID of the item the user wants to buy as a
parameter.
    private void buyItem(String productID) {
        if (inAppBillingService != null) {
            try {
                Bundle buyIntentBundle = inAppBillingService.getBuyIntent(3, getPackageName(),
productID, "inapp", "bGoa+V7g/yqDXvKRqq+JTFn4uQZbPiQJo4pf9RzJ");
                PendingIntent pendingIntent = buyIntentBundle.getParcelable("BUY_INTENT");
                startIntentSenderForResult(pendingIntent.getIntentSender(), 1003, new
Intent(), 0, 0, 0);
            } catch (RemoteException | IntentSender.SendIntentException e) {
                e.printStackTrace();
            }
        }
    }

    // Unbind the service in onDestroy(). If you don't unbind, the open
    // service connection could cause your device's performance to degrade.
    @Override
    public void onDestroy() {
        super.onDestroy();
        if (inAppBillingService != null) {
            unbindService(serviceConnection);
        }
    }

    // Check here if the in-app purchase was successful or not. If it was successful,
    // then consume the product, and let the app make the required changes.
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
```

```
        if (requestCode == 1003 && resultCode == RESULT_OK) {

            final String purchaseData = data.getStringExtra("INAPP_PURCHASE_DATA");

            // Attention: You need to create a new thread here because
            // consumePurchase() triggers a network request, which can
            // cause lag to your app if it was called from the main thread.
            Thread thread = new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        JSONObject jo = new JSONObject(purchaseData);
                        // Get the productID of the purchased item.
                        String sku = jo.getString("productId");
                        String productName = null;

                        // increaseCoins() here is a method used as an example in a game to
                        // increase the in-game currency if the purchase was successful.
                        // You should implement your own code here, and let the app apply
                        // the required changes after the purchase was successful.
                        switch (sku) {
                            case item1:
                                productName = "Item 1";
                                increaseCoins(2000);
                                break;
                            case item2:
                                productName = "Item 2";
                                increaseCoins(8000);
                                break;
                            case item3:
                                productName = "Item 3";
                                increaseCoins(18000);
                                break;
                            case item4:
                                productName = "Item 4";
                                increaseCoins(30000);
                                break;
                        }

                        // Consume the purchase so that the user is able to purchase the same
product again.
                        inAppBillingService.consumePurchase(3, getPackageName(),
jo.getString("purchaseToken"));
                        Toast.makeText(MainActivity.this, productName + " is successfully
purchased. Excellent choice, master!", Toast.LENGTH_LONG).show();
                    } catch (JSONException | RemoteException e) {
                        Toast.makeText(MainActivity.this, "Failed to parse purchase data.",
Toast.LENGTH_LONG).show();
                        e.printStackTrace();
                    }
                }
            });
            thread.start();
        }
    }
}
```

Paso 6:

Después de implementar el código, puede probarlo implementando su apk en el canal beta / alfa, y permitir que otros usuarios prueben el código por usted. Sin embargo, no se pueden realizar compras reales en la aplicación mientras se está en modo de prueba. Debes publicar tu aplicación / juego primero en Play Store para que todos los productos estén completamente activados.

Más información sobre las pruebas de facturación en la aplicación se puede encontrar aquí .

(Tercero) In-App v3 Library

Paso 1: En primer lugar, siga estos dos pasos para agregar la funcionalidad de la aplicación:

1. Agregue la biblioteca usando:

```
repositories {
           mavenCentral()
       }
       dependencies {
          compile 'com.anjlab.android.iab.v3:library:1.0.+'
       }
```

2. Agregar permiso en el archivo de manifiesto.

```
<uses-permission android:name="com.android.vending.BILLING" />
```

Paso 2: Inicialice su procesador de facturación:

```
BillingProcessor bp = new BillingProcessor(this, "YOUR LICENSE KEY FROM GOOGLE PLAY CONSOLE
HERE", this);
```

e implemente Billing Handler: BillingProcessor.IBillingHandler que contiene 4 métodos: a. onBillingInitialized (); segundo. onProductPurchased (String productId, detalles de TransactionDetails): aquí es donde debe manejar las acciones que se realizarán después de la compra exitosa c. onBillingError (int errorCode, Throwable error): maneja cualquier error ocurrido durante el proceso de compra d. onPurchaseHistoryRestored (): para restaurar en compras de aplicaciones

Paso 3: Cómo comprar un producto.

Para comprar un producto gestionado:

```
bp.purchase(YOUR_ACTIVITY, "YOUR PRODUCT ID FROM GOOGLE PLAY CONSOLE HERE");
```

Y para comprar una suscripción:

```
bp.subscribe(YOUR_ACTIVITY, "YOUR SUBSCRIPTION ID FROM GOOGLE PLAY CONSOLE HERE");
```

Paso 4: Consumir un producto.

Para consumir un producto simplemente llame al método consumPurchase.

bp.consumePurchase ("SU ID DE PRODUCTO DE LA CONSOLA DE GOOGLE PLAY AQUÍ");

Para otros métodos relacionados con la visita a la aplicación github

Lea Facturación en la aplicación en línea: https://riptutorial.com/es/android/topic/2843/facturacion-en-la-aplicacion

Capítulo 105: Fastjson

Introducción

Fastjson es una biblioteca de Java que se puede usar para convertir objetos Java en su representación JSON. También se puede utilizar para convertir una cadena JSON en un objeto Java equivalente.

Características de Fastjson:

Proporcionar el mejor rendimiento en el lado del servidor y el cliente de Android

Proporcione `toJSONString()` simples `toJSONString()` y `parseObject()` para convertir objetos Java a JSON y viceversa

Permitir que los objetos no modificables preexistentes se conviertan ay desde JSON

Amplio soporte de Java Generics

Sintaxis

- Objeto parse (texto de cadena)
- JSONObject parseObject (texto de cadena)
- T parseObject (String text, Class <T> clazz)
- JSONArray parseArray (texto de cadena)
- <T> List <T> parseArray (texto de cadena, clase <T> garabato)
- String toJSONString (objeto objeto)
- String toJSONString (objeto Object, boolean prettyFormat)
- Object toJSON (Object javaObject)

Examples

Analizando JSON con Fastjson

Puedes ver el ejemplo en la biblioteca de Fastjson.

Codificar

```
import com.alibaba.fastjson.JSON;

Group group = new Group();
group.setId(0L);
group.setName("admin");

User guestUser = new User();
guestUser.setId(2L);
guestUser.setName("guest");
```

```
User rootUser = new User();
rootUser.setId(3L);
rootUser.setName("root");

group.addUser(guestUser);
group.addUser(rootUser);

String jsonString = JSON.toJSONString(group);

System.out.println(jsonString);
```

Salida

```
{"id":0,"name":"admin","users":[{"id":2,"name":"guest"},{"id":3,"name":"root"}]}
```

Descodificar

```
String jsonString = ...;
Group group = JSON.parseObject(jsonString, Group.class);
```

Grupo.java

```
public class Group {

    private Long id;
    private String name;
    private List<User> users = new ArrayList<User>();

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public List<User> getUsers() {
        return users;
    }

    public void setUsers(List<User> users) {
        this.users = users;
    }

    public void addUser(User user) {
        users.add(user);
    }
}
```

Usuario.java

```
public class User {

    private Long   id;
    private String name;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Convertir los datos de tipo Map to JSON String

Código

```
    Group group = new Group();
    group.setId(1);
    group.setName("Ke");

    User user1 = new User();
    user1.setId(2);
    user1.setName("Liu");

    User user2 = new User();
    user2.setId(3);
    user2.setName("Yue");
    group.getList().add(user1);
    group.getList().add(user2);

    Map<Integer, Object> map = new HashMap<Integer,Object>();
    map.put(1, "No.1");
    map.put(2, "No.2");
    map.put(3, group.getList());

    String jsonString = JSON.toJSONString(map);
    System.out.println(jsonString);
```

Salida

```
{1:"No.1",2:"No.2",3:[{"id":2,"name":"Liu"},{"id":3,"name":"Yue"}]}
```

Lea Fastjson en línea: https://riptutorial.com/es/android/topic/10865/fastjson

Capítulo 106: Fecha / Hora localizada en Android

Observaciones

Se recomienda utilizar los métodos de la clase DateUtils para formatear fechas que tengan en cuenta la configuración regional, es decir, que tengan en cuenta las preferencias del usuario (por ejemplo, formatos de hora de reloj de 12h / 24h). Estos métodos son los más apropiados para las fechas que se muestran al usuario.

Para representaciones de fecha totalmente personalizadas, se recomienda usar la clase SimpleDateFormat , ya que permite controlar completamente todos los elementos de fecha.

Examples

Formato de fecha personalizado localizado con DateUtils.formatDateTime ()

DateUtils.formatDateTime () le permite proporcionar una hora y, en función de los indicadores que proporcione, crea una cadena de fecha y hora localizada. Las marcas le permiten especificar si incluir elementos específicos (como el día de la semana).

```
Date date = new Date();
String localizedDate = DateUtils.formatDateTime(context, date.getTime(),
DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_WEEKDAY);
```

formatDateTime () cuida automáticamente los formatos de fecha apropiados.

Formato de fecha / hora estándar en Android

Formato de una fecha:

```
Date date = new Date();
DateFormat df = DateFormat.getDateInstance(DateFormat.MEDIUM);
String localizedDate = df.format(date)
```

Formato de fecha y hora. La fecha es en formato corto, la hora es en formato largo:

```
Date date = new Date();
DateFormat df = DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.LONG);
String localizedDate = df.format(date)
```

Fecha / hora totalmente personalizada

```
Date date = new Date();
df = new SimpleDateFormat("HH:mm", Locale.US);
```

```
String localizedDate = df.format(date)
```

Patrones de uso común:

- HH: hora (0-23)
- hh: hora (1-12)
- a: marcador AM / PM
- mm: minuto (0-59)
- ss: segundo
- dd: día en mes (1-31)
- Mm: mes
- yyyy: año

Lea Fecha / Hora localizada en Android en línea:
https://riptutorial.com/es/android/topic/6057/fecha---hora-localizada-en-android

Capítulo 107: FileIO con Android

Introducción

Leer y escribir archivos en Android no es diferente de leer y escribir archivos en Java estándar. Se puede usar el mismo paquete `java.io` Sin embargo, hay algunos aspectos específicos relacionados con las carpetas donde se le permite escribir, los permisos en general y las soluciones MTP.

Observaciones

Android proporciona medios para compartir el archivo entre varias aplicaciones como se documenta aquí . Esto no es necesario si solo hay una aplicación que crea y utiliza el archivo.

Android ofrece opciones de almacenamiento alternativas, como preferencias compartidas y privadas, paquetes guardados y base de datos incorporada. En algunos casos, son una mejor opción que usar archivos simples.

La actividad de Android tiene pocos métodos específicos que se parecen a los reemplazos de los métodos estándar de archivo IO de Java. Por ejemplo, en lugar de `File.delete()` puede llamar a `Context.deleteFile()` , y en lugar de aplicar `File.listFiles()` recursiva, puede llamar a `Context.fileList()` para obtener la lista de todos los archivos específicos de su aplicación con algo menos código. Sin embargo, no proporcionan funcionalidad adicional más allá del paquete `java.io` estándar.

Examples

Obtención de la carpeta de trabajo.

Puede obtener su carpeta de trabajo llamando al método `getFilesDir()` en su Actividad (Actividad es la clase central en su aplicación que se hereda de Context. Consulte aquí). La lectura no es diferente. Solo su aplicación tendrá acceso a esta carpeta.

Su actividad podría contener el siguiente código, por ejemplo:

```
File myFolder = getFilesDir();
File myFile = new File(myFolder, "myData.bin");
```

Escritura de matriz en bruto de bytes

```
File myFile = new File(getFilesDir(), "myData.bin");
FileOutputStream out = new FileOutputStream(myFile);

// Write four bytes one two three four:
out.write(new byte [] { 1, 2, 3, 4}
```

```
out.close()
```

No hay nada específico de Android con este código. Si escribe muchos valores pequeños a menudo, use BufferedOutputStream para reducir el desgaste del SSD interno del dispositivo.

Serialización del objeto.

La vieja y buena serialización de objetos Java está disponible para usted en Android. Puedes definir clases serializables como:

```
class Cirle implements Serializable {
  final int radius;
  final String name;

  Circle(int radius, int name) {
    this.radius = radius;
    this.name = name;
  }
}
```

y luego escriba luego en ObjectOutputStream:

```
File myFile = new File(getFilesDir(), "myObjects.bin");
FileOutputStream out = new FileOutputStream(myFile);
ObjectOutputStream oout = new ObjectOutputStream(new BufferedOutputStream(out));

oout.writeObject(new Circle(10, "One"));
oout.writeObject(new Circle(12, "Two"));

oout.close()
```

La serialización de objetos Java puede ser una opción perfecta o realmente mala, dependiendo de lo que quieras hacer con ella, fuera del alcance de este tutorial y, a veces, basado en la opinión. Lea acerca del control de versiones primero si decide usarlo.

Escritura en almacenamiento externo (tarjeta SD)

También puede leer y escribir desde / a la tarjeta de memoria (tarjeta SD) que está presente en muchos dispositivos Android. Se puede acceder a los archivos en esta ubicación mediante otros programas, también directamente por el usuario después de conectar el dispositivo a la PC mediante un cable USB y habilitar el protocolo MTP.

Encontrar la ubicación de la tarjeta SD es algo más problemático. La clase de entorno contiene métodos estáticos para obtener "directorios externos" que normalmente deberían estar dentro de la tarjeta SD, también información si la tarjeta SD existe y se puede escribir. Esta pregunta contiene respuestas valiosas sobre cómo asegurarse de que se encontrará la ubicación correcta.

El acceso al almacenamiento externo requiere permisos en tu manifiesto de Android:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Para las versiones anteriores de Android que ponen permisos, es suficiente poner estos permisos en manifiesto (el usuario debe aprobar durante la instalación). Sin embargo, a partir de Android 6.0, Android solicita la aprobación del usuario en el momento del primer acceso, y debe admitir este nuevo enfoque. De lo contrario se niega el acceso, independientemente de su manifiesto.

En Android 6.0, primero debe verificar el permiso y luego, si no se le otorga, solicitarlo. Los ejemplos de código se pueden encontrar dentro de esta pregunta SO .

Resolviendo el problema de "archivos MTP invisibles".

Si crea archivos para exportarlos a través del cable USB al escritorio mediante el protocolo MTP, es posible que los archivos recién creados no se vean inmediatamente en el explorador de archivos que se ejecuta en la computadora de escritorio conectada. Para hacer visibles los nuevos archivos, debe llamar a MediaScannerConnection :

```
File file = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_DOCUMENTS), "theDocument.txt");
FileOutputStream out = new FileOutputStream(file)

... (write the document)

out.close()
MediaScannerConnection.scanFile(this, new String[] {file.getPath()}, null, null);
    context.sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE,
            Uri.fromFile(file)));
```

Este código de llamada MediaScannerConnection funciona solo para archivos, no para directorios. El problema se describe en este informe de error de Android . Esto puede solucionarse para algunas versiones en el futuro o en algunos dispositivos.

Trabajando con archivos grandes

Los archivos pequeños se procesan en una fracción de segundo y puede leerlos / escribirlos en lugar del código donde lo necesite. Sin embargo, si el archivo es más grande o más lento de procesar, es posible que necesite usar AsyncTask en Android para trabajar con el archivo en segundo plano:

```
    class FileOperation extends AsyncTask<String, Void, File> {

        @Override
        protected File doInBackground(String... params) {
          try {
            File file = new File(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_DOCUMENTS), "bigAndComplexDocument.odf");
            FileOutputStream out = new FileOutputStream(file)

           ... (write the document)

            out.close()
            return file;
            } catch (IOException ex) {
               Log.e("Unable to write", ex);
               return null;
```

```
            }
        }

        @Override
        protected void onPostExecute(File result) {
          // This is called when we finish
        }

        @Override
        protected void onPreExecute() {
           // This is called before we begin
        }

        @Override
        protected void onProgressUpdate(Void... values) {
            // Unlikely required for this example
        }
    }
}
```

y entonces

```
new FileOperation().execute("Some parameters");
```

Esta pregunta SO contiene el ejemplo completo sobre cómo crear y llamar a AsyncTask. También vea la pregunta sobre el manejo de errores sobre cómo manejar las excepciones de IO y otros errores.

Lea FileIO con Android en línea: https://riptutorial.com/es/android/topic/8689/fileio-con-android

Capítulo 108: FileProvider

Examples

Compartiendo un archivo

En este ejemplo, aprenderá cómo compartir un archivo con otras aplicaciones. Usaremos un archivo pdf en este ejemplo, aunque el código también funciona con cualquier otro formato.

La hoja de ruta:

Especifique los directorios en los que se ubican los archivos que desea compartir.

Para compartir archivos usaremos un FileProvider, una clase que permite compartir archivos de forma segura entre aplicaciones. Un FileProvider solo puede compartir archivos en directorios predefinidos, así que definamos estos.

1. Cree un nuevo archivo XML que contendrá las rutas, por ejemplo, *res / xml / filepaths.xml*
2. Agrega los caminos

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <files-path name="pdf_folder" path="documents/"/>
</paths>
```

Defina un FileProvider y vincúlelo con las rutas del archivo

Esto se hace en el manifiesto:

```
<manifest>
    ...
    <application>
        ...
        <provider
            android:name="android.support.v4.context.FileProvider"
            android:authorities="com.mydomain.fileprovider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/filepaths" />
        </provider>
        ...
```

```
    </application>
    ...
</manifest>
```

Generar el URI para el archivo

Para compartir el archivo debemos proporcionar un identificador para el archivo. Esto se hace mediante el uso de un URI (Identificador uniforme de recursos).

```
// We assume the file we want to load is in the documents/ subdirectory
// of the internal storage
File documentsPath = new File(Context.getFilesDir(), "documents");
File file = new File(documentsPath, "sample.pdf");
// This can also in one line of course:
// File file = new File(Context.getFilesDir(), "documents/sample.pdf");

Uri uri = FileProvider.getUriForFile(getContext(), "com.mydomain.fileprovider", file);
```

Como puede ver en el código, primero creamos una nueva clase de archivo que representa el archivo. Para obtener un URI, le pedimos a FileProvider que nos obtenga uno. El segundo argumento es importante: pasa la autorización de un proveedor de archivos. Debe ser igual a la autoridad del FileProvider definido en el manifiesto.

Comparte el archivo con otras aplicaciones

Usamos ShareCompat para compartir el archivo con otras aplicaciones:

```
Intent intent = ShareCompat.IntentBuilder.from(getContext())
    .setType("application/pdf")
    .setStream(uri)
    .setChooserTitle("Choose bar")
    .createChooserIntent()
    .addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

Context.startActivity(intent);
```

Un selector es un menú desde el cual el usuario puede elegir con qué aplicación quiere compartir el archivo. La bandera Intent.FLAG_GRANT_READ_URI_PERMISSION es necesaria para otorgar un permiso de acceso de lectura temporal al URI.

Lea FileProvider en línea: https://riptutorial.com/es/android/topic/6266/fileprovider

Capítulo 109: Firebase Cloud Messaging

Introducción

Firebase Cloud Messaging (FCM) es una solución de mensajería multiplataforma que le permite entregar mensajes de manera confiable sin costo alguno.

Con FCM, puede notificar a una aplicación cliente que hay un nuevo correo electrónico u otros datos disponibles para sincronizar. Puede enviar mensajes de notificación para impulsar la reinserción y retención de usuarios. Para casos de uso, como la mensajería instantánea, un mensaje puede transferir una carga útil de hasta 4 `4KB` a una aplicación cliente.

Examples

Configurar una aplicación de cliente de mensajería en la nube Firebase en Android

1. Complete la parte de Instalación y configuración para conectar su aplicación a Firebase. Esto creará el proyecto en Firebase.

2. Agregue la dependencia para Firebase Cloud Messaging a su archivo `build.gradle` nivel de `build.gradle` :

```
dependencies {
     compile 'com.google.firebase:firebase-messaging:10.2.1'
}
```

Ahora estás listo para trabajar con el FCM en Android.

Los clientes de FCM requieren dispositivos con `Android 2.3` o superior que también tengan instalada la aplicación Google Play Store o un emulador con Android 2.3 con API de Google.

Edita tu archivo `AndroidManifest.xml`

```
<service
    android:name=".MyFirebaseMessagingService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
</service>

<service
    android:name=".MyFirebaseInstanceIDService">
    <intent-filter>
        <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
    </intent-filter>
</service>
```

Token de registro

En el inicio inicial de su aplicación, FCM SDK genera un token de registro para la instancia de la aplicación cliente.
Si desea apuntar a dispositivos individuales o crear grupos de dispositivos, deberá acceder a este token extendiendo `FirebaseInstanceIdService` .

La `onTokenRefresh` llamada `onTokenRefresh` cada vez que se genera un token nuevo y puede usar el método `FirebaseInstanceID.getToken()` para recuperar el token actual.

Ejemplo:

```
public class MyFirebaseInstanceIDService extends FirebaseInstanceIdService {

   /**
     * Called if InstanceID token is updated. This may occur if the security of
     * the previous token had been compromised. Note that this is called when the InstanceID
token
     * is initially generated so this is where you would retrieve the token.
     */

    @Override
    public void onTokenRefresh() {
        // Get updated InstanceID token.
        String refreshedToken = FirebaseInstanceId.getInstance().getToken();
        Log.d(TAG, "Refreshed token: " + refreshedToken);

    }

}
```

Este código que he implementado en mi aplicación para enviar imágenes, mensajes y también enlaces para abrir en su webView

Este es mi FirebaseMessagingService

```
    public class MyFirebaseMessagingService extends FirebaseMessagingService {
Bitmap bitmap;
@Override
public void onMessageReceived(RemoteMessage remoteMessage) {
    String message = remoteMessage.getData().get("message");
    //imageUri will contain URL of the image to be displayed with Notification
    String imageUri = remoteMessage.getData().get("image");
    String link=remoteMessage.getData().get("link");

    //To get a Bitmap image from the URL received
    bitmap = getBitmapfromUrl(imageUri);
    sendNotification(message, bitmap,link);

}


/**
 * Create and show a simple notification containing the received FCM message.
 */
```

```
private void sendNotification(String messageBody, Bitmap image, String link) {
    Intent intent = new Intent(this, NewsListActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    intent.putExtra("LINK",link);
    PendingIntent pendingIntent = PendingIntent.getActivity(this, 0 /* Request code */,
intent,
            PendingIntent.FLAG_ONE_SHOT);
    Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
    NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(this)
            .setLargeIcon(image)/*Notification icon image*/
            .setSmallIcon(R.drawable.hindi)
            .setContentTitle(messageBody)
            .setStyle(new NotificationCompat.BigPictureStyle()
                    .bigPicture(image))/*Notification with Image*/
            .setAutoCancel(true)
            .setSound(defaultSoundUri)
            .setContentIntent(pendingIntent);
    NotificationManager notificationManager =
            (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

    notificationManager.notify(0 /* ID of notification */, notificationBuilder.build());
}
public Bitmap getBitmapfromUrl(String imageUrl) {
    try {
        URL url = new URL(imageUrl);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setDoInput(true);
        connection.connect();
        InputStream input = connection.getInputStream();
        Bitmap bitmap = BitmapFactory.decodeStream(input);
        return bitmap;

    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        return null;

    }
}}
```

Y este es MainActivity para abrir el enlace en mi WebView u otro buscador de su requerimiento a través de intenciones.

```
if (getIntent().getExtras() != null) {
    if (getIntent().getStringExtra("LINK")!=null) {
        Intent i=new Intent(this,BrowserActivity.class);
        i.putExtra("link",getIntent().getStringExtra("LINK"));
        i.putExtra("PUSH","yes");
        NewsListActivity.this.startActivity(i);
        finish();
    }}
```

Recibir mensajes

Para recibir mensajes, use un servicio que extienda `FirebaseMessagingService` y anule el método `onMessageReceived` .

```
public class MyFcmListenerService extends FirebaseMessagingService {

    /**
     * Called when message is received.
     *
     * @param remoteMessage Object representing the message received from Firebase Cloud
Messaging.
     */
    @Override
    public void onMessageReceived(RemoteMessage message) {
        String from = message.getFrom();

        // Check if message contains a data payload.
        if (remoteMessage.getData().size() > 0) {
            Log.d(TAG, "Message data payload: " + remoteMessage.getData());
            Map<String, String> data = message.getData();
        }

        // Check if message contains a notification payload.
        if (remoteMessage.getNotification() != null) {
            Log.d(TAG, "Message Notification Body: " +
remoteMessage.getNotification().getBody());
        }

        //.....
    }
```

Cuando la aplicación está en segundo plano, Android dirige los mensajes de notificación a la bandeja del sistema. Un toque de usuario en la notificación abre el iniciador de la aplicación de forma predeterminada.

Esto incluye mensajes que contienen tanto la notificación como la carga de datos (y todos los mensajes enviados desde la consola de notificaciones). En estos casos, la notificación se entrega a la bandeja del sistema del dispositivo, y la carga útil de datos se entrega en los extras de la intención de su Actividad del iniciador.

Aquí un breve resumen:

Estado de la aplicación	Notificación	Datos	Ambos
Primer plano	`onMessageReceived`	`onMessageReceived`	`onMessageReceived`
Fondo	Bandeja del sistema	`onMessageReceived`	Notificación: bandeja del sistema
			Datos: en extras de la intención.

Suscribirse a un tema

Las aplicaciones cliente pueden suscribirse a cualquier tema existente, o pueden crear un tema nuevo. Cuando una aplicación cliente se suscribe a un nuevo nombre de tema, se crea un nuevo

tema con ese nombre en FCM y cualquier cliente puede subscribirse posteriormente.

Para suscribirse a un tema, use el método `subscribeToTopic()` que especifica el nombre del tema:

```
FirebaseMessaging.getInstance().subscribeToTopic("myTopic");
```

Lea Firebase Cloud Messaging en línea: https://riptutorial.com/es/android/topic/8826/firebase-cloud-messaging

Capítulo 110: Firebase Crash Reporting

Examples

Cómo agregar Firebase Crash Reporting a tu aplicación

Para agregar *Firebase Crash Reporting* a su aplicación, realice los siguientes pasos:

- Crea una aplicación en la *Consola Firebase* aquí .
- Copie el archivo `google-services.json` de su proyecto en su directorio in `app/` .
- Agregue las siguientes reglas a su archivo *build.gradle de* nivel *raíz* para incluir el complemento de `google-services` :

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.0.0'
    }
}
```

- En su módulo de archivo Gradle, agregue la línea de `apply plugin` en la parte inferior del archivo para habilitar el complemento de Gradle:

```
apply plugin: 'com.google.gms.google-services'
```

- Agregue la dependencia de *Crash Reporting* a su archivo *build.gradle a* nivel de *aplicación* :

```
compile 'com.google.firebase:firebase-crash:10.2.1'
```

- Luego puede activar una excepción personalizada desde su aplicación usando la siguiente línea:

```
FirebaseCrash.report(new Exception("Non Fatal Error logging"));
```

  Todas sus excepciones fatales serán reportadas a su *Consola Firebase* .

- Si desea agregar registros personalizados a una consola, puede usar el siguiente código:

```
FirebaseCrash.log("Level 2 completed.");
```

Para mayor información por favor visite:

- Documentacion oficial
- Tema dedicado de desbordamiento de pila

Cómo reportar un error

Firebase Crash Reporting genera automáticamente informes de errores fatales (o excepciones no detectadas).

Puede crear su informe personalizado utilizando:

```
FirebaseCrash.report(new Exception("My first Android non-fatal error"));
```

Puede verificar el registro cuando FirebaseCrash inicializó el módulo:

> 07–20 08: 57: 24,442 D / FirebaseCrashApiImpl: **API de informes de FirebaseCrash inicializada** 07–20 08: 57: 24,442 I / FirebaseCrash: **Informes de FirebaseCrash inicializada** d com.google.firebase.crash.internal.zzg@3333d325 07–20 08: 57: 24.442 D / FirebaseApp: **Clase inicializada com.google.firebase.crash.FirebaseCrash.**

Y luego, cuando envió la excepción:

> 07–20 08: 57: 47.052 D / FirebaseCrashApiImpl: **java.lang. ThrowableException: Mi primer error no fatal de Android** 07–20 08: 58: 18.822 D / FirebaseCrashSenderServiceImpl: **Código de respuesta: 200** 07–20 08: 58: 18.822 D / FirebaseCrashSenderServiceImpl: **informe enviado**

Puede agregar registros personalizados a su informe con

```
FirebaseCrash.log("Activity created");
```

Lea Firebase Crash Reporting en línea: https://riptutorial.com/es/android/topic/5965/firebase-crash-reporting

Capítulo 111: Firma tu aplicación de Android para su lanzamiento

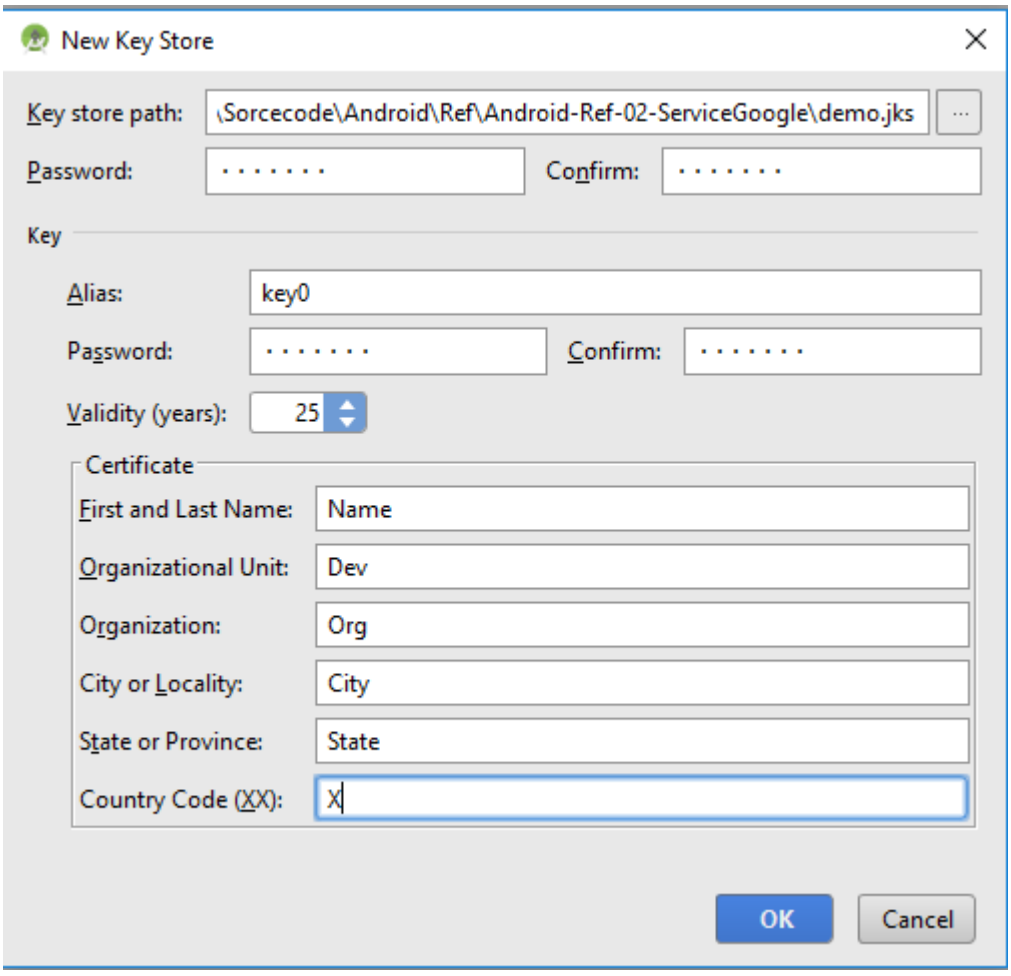
Introducción

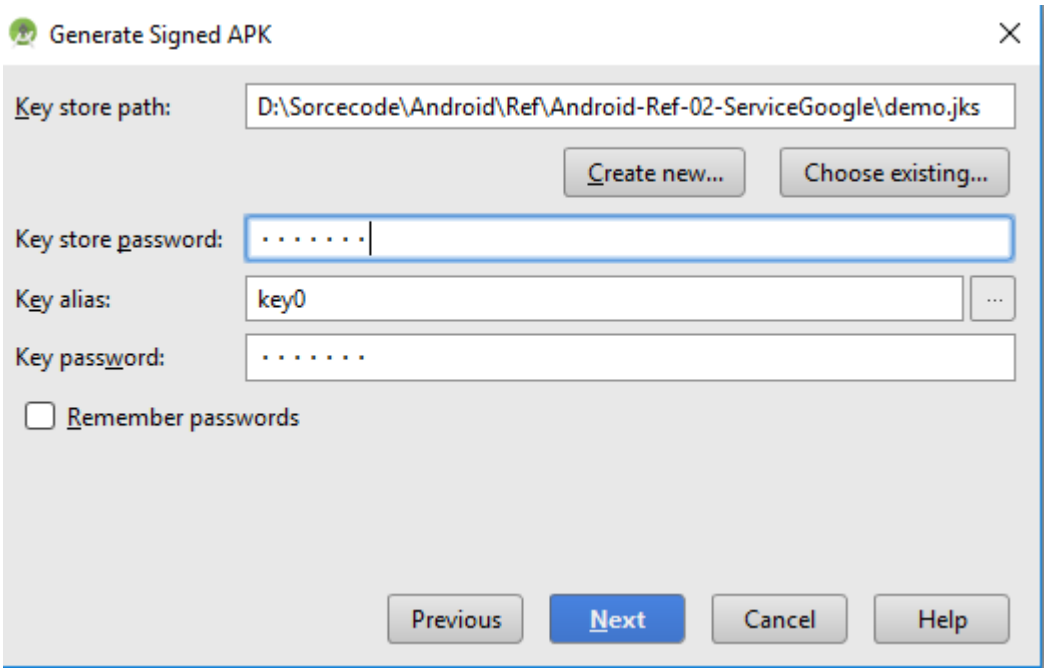
Android requiere que todos los APK estén firmados para su lanzamiento.

Examples

Firma tu aplicación

1. En la barra de menú, haga clic en Crear> Generar APK firmado.
2. Seleccione el módulo que desea liberar del menú desplegable y haga clic en Siguiente.
3. Para crear un nuevo almacén de claves, haga clic en Crear nuevo. Ahora complete la información requerida y presione OK en New Key Store.





4. En el Generar Asistente de APK firmado, los campos ya están completos si usted acaba de crear un nuevo almacén de claves; de lo contrario, llénelo y haga clic en Siguiente.

5. En la siguiente ventana, seleccione un destino para el APK firmado, seleccione el tipo de compilación y haga clic en finalizar.

Configurar el build.gradle con la configuración de firma

Puede definir la configuración de firma para firmar el apk en el archivo `build.gradle` .

Puedes definir:

- `storeFile` : el archivo de almacén de claves
- `storePassword` : la contraseña del almacén de claves
- `keyAlias` : un nombre de alias de clave
- `keyPassword` : una contraseña de alias de clave

Usted tiene que **definir** las `signingConfigs` bloquean para crear una configuración de firma:

```
android {
    signingConfigs {

        myConfig {
            storeFile file("myFile.keystore")
            storePassword "xxxx"
            keyAlias "xxxx"
            keyPassword "xxxx"
        }
    }
    //....
}
```

Luego puedes **asignarlo** a uno o más tipos de compilación.

```
android {

    buildTypes {
        release {
            signingConfig signingConfigs.myConfig
        }
    }
}
```

Lea Firma tu aplicación de Android para su lanzamiento en línea:
https://riptutorial.com/es/android/topic/9721/firma-tu-aplicacion-de-android-para-su-lanzamiento

Capítulo 112: Formato de cadenas

Examples

Formato de un recurso de cadena

Puede agregar comodines en los recursos de cadena y rellenarlos en tiempo de ejecución:

1. Editar cadenas.xml

```
<string name="my_string">This is %1$s</string>
```

2. Formato de cadena según sea necesario

```
String fun = "fun";
context.getString(R.string.my_string, fun);
```

Formato de una marca de tiempo a la cadena

Para una descripción completa de los patrones, vea la referencia SimpleDateFormat

```
Date now = new Date();
long timestamp = now.getTime();
SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy", Locale.US);
String dateStr = sdf.format(timestamp);
```

Formateo de tipos de datos a cadena y viceversa

Tipos de datos a formato de cadena

Los tipos de datos como int, float, double, long, boolean pueden formatearse en cadena usando String.valueOf ().

```
String.valueOf(1); //Output -> "1"
String.valueOf(1.0); //Output -> "1.0"
String.valueOf(1.2345); //Output -> "1.2345"
String.valueOf(true); //Output -> "true"
```

Vise el versa de esto, formateando la cadena a otro tipo de datos

```
Integer.parseInt("1"); //Output -> 1
Float.parseFloat("1.2"); //Output -> 1.2
Boolean.parseBoolean("true"); //Output -> true
```

Lea Formato de cadenas en línea: https://riptutorial.com/es/android/topic/1346/formato-de-cadenas

Capítulo 113: Formato de números de teléfono con patrón.

Introducción

Este ejemplo le muestra cómo dar formato a los números de teléfono con un patrón

Necesitarás la siguiente biblioteca en tu gradle.

compile 'com.googlecode.libphonenumber: libphonenumber: 7.2.2'

Examples

Patrones + 1 (786) 1234 5678

Dado un número de teléfono normalizado como +178612345678 obtendremos un número formateado con el patrón proporcionado.

```
private String getFormattedNumber(String phoneNumber) {

    PhoneNumberUtil phoneNumberUtil = PhoneNumberUtil.getInstance();

    Phonemetadata.NumberFormat numberFormat = new Phonemetadata.NumberFormat();

    numberFormat.pattern = "(\\d{3})(\\d{3})(\\d{4})";

    numberFormat.format = "($1) $2-$3";

    List<Phonemetadata.NumberFormat> newNumberFormats = new ArrayList<>();

    newNumberFormats.add(numberFormat);

    Phonenumber.PhoneNumber phoneNumberPN = null;

    try {
        phoneNumberPN = phoneNumberUtil.parse(phoneNumber, Locale.US.getCountry());
        phoneNumber = phoneNumberUtil.formatByPattern(phoneNumberPN,
PhoneNumberUtil.PhoneNumberFormat.INTERNATIONAL, newNumberFormats);

    } catch (NumberParseException e) {
        e.printStackTrace();
    }

    return phoneNumber;
}
```

Lea Formato de números de teléfono con patrón. en línea:
https://riptutorial.com/es/android/topic/9083/formato-de-numeros-de-telefono-con-patron-

Capítulo 114: Fragmentos

Introducción

Introducción a los fragmentos y su mecanismo de intercomunicación.

Sintaxis

- void onActivityCreated (Bundle savedInstanceState) // Se invoca cuando se crea la actividad del fragmento y se crea una instancia de la jerarquía de vistas de este fragmento.
- void onActivityResult (int requestCode, int resultCode, Intent data) // Recibe el resultado de una llamada anterior a startActivityForResult (Intent, int).
- void onAttach (Actividad de actividad) // Este método ha quedado en desuso en el nivel 23 de la API. Utilice onAttach (Contexto) en su lugar.
- void onAttach (contexto contextual) // Se invoca cuando un fragmento se adjunta por primera vez a su contexto.
- void onAttachFragment (Fragment childFragment) // Se llama cuando un fragmento se adjunta como elemento secundario de este fragmento.
- void onConfigurationChanged (Configuration newConfig) // Llamado por el sistema cuando la configuración del dispositivo cambia mientras su componente se está ejecutando.
- void onCreate (Bundle savedInstanceState) // Llamado para hacer la creación inicial de un fragmento.
- Ver onCreateView (inflador de LayoutInflater, contenedor ViewGroup, Bundle savedInstanceState) // Llamado para que el fragmento cree una instancia de su vista de interfaz de usuario.
- void onDestroy () // Llamado cuando el fragmento ya no está en uso.
- void onDestroyView () // Llamado cuando la vista creada previamente por onCreateView (LayoutInflater, ViewGroup, Bundle) se ha separado del fragmento.
- void onDetach () // Llamado cuando el fragmento ya no está adjunto a su actividad.
- void onInflate (Actividad, AttributeSet attrs, Bundle savedInstanceState) // Este método fue obsoleto en el nivel de API 23. Utilice onInflate (Context, AttributeSet, Bundle) en su lugar.
- void onInflate (contexto de contexto, atributos AttributeSet, Bundle savedInstanceState) // Llamado cuando se crea un fragmento como parte de la inflación de un diseño de vista, generalmente desde la configuración de la vista de contenido de una actividad.

- void onPause () // Llamado cuando el fragmento ya no se reanuda.
- void onResume () // Llamado cuando el fragmento es visible para el usuario y se está ejecutando activamente.
- void onSaveInstanceState (Bundle outState) // Llamado para pedirle al fragmento que guarde su estado dinámico actual, para que luego pueda reconstruirse en una nueva instancia de su proceso que se reinicie.
- void onStart () // Llamado cuando el Fragmento es visible para el usuario.
- void onStop () // Llamado cuando el Fragmento ya no se inicia.
- void onViewStateRestored (Bundle savedInstanceState) // Llamado cuando todo el estado guardado se ha restaurado en la jerarquía de vistas del fragmento.

Observaciones

Un fragmento representa un comportamiento o una parte de la interfaz de usuario en una actividad. Puede combinar varios fragmentos en una sola actividad para crear una interfaz de usuario de múltiples paneles y reutilizar un fragmento en varias actividades. Puede pensar que un fragmento es una sección modular de una actividad, que tiene su propio ciclo de vida, recibe sus propios eventos de entrada y que puede agregar o eliminar mientras la actividad se está ejecutando (como una "subactividad" que puede reutilización en diferentes actividades).

Constructor

Cada fragmento debe tener un **constructor vacío** , por lo que se puede crear una instancia al restaurar el estado de su actividad. Se recomienda encarecidamente que las subclases no tengan otros constructores con parámetros, ya que estos constructores no se llamarán cuando se vuelva a crear una instancia del fragmento; en su lugar, los argumentos pueden ser proporcionados por el llamador con setArguments (Bundle) y luego recuperados por el Fragmento con getArguments ().

Examples

El patrón newInstance ()

Aunque es posible crear un constructor de fragmentos con parámetros, Android llama internamente al constructor de cero argumentos cuando vuelve a crear fragmentos (por ejemplo, si se restauran después de ser eliminados por razones propias de Android). Por esta razón, no es recomendable confiar en un constructor que tenga parámetros.

Para asegurarse de que sus argumentos de fragmentos esperados estén siempre presentes, puede usar un `newInstance()` estático `newInstance()` para crear el fragmento y colocar los parámetros que desee en un paquete que estará disponible cuando cree una nueva instancia.

```
import android.os.Bundle;
import android.support.v4.app.Fragment;

public class MyFragment extends Fragment
{
  // Our identifier for obtaining the name from arguments
  private static final String NAME_ARG = "name";

  private String mName;

  // Required
  public MyFragment(){}

  // The static constructor.  This is the only way that you should instantiate
  // the fragment yourself
  public static MyFragment newInstance(final String name) {
    final MyFragment myFragment = new MyFragment();
    // The 1 below is an optimization, being the number of arguments that will
    // be added to this bundle.  If you know the number of arguments you will add
    // to the bundle it stops additional allocations of the backing map.  If
    // unsure, you can construct Bundle without any arguments
    final Bundle args = new Bundle(1);

    // This stores the argument as an argument in the bundle.  Note that even if
    // the 'name' parameter is NULL then this will work, so you should consider
    // at this point if the parameter is mandatory and if so check for NULL and
    // throw an appropriate error if so
    args.putString(NAME_ARG, name);

    myFragment.setArguments(args);
    return myFragment;
  }

  @Override
  public void onCreate(final Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    final Bundle arguments = getArguments();
    if (arguments == null || !arguments.containsKey(NAME_ARG)) {
      // Set a default or error as you see fit
    } else {
      mName = arguments.getString(NAME_ARG);
    }
  }
}
```

Ahora, en la Actividad:

```
FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
MyFragment mFragment = MyFragment.newInstance("my name");
ft.replace(R.id.placeholder, mFragment);
//R.id.placeholder is where we want to load our fragment
ft.commit();
```

Este patrón es una práctica recomendada para garantizar que todos los argumentos necesarios se pasarán a los fragmentos en la creación. Tenga en cuenta que cuando el sistema destruye el fragmento y lo vuelve a crear más tarde, restaurará automáticamente su estado, pero debe proporcionarle una `onSaveInstanceState(Bundle)` .

Navegación entre fragmentos usando backstack y patrón de tela estático

En primer lugar, debemos agregar nuestro primer `Fragment` al principio, debemos hacerlo en el método `onCreate()` de nuestra Actividad:

```
if (null == savedInstanceState) {
    getSupportFragmentManager().beginTransaction()
      .addToBackStack("fragmentA")
      .replace(R.id.container, FragmentA.newInstance(), "fragmentA")
      .commit();
}
```

A continuación, tenemos que gestionar nuestro backstack. La forma más fácil es usar una función agregada en nuestra actividad que se usa para todas las Transacciones de fragmentos.

```
public void replaceFragment(Fragment fragment, String tag) {
    //Get current fragment placed in container
    Fragment currentFragment = getSupportFragmentManager().findFragmentById(R.id.container);

    //Prevent adding same fragment on top
    if (currentFragment.getClass() == fragment.getClass()) {
        return;
    }

    //If fragment is already on stack, we can pop back stack to prevent stack infinite growth
    if (getSupportFragmentManager().findFragmentByTag(tag) != null) {
        getSupportFragmentManager().popBackStack(tag,
FragmentManager.POP_BACK_STACK_INCLUSIVE);
    }

    //Otherwise, just replace fragment
    getSupportFragmentManager()
            .beginTransaction()
            .addToBackStack(tag)
            .replace(R.id.container, fragment, tag)
            .commit();
}
```

Finalmente, deberíamos anular `onBackPressed()` para salir de la aplicación cuando regresemos del último Fragmento disponible en el backstack.

```
@Override
public void onBackPressed() {
    int fragmentsInStack = getSupportFragmentManager().getBackStackEntryCount();
    if (fragmentsInStack > 1) { // If we have more than one fragment, pop back stack
        getSupportFragmentManager().popBackStack();
    } else if (fragmentsInStack == 1) { // Finish activity, if only one fragment left, to
prevent leaving empty screen
        finish();
    } else {
        super.onBackPressed();
    }
}
```

Ejecución en actividad:

```
replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Ejecución fuera de la actividad (asumiendo que `MainActivity` es nuestra actividad):

```
((MainActivity) getActivity()).replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Pasa los datos de la Actividad al Fragmento usando Bundle

Todos los fragmentos deben tener un constructor vacío (es decir, un método constructor que no tenga argumentos de entrada). Por lo tanto, para pasar sus datos al Fragmento que se está creando, debe usar el método `setArguments()` . Este método obtiene un paquete, en el que almacena sus datos, y almacena el paquete en los argumentos. Posteriormente, este paquete se puede recuperar en las `onCreate()` de `onCreate()` y `onCreateView()` del Fragmento.

Actividad:

```
 Bundle bundle = new Bundle();
 String myMessage = "Stack Overflow is cool!";
 bundle.putString("message", myMessage );
 FragmentClass fragInfo = new FragmentClass();
 fragInfo.setArguments(bundle);
 transaction.replace(R.id.fragment_single, fragInfo);
 transaction.commit();
```

Fragmento:

```
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    String myValue = this.getArguments().getString("message");
    ...
 }
```

Enviar eventos de nuevo a una actividad con interfaz de devolución de llamada

Si necesita enviar eventos de fragmento a actividad, una de las posibles soluciones es definir la interfaz de devolución de llamada y requerir que la actividad del host lo implemente.

Ejemplo

Enviar devolución de llamada a una actividad, cuando se hace clic en el botón del fragmento

En primer lugar, definir la interfaz de devolución de llamada:

```
public interface SampleCallback {
```

```
    void onButtonClicked();
}
```

El siguiente paso es asignar esta devolución de llamada en el fragmento:

```
public final class SampleFragment extends Fragment {

    private SampleCallback callback;

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof SampleCallback) {
            callback = (SampleCallback) context;
        } else {
            throw new RuntimeException(context.toString()
                    + " must implement SampleCallback");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        callback = null;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        final View view = inflater.inflate(R.layout.sample, container, false);
        // Add button's click listener
        view.findViewById(R.id.actionButton).setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                callback.onButtonClicked(); // Invoke callback here
            }
        });
        return view;
    }
}
```

Y finalmente, implementar callback en actividad:

```
public final class SampleActivity extends Activity implements SampleCallback {

    // ... Skipped code with settings content view and presenting the fragment

    @Override
    public void onButtonClicked() {
        // Invoked when fragment's button has been clicked
    }
}
```

Animar la transición entre fragmentos.

Para animar la transición entre fragmentos, o para animar el proceso de mostrar u ocultar un fragmento, utilice `FragmentManager` para crear un `FragmentTransaction` .

Para una `FragmentTransaction` individual, hay dos formas diferentes de realizar animaciones: puede usar una animación estándar o puede proporcionar sus propias animaciones personalizadas.

Las animaciones estándar se especifican llamando a `FragmentTransaction.setTransition(int transit)` y utilizando una de las constantes predefinidas disponibles en la clase `FragmentTransaction` . Al momento de escribir, estas constantes son:

```
FragmentTransaction.TRANSIT_NONE
FragmentTransaction.TRANSIT_FRAGMENT_OPEN
FragmentTransaction.TRANSIT_FRAGMENT_CLOSE
FragmentTransaction.TRANSIT_FRAGMENT_FADE
```

La transacción completa podría ser algo como esto:

```
getSupportFragmentManager()
    .beginTransaction()
    .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

Las animaciones personalizadas se especifican llamando a `FragmentTransaction.setCustomAnimations(int enter, int exit)` o `FragmentTransaction.setCustomAnimations(int enter, int exit, int popEnter, int popExit)` .

Las animaciones de `enter` y `exit` se reproducirán para `FragmentTransaction` s que no impliquen fragmentos extraídos de la pila trasera. Las `popEnter` y `popExit` se reproducirán cuando se saque un fragmento de la pila trasera.

El siguiente código muestra cómo reemplazaría un fragmento al deslizar un fragmento y al otro en su lugar.

```
getSupportFragmentManager()
    .beginTransaction()
    .setCustomAnimations(R.anim.slide_in_left, R.anim.slide_out_right)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

Las definiciones de animación XML usarían la etiqueta `objectAnimator` . Un ejemplo de **slide_in_left.xml** podría verse así:

```
<?xml version="1.0" encoding="utf-8"?>
<set>
  <objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:propertyName="x"
    android:valueType="floatType"
    android:valueFrom="-1280"
    android:valueTo="0"
    android:duration="500"/>
</set>
```

Comunicación entre fragmentos

Todas las comunicaciones entre Fragmentos deben ir a través de una Actividad. Los fragmentos **NO PUEDEN** comunicarse entre sí sin una Actividad.

Recursos adicionales

- Cómo implementar OnFragmentInteractionListener
- Android | Comunicación con otros fragmentos

En este ejemplo, tenemos una `MainActivity` que aloja dos fragmentos, `SenderFragment` y `ReceiverFragment` , para enviar y recibir un `message` (una cadena simple en este caso) respectivamente.

Un botón en `SenderFragment` inicia el proceso de envío del mensaje. Un TextView en `ReceiverFragment` se actualiza cuando recibe el mensaje.

A continuación se encuentra el fragmento de MainActivity con comentarios que explican las líneas de código importantes:

```
// Our MainActivity implements the interface defined by the SenderFragment. This enables
// communication from the fragment to the activity
public class MainActivity extends AppCompatActivity implements
SenderFragment.SendMessageListener {

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}

/**
 * This method is called when we click on the button in the SenderFragment
 * @param message The message sent by the SenderFragment
 */
@Override
public void onSendMessage(String message) {
    // Find our ReceiverFragment using the SupportFragmentManager and the fragment's id
    ReceiverFragment receiverFragment = (ReceiverFragment)
            getSupportFragmentManager().findFragmentById(R.id.fragment_receiver);

    // Make sure that such a fragment exists
    if (receiverFragment != null) {
        // Send this message to the ReceiverFragment by calling its public method
        receiverFragment.showMessage(message);
    }
}
}
```

El archivo de diseño de `MainActivity` aloja dos fragmentos dentro de un LinearLayout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
```

```
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.naru.fragmentcommunication.MainActivity">

<fragment
    android:id="@+id/fragment_sender"
    android:name="com.naru.fragmentcommunication.SenderFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    tools:layout="@layout/fragment_sender" />

<fragment
    android:id="@+id/fragment_receiver"
    android:name="com.naru.fragmentcommunication.ReceiverFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    tools:layout="@layout/fragment_receiver" />
</LinearLayout>
```

El `SenderFragment` expone una interfaz `SendMessageListener` que ayuda a `MainActivity` saber cuándo se hizo clic en el botón en el `SenderFragment` .

A continuación se encuentra el fragmento de código para el `SenderFragment` explica las líneas importantes de código:

```
public class SenderFragment extends Fragment {

private SendMessageListener commander;

/**
 * This interface is created to communicate between the activity and the fragment. Any
activity
 * which implements this interface will be able to receive the message that is sent by this
 * fragment.
 */
public interface SendMessageListener {
    void onSendMessage(String message);
}

/**
 * API LEVEL >= 23
 * <p>
 * This method is called when the fragment is attached to the activity. This method here will
 * help us to initialize our reference variable, 'commander' , for our interface
 * 'SendMessageListener'
 *
 * @param context
 */
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    // Try to cast the context to our interface SendMessageListener i.e. check whether the
    // activity implements the SendMessageListener. If not a ClassCastException is thrown.
    try {
        commander = (SendMessageListener) context;
```

```
    } catch (ClassCastException e) {
        throw new ClassCastException(context.toString()
                + "must implement the SendMessageListener interface");
    }
}

/**
 * API LEVEL < 23
 * <p>
 * This method is called when the fragment is attached to the activity. This method here will
 * help us to initialize our reference variable, 'commander' , for our interface
 * 'SendMessageListener'
 *
 * @param activity
 */
@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    // Try to cast the context to our interface SendMessageListener i.e. check whether the
    // activity implements the SendMessageListener. If not a ClassCastException is thrown.
    try {
        commander = (SendMessageListener) activity;
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString()
                + "must implement the SendMessageListener interface");
    }
}

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
                         @Nullable Bundle savedInstanceState) {
    // Inflate view for the sender fragment.
    View view = inflater.inflate(R.layout.fragment_receiver, container, false);

    // Initialize button and a click listener on it
    Button send = (Button) view.findViewById(R.id.bSend);
    send.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            // Sanity check whether we were able to properly initialize our interface
reference
            if (commander != null) {

                // Call our interface method. This enables us to call the implemented method
                // in the activity, from where we can send the message to the
ReceiverFragment.
                commander.onSendMessage("HELLO FROM SENDER FRAGMENT!");
            }
        }
    });

    return view;
}
}
```

El archivo de diseño para el `SenderFragment` :

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

<Button
    android:id="@+id/bSend"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="SEND"
    android:layout_gravity="center_horizontal" />
</LinearLayout>
```

`ReceiverFragment` es simple y expone un método público simple para actualizar su TextView. Cuando `MainActivity` recibe el mensaje de `SenderFragment` , llama a este método público de `ReceiverFragment`

A continuación se muestra el fragmento de código para `ReceiverFragment` con comentarios que explican las líneas importantes de código:

```
public class ReceiverFragment extends Fragment {
TextView tvMessage;

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
                         @Nullable Bundle savedInstanceState) {
    // Inflate view for the sender fragment.
    View view = inflater.inflate(R.layout.fragment_receiver, container, false);

    // Initialize the TextView
    tvMessage = (TextView) view.findViewById(R.id.tvReceivedMessage);

    return view;
}


/**
 * Method that is called by the MainActivity when it receives a message from the
SenderFragment.
 * This method helps update the text in the TextView to the message sent by the
SenderFragment.
 * @param message Message sent by the SenderFragment via the MainActivity.
 */
public void showMessage(String message) {
    tvMessage.setText(message);
}
}
```

El archivo de diseño para el `ReceiverFragment` :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">
```

```
<TextView
    android:id="@+id/tvReceivedMessage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Waiting for message!" />
</LinearLayout>
```

Lea Fragmentos en línea: https://riptutorial.com/es/android/topic/1396/fragmentos

Capítulo 115: Fresco

Introducción

Fresco es un poderoso sistema para mostrar imágenes en aplicaciones de Android.

En Android 4.x e inferior, Fresco coloca las imágenes en una región especial de la **memoria** de **Android** (llamada ashmem). Esto permite que su aplicación se ejecute más rápido y sufra el temido OutOfMemoryError con mucha menos frecuencia.

Fresco también admite la transmisión de archivos JPEG.

Observaciones

Cómo configurar dependencias en el archivo build.gradle de nivel de aplicación:

```
dependencies {
    // Your app's other dependencies.
    compile 'com.facebook.fresco:fresco:0.14.1' // Or a newer version if available.
}
```

Más información se puede encontrar aquí .

Examples

Empezando con Fresco

Primero, agregue Fresco a su `build.gradle` como se muestra en la sección de Comentarios:

Si necesita funciones adicionales, como soporte GIF animado o WebP, también debe agregar los artefactos de Fresco correspondientes.

Fresco necesita ser inicializado. Solo debe hacer esto 1 vez, por lo que es una buena idea colocar la inicialización en su `Application` . Un ejemplo para esto sería:

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        Fresco.initialize(this);
    }
}
```

Si desea cargar imágenes remotas desde un servidor, su aplicación necesita el permiso interno. Simplemente `AndroidManifest.xml` a tu `AndroidManifest.xml` :

```
<uses-permission android:name="android.permission.INTERNET" />
```

Luego, agregue un `SimpleDraweeView` a su diseño XML. Fresco no admite `wrap_content` para las dimensiones de la imagen, ya que puede tener varias imágenes con diferentes dimensiones (imagen de marcador de posición, imagen de error, imagen real, ...).

Entonces, puedes agregar un `SimpleDraweeView` con dimensiones fijas (o `match_parent`):

```
<com.facebook.drawee.view.SimpleDraweeView
    android:id="@+id/my_image_view"
    android:layout_width="120dp"
    android:layout_height="120dp"
    fresco:placeholderImage="@drawable/placeholder" />
```

O proporcione una *relación de aspecto* para su imagen:

```
<com.facebook.drawee.view.SimpleDraweeView
    android:id="@+id/my_image_view"
    android:layout_width="120dp"
    android:layout_height="wrap_content"
    fresco:viewAspectRatio="1.33"
    fresco:placeholderImage="@drawable/placeholder" />
```

Finalmente, puedes configurar tu imagen URI en Java:

```
SimpleDraweeView draweeView = (SimpleDraweeView) findViewById(R.id.my_image_view);
draweeView.setImageURI("http://yourdomain.com/yourimage.jpg");
```

¡Eso es! Debería ver su marcador de posición dibujable hasta que se haya recuperado la imagen de red.

Usando OkHttp 3 con Fresco

Primero, además de la dependencia normal de Fresco Gradle, debe agregar la dependencia OkHttp 3 a su `build.gradle` :

```
compile "com.facebook.fresco:imagepipeline-okhttp3:1.2.0" // Or a newer version.
```

Cuando inicializa Fresco (generalmente en la implementación de su `Application` personalizada), ahora puede especificar su cliente OkHttp:

```
OkHttpClient okHttpClient = new OkHttpClient(); // Build on your own OkHttpClient.

Context context = ... // Your Application context.
ImagePipelineConfig config = OkHttpImagePipelineConfigFactory
        .newBuilder(context, okHttpClient)
        .build();
Fresco.initialize(context, config);
```

Streaming JPEG con Fresco utilizando DraweeController

Este ejemplo asume que ya ha agregado Fresco a su aplicación (vea este ejemplo):

```
SimpleDraweeView img = new SimpleDraweeView(context);
ImageRequest request = ImageRequestBuilder
        .newBuilderWithSource(Uri.parse("http://example.com/image.png"))
        .setProgressiveRenderingEnabled(true) // This is where the magic happens.
        .build();

DraweeController controller = Fresco.newDraweeControllerBuilder()
        .setImageRequest(request)
        .setOldController(img.getController()) // Get the current controller from our
SimpleDraweeView.
        .build();

img.setController(controller); // Set the new controller to the SimpleDraweeView to enable
progressive JPEGs.
```

Lea Fresco en línea: https://riptutorial.com/es/android/topic/5217/fresco

Capítulo 116: Fuentes personalizadas

Examples

Poner una fuente personalizada en tu aplicación

1. Ir a la (carpeta de proyectos)
2. Entonces la aplicación -> src -> main.
3. Cree la carpeta 'elementos - - fuentes' en la carpeta principal.
4. Ponga su 'fontfile.ttf' en la carpeta de fuentes.

Inicializando una fuente

```
private Typeface myFont;

// A good practice might be to call this in onCreate() of a custom
// Application class and pass 'this' as Context. Your font will be ready to use
// as long as your app lives
public void initFont(Context context) {
    myFont = Typeface.createFromAsset(context.getAssets(), "fonts/Roboto-Light.ttf");
}
```

Usando una fuente personalizada en un TextView

```
public void setFont(TextView textView) {
    textView.setTypeface(myFont);
}
```

Aplicar fuente en TextView por xml (No requiere código Java)

TextViewPlus.java:

```
public class TextViewPlus extends TextView {
    private static final String TAG = "TextView";

    public TextViewPlus(Context context) {
        super(context);
    }

    public TextViewPlus(Context context, AttributeSet attrs) {
        super(context, attrs);
        setCustomFont(context, attrs);
    }

    public TextViewPlus(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        setCustomFont(context, attrs);
    }

    private void setCustomFont(Context ctx, AttributeSet attrs) {
```

```
        TypedArray a = ctx.obtainStyledAttributes(attrs, R.styleable.TextViewPlus);
        String customFont = a.getString(R.styleable.TextViewPlus_customFont);
        setCustomFont(ctx, customFont);
        a.recycle();
    }

    public boolean setCustomFont(Context ctx, String asset) {
        Typeface typeface = null;
        try {
            typeface = Typeface.createFromAsset(ctx.getAssets(), asset);
        } catch (Exception e) {
            Log.e(TAG, "Unable to load typeface: "+e.getMessage());
            return false;
        }

        setTypeface(typeface);
        return true;
    }
}
```

attrs.xml: (Dónde colocar **res / valores**)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="TextViewPlus">
        <attr name="customFont" format="string"/>
    </declare-styleable>
</resources>
```

Cómo utilizar:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:foo="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <com.mypackage.TextViewPlus
        android:id="@+id/textViewPlus1"
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:text="@string/showingOffTheNewTypeface"
        foo:customFont="my_font_name_regular.otf">
    </com.mypackage.TextViewPlus>
</LinearLayout>
```

Fuente personalizada en texto lienzo

Dibujar texto en lienzo con su fuente de activos.

```
Typeface typeface = Typeface.createFromAsset(getAssets(), "fonts/SomeFont.ttf");
Paint textPaint = new Paint();
textPaint.setTypeface(typeface);
canvas.drawText("Your text here", x, y, textPaint);
```

Tipografía eficiente cargando

Cargar fuentes personalizadas puede llevar a un mal rendimiento. Recomiendo usar este pequeño ayudante que guarda / carga sus fuentes ya utilizadas en un Hashtable.

```
public class TypefaceUtils {

private static final Hashtable<String, Typeface> sTypeFaces = new Hashtable<>();

/**
 * Get typeface by filename from assets main directory
 *
 * @param context
 * @param fileName the name of the font file in the asset main directory
 * @return
 */
public static Typeface getTypeFace(final Context context, final String fileName) {
    Typeface tempTypeface = sTypeFaces.get(fileName);

    if (tempTypeface == null) {
        tempTypeface = Typeface.createFromAsset(context.getAssets(), fileName);
        sTypeFaces.put(fileName, tempTypeface);
    }

    return tempTypeface;
}
```

}

Uso:

```
Typeface typeface = TypefaceUtils.getTypeface(context, "RobotoSlab-Bold.ttf");
setTypeface(typeface);
```

Fuente personalizada para toda la actividad.

```
public class ReplaceFont {

public static void changeDefaultFont(Context context, String oldFont, String assetsFont) {
    Typeface typeface = Typeface.createFromAsset(context.getAssets(), assetsFont);
    replaceFont(oldFont, typeface);
}

private static void replaceFont(String oldFont, Typeface typeface) {
    try {
        Field myField = Typeface.class.getDeclaredField(oldFont);
        myField.setAccessible(true);
        myField.set(null, typeface);
    } catch (NoSuchFieldException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    }
}
```

Luego en tu actividad, en el método `onCreate()` :

```
// Put your font to assets folder...

ReplaceFont.changeDefaultFont(getApplication(), "DEFAULT", "LinLibertine.ttf");
```

Trabajando con fuentes en Android O

Android O cambia la forma de trabajar con las fuentes.

Android O presenta una nueva función, denominada *Fuentes en XML* , que le permite usar fuentes como recursos. Esto significa que no hay necesidad de agrupar fuentes como activos. Las fuentes ahora se compilan en un archivo *R* y están disponibles automáticamente en el sistema como un recurso.

Para agregar una nueva **fuente** , debes hacer lo siguiente:

- Crear un nuevo directorio de recursos: `res/font` .
- Agrega tus archivos de fuentes en esta carpeta de fuentes. Por ejemplo, al agregar `myfont.ttf` , podrá usar esta fuente a través de `R.font.myfont` .

También puede crear su propia **familia de fuentes** agregando el siguiente archivo XML en el directorio `res/font` :

```
<?xml version="1.0" encoding="utf-8"?>
<font-family xmlns:android="http://schemas.android.com/apk/res/android">
    <font
        android:fontStyle="normal"
        android:fontWeight="400"
        android:font="@font/lobster_regular" />
    <font
        android:fontStyle="italic"
        android:fontWeight="400"
        android:font="@font/lobster_italic" />
</font-family>
```

Puede utilizar tanto el archivo de **fuente** como el archivo de **familia de fuentes** de la misma manera:

- **En un archivo XML,** utilizando el atributo `android:fontFamily` , por ejemplo, como este:

  ```
  <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:fontFamily="@font/myfont"/>
  ```

  O así:

  ```
  <style name="customfontstyle" parent="@android:style/TextAppearance.Small">
      <item name="android:fontFamily">@font/myfont</item>
  </style>
  ```

- **En su código** , utilizando las siguientes líneas de código:

```
Typeface typeface = getResources().getFont(R.font.myfont);
textView.setTypeface(typeface);
```

Lea Fuentes personalizadas en línea: https://riptutorial.com/es/android/topic/3358/fuentes-personalizadas

Capítulo 117: Genymotion para android

Introducción

Genymotion es un emulador rápido de terceros que se puede usar en lugar del emulador de Android predeterminado. ¡En algunos casos es tan bueno o mejor que el desarrollo en dispositivos reales!

Examples

Instalando Genymotion, la versión gratuita

Paso 1 - instalando `VirtualBox`

Descargue e instale VirtualBox de acuerdo a su sistema operativo. , se requiere para ejecutar `Genymotion` .

Paso 2 - descargando `Genymotion`

Vaya a la página de descarga de `Genymotion` y descargue `Genymotion` acuerdo con su sistema operativo.

Nota: deberá crear una nueva cuenta O iniciar sesión con su cuenta.

Paso 3 - Instalando `Genymotion`

Si está en `Linux` , consulte esta respuesta para instalar y ejecutar un archivo `.bin` .

Paso 4 - Instalando los emuladores de `Genymotion`

- corre `Genymotion`
- Presione el botón Agregar (en la barra superior).
- Inicie sesión con su cuenta y podrá navegar por los emuladores disponibles.
- Selecciona e instala lo que necesites.

Paso 5 - Integración de `genymotion` con `Android Studio`

`Genymotion` , se puede integrar con `Android Studio` través de un complemento, aquí encontrará los pasos para instalarlo en `Android Studio`

- vaya a Archivo / Configuración (para Windows y Linux) o a Android Studio / Preferencias (para Mac OS X)
- Seleccione Complementos y haga clic en Examinar Repositorios.
- Haga clic con el botón derecho en Genymotion y haga clic en Descargar e instalar.

Ahora deberías poder ver el icono del complemento, ver esta imagen

Tenga en cuenta que es posible que desee mostrar la barra de herramientas haciendo clic en Ver> Barra de herramientas.

Paso 6 - Ejecutando `Genymotion` desde `Android Studio`

- vaya a Archivo / Configuración (para Windows y Linux) o a Android Studio / Preferencias (para Mac OS X)
- vaya a Otras configuraciones / Genymotion y agregue la ruta de `Genymotion's` carpeta de `Genymotion's` y aplique sus cambios.

¡Ahora deberías poder ejecutar `Genymotion's` emulador `Genymotion's` presionando el ícono del complemento, seleccionando un emulador instalado y luego presionando el botón de inicio!

Marco de Google en Genymotion

Si los desarrolladores desean probar Google Maps o cualquier otro servicio de Google como Gmail, Youtube, Google drive, etc., primero deben instalar el marco de Google en Genymotion. Aquí están los pasos:

4.4 Kitkat
5.0 Lollipop
5.1 Lollipop
6.0 Malvavisco
7.0 Turrón
7.1 Turrón (parche webview)

1. Descargar desde el enlace de arriba
2. Solo arrastre y suelte el archivo zip descargado a genymotion y reinicie
3. Agrega una cuenta de google y descarga "Google Play Music" y ejecuta.

Referencia:-
Apilar la pregunta de desbordamiento en este tema

Lea Genymotion para android en línea: https://riptutorial.com/es/android/topic/9245/genymotion-para-android

Capítulo 118: Gerente de empaquetación

Examples

Recuperar la versión de la aplicación

```
public String getAppVersion() throws PackageManager.NameNotFoundException {
    PackageManager manager = getApplicationContext().getPackageManager();
    PackageInfo info = manager.getPackageInfo(
                            getApplicationContext().getPackageName(),
                            0);

    return info.versionName;
}
```

Nombre de la versión y código de la versión

Para obtener `versionName` y `versionCode` de la compilación actual de su aplicación, debe consultar el administrador de paquetes de Android.

```
try {
    // Reference to Android's package manager
    PackageManager packageManager = this.getPackageManager();

    // Getting package info of this application
    PackageInfo info = packageManager.getPackageInfo(this.getPackageName(), 0);

    // Version code
    info.versionCode

    // Version name
    info.versionName

} catch (NameNotFoundException e) {
    // Handle the exception
}
```

Instalar tiempo y tiempo de actualización

Para obtener la hora en que se instaló o actualizó su aplicación, debe consultar el administrador de paquetes de Android.

```
try {
    // Reference to Android's package manager
    PackageManager packageManager = this.getPackageManager();

    // Getting package info of this application
    PackageInfo info = packageManager.getPackageInfo(this.getPackageName(), 0);

    // Install time. Units are as per currentTimeMillis().
    info.firstInstallTime
```

```
    // Last update time. Units are as per currentTimeMillis().
    info.lastUpdateTime

} catch (NameNotFoundException e) {
    // Handle the exception
}
```

Método de utilidad utilizando PackageManager

Aquí podemos encontrar algún método útil utilizando PackageManager,

El siguiente método ayudará a obtener el nombre de la aplicación usando el nombre del paquete

```
private String getAppNameFromPackage(String packageName, Context context) {
    Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
    mainIntent.addCategory(Intent.CATEGORY_LAUNCHER);
    List<ResolveInfo> pkgAppsList = context.getPackageManager()
            .queryIntentActivities(mainIntent, 0);
    for (ResolveInfo app : pkgAppsList) {
        if (app.activityInfo.packageName.equals(packageName)) {
            return app.activityInfo.loadLabel(context.getPackageManager()).toString();
        }
    }
    return null;
}
```

El siguiente método ayudará a obtener el ícono de la aplicación usando el nombre del paquete,

```
private Drawable getAppIcon(String packageName, Context context) {
    Drawable appIcon = null;
    try {
        appIcon = context.getPackageManager().getApplicationIcon(packageName);
    } catch (PackageManager.NameNotFoundException e) {
    }

    return appIcon;
}
```

El siguiente método ayudará a obtener la lista de aplicaciones instaladas.

```
public static List<ApplicationInfo> getLaunchIntent(PackageManager packageManager) {

    List<ApplicationInfo> list =
packageManager.getInstalledApplications(PackageManager.GET_META_DATA);

    return list;
}
```

Nota: el método anterior dará la aplicación de inicio también.

El siguiente método ayudará a ocultar el icono de la aplicación desde el iniciador.

```
public static void hideLockerApp(Context context, boolean hide) {
```

```
    ComponentName componentName = new ComponentName(context.getApplicationContext(),
            SplashActivity.class);

    int setting = hide ? PackageManager.COMPONENT_ENABLED_STATE_DISABLED
            : PackageManager.COMPONENT_ENABLED_STATE_ENABLED;

    int current = context.getPackageManager().getComponentEnabledSetting(componentName);

    if (current != setting) {
        context.getPackageManager().setComponentEnabledSetting(componentName, setting,
                PackageManager.DONT_KILL_APP);
    }
}
```

Nota: Después de apagar el dispositivo y encender este icono, volverá al iniciador.

Lea Gerente de empaquetación en línea: https://riptutorial.com/es/android/topic/4670/gerente-de-empaquetacion

Capítulo 119: Google Play Store

Examples

Abra el listado de Google Play Store para su aplicación

El siguiente fragmento de código muestra cómo abrir la Lista de Google Play Store de su aplicación de una manera segura. Por lo general, desea utilizarlo cuando le pide al usuario que deje una revisión para su aplicación.

```
private void openPlayStore() {
    String packageName = getPackageName();
    Intent playStoreIntent = new Intent(Intent.ACTION_VIEW,
            Uri.parse("market://details?id=" + packageName));
    setFlags(playStoreIntent);
    try {
        startActivity(playStoreIntent);
    } catch (Exception e) {
        Intent webIntent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("https://play.google.com/store/apps/details?id=" + packageName));
        setFlags(webIntent);
        startActivity(webIntent);
    }
}

@SuppressWarnings("deprecation")
private void setFlags(Intent intent) {
    intent.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP)
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_DOCUMENT);
    else
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
}
```

Nota : el código abre Google Play Store si la aplicación está instalada. De lo contrario, simplemente se abrirá el navegador web.

Abra Google Play Store con la lista de todas las aplicaciones de su cuenta de editor

Puede agregar un botón "Buscar nuestras otras aplicaciones" en su aplicación, enumerando todas sus aplicaciones (editor) en la aplicación Google Play Store.

```
String urlApp = "market://search?q=pub:Google+Inc.";
String urlWeb = "http://play.google.com/store/search?q=pub:Google+Inc.";
try {
    Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(urlApp));
    setFlags(i);
    startActivity(i);
} catch (android.content.ActivityNotFoundException anfe) {
    Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(urlWeb)));
    setFlags(i);
```

```
    startActivity(i);
}


@SuppressWarnings("deprecation")
public void setFlags(Intent i) {
    i.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP)  {
        i.addFlags(Intent.FLAG_ACTIVITY_NEW_DOCUMENT);
    }
    else  {
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
    }
}
```

Lea Google Play Store en línea: https://riptutorial.com/es/android/topic/10900/google-play-store

Capítulo 120: Gradle para Android

Introducción

Gradle es un sistema de compilación basado en JVM que permite a los desarrolladores escribir scripts de alto nivel que pueden utilizarse para automatizar el proceso de compilación y producción de aplicaciones. Es un sistema flexible basado en complementos, que le permite automatizar varios aspectos del proceso de construcción; incluyendo compilar y firmar un `.jar` , descargar y administrar dependencias externas, inyectar campos en el `AndroidManifest` o utilizar versiones específicas del SDK.

Sintaxis

- `apply plugin` : los complementos que deberían usarse normalmente solo `'com.android.application'` o `'com.android.library'` .
- `android` : la configuración principal de tu aplicación.
    - `compileSdkVersion` : la versión SDK de compilación
    - `buildToolsVersion` : la versión de herramientas de construcción
    - `defaultConfig` : la configuración predeterminada que puede ser sobrescrita por tipos y tipos de compilación
        - `applicationId` : el ID de la aplicación que usas, por ejemplo, en PlayStore, es casi igual al nombre de tu paquete
        - `minSdkVersion` : la versión mínima de SDK requerida
        - `targetSdkVersion` : la versión de SDK con la que compila (debe ser siempre la primera)
        - `versionCode` : el número de versión interna que debe ser mayor en cada actualización
        - `versionName` : el número de versión que el usuario puede ver en la página de detalles de la aplicación
    - `buildTypes` : ver en otro lugar (TODO)
- `dependencies` : las dependencias locales o locales de su aplicación
    - `compile` una sola dependencia
    - `testCompile` : una dependencia para la unidad o pruebas de integración

Observaciones

Ver también

- La página oficial de Gradle.
- Cómo configurar compilaciones de gradle
- El plugin de android para gradle

- Android Gradle DSL

Gradle para Android - Documentación extendida:

Hay otra etiqueta donde puedes encontrar más temas y ejemplos sobre el uso de gradle en Android.
http://www.riptutorial.com/topic/2092

Examples

Un archivo build.gradle básico

Este es un ejemplo de un archivo `build.gradle` predeterminado en un módulo.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion '25.0.3'

    signingConfigs {
        applicationName {
            keyAlias 'applicationName'
            keyPassword 'password'
            storeFile file('../key/applicationName.jks')
            storePassword 'keystorePassword'
        }
    }
    defaultConfig {
        applicationId 'com.company.applicationName'
        minSdkVersion 14
        targetSdkVersion 25
        versionCode 1
        versionName '1.0'
        signingConfig signingConfigs.applicationName
    }
    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])

    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support:design:25.3.1'

    testCompile 'junit:junit:4.12'
}
```

DSL (lenguaje específico de dominio)

Cada bloque en el archivo anterior se llama un `DSL` (lenguaje específico del dominio).

Complementos

La primera línea, `apply plugin: 'com.android.application'` , aplica el complemento de Android para Gradle a la compilación y hace que el bloque `android {}` esté disponible para declarar las opciones de compilación específicas de Android.

Para una **aplicación de Android** :

```
apply plugin: 'com.android.application'
```

Para una **biblioteca de Android** :

```
apply plugin: 'com.android.library'
```

Entendiendo los DSLs en el ejemplo anterior

La segunda parte, el bloque de `android {...}` , es el `DSL` Android que contiene información sobre su proyecto.

Por ejemplo, puede configurar el `compileSdkVersion` que especifica el nivel de la API de Android, que Gradle debe usar para compilar su aplicación.
El subbloque `defaultConfig` contiene los valores predeterminados para su manifiesto. Puede `override` con Sabores del producto .

Puedes encontrar más información en estos ejemplos:

- DSL para el módulo de la aplicación
- Tipos de construcción
- Sabores del producto
- Configuracion de firma

Dependencias

El bloque de `dependencies` se define fuera del bloque de `android {...}` : Esto significa que no está definido por el complemento de Android, pero es Gradle estándar.
El bloque de `dependencies` especifica qué bibliotecas externas (normalmente las bibliotecas de

Android, pero las bibliotecas de Java también son válidas) que desea incluir en su aplicación. Gradle descargará automáticamente estas dependencias por usted (si no hay una copia local disponible), solo necesita agregar líneas de `compile` similares cuando desee agregar otra biblioteca.

Veamos una de las líneas aquí presentes:

```
compile 'com.android.support:design:25.3.1'
```

Esta línea básicamente dice

> agregar una dependencia de la biblioteca de diseño de soporte de Android a mi proyecto.

Gradle se asegurará de que la biblioteca esté descargada y presente para que pueda usarla en su aplicación, y su código también se incluirá en su aplicación.

Si está familiarizado con Maven, esta sintaxis es *GroupId* , dos puntos, *ArtifactId* , otros dos puntos, luego la versión de la dependencia que desea incluir, lo que le da un control total sobre las versiones.

Si bien es posible especificar versiones de artefactos usando el signo más (+), la mejor práctica es evitar hacerlo; puede llevar a problemas si la biblioteca se actualiza con cambios de última hora sin su conocimiento, lo que probablemente provocaría bloqueos en su aplicación.

Puedes agregar diferentes tipos de dependencias:

- dependencias binarias locales
- dependencias del módulo
- dependencias remotas

Se debe dedicar una atención particular a las dependencias planas .

Puede encontrar más detalles en este tema.

Nota sobre el **-v7 en *appcompat-v7***

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

Esto simplemente significa que esta **biblioteca** (`appcompat`) es compatible con la API de Android de nivel 7 y `appcompat` .

Nota sobre el **junit: junit: 4.12**

Esta es la dependencia de prueba para la prueba de unidad.

Especificando dependencias específicas para diferentes configuraciones de compilación

Puede especificar que una dependencia solo se use para una determinada configuración de compilación o puede definir diferentes dependencias para los tipos de compilación o las versiones del producto (por ejemplo, depuración, prueba o lanzamiento) utilizando `debugCompile` , `testCompile` o `releaseCompile` lugar de la `compile` habitual .

Esto es útil para mantener las dependencias relacionadas con la prueba y la depuración fuera de su versión de lanzamiento, lo que mantendrá su `APK` versión lo más delgado posible y ayudará a garantizar que no se pueda usar ninguna información de depuración para obtener información interna sobre su aplicación.

---

firmaConfig

La `signingConfig` permite configurar su Gradle para incluir información del `keystore` y garantizar que el APK creado con estas configuraciones esté firmado y listo para la versión de Play Store.

Aquí puedes encontrar un tema dedicado .

Nota : no se recomienda mantener las credenciales de firma dentro de su archivo de Gradle. Para eliminar las configuraciones de firma, basta con omitir la `signingConfigs` parte.
Puedes especificarlos de diferentes maneras:

- almacenar en un archivo externo
- Almacenándolos en la configuración de variables de entorno .

Consulte este tema para obtener más detalles: Firmar APK sin exponer la contraseña del almacén de claves .

Puede encontrar más información sobre Gradle para Android en el tema dedicado de Gradle .

Definición de sabores de producto.

Los sabores del producto se definen en el archivo `build.gradle` dentro del bloque de `android { ... }` como se ve a continuación.

```
...
android {
    ...
    productFlavors {
        free {
            applicationId "com.example.app.free"
            versionName "1.0-free"
        }
        paid {
            applicationId "com.example.app.paid"
            versionName "1.0-paid"
        }
    }
```

```
}
```

Al hacer esto, ahora tenemos dos sabores de productos adicionales: `free` y de `paid` . Cada uno puede tener su propia configuración y atributos específicos. Por ejemplo, nuestros dos nuevos sabores tienen un `applicationId` y `versionName` separados de nuestro `main` sabor existente (disponible por defecto, por lo que no se muestra aquí).

Adición de dependencias específicas del sabor del producto.

Se pueden agregar dependencias para un sabor de producto específico, similar a cómo se pueden agregar para configuraciones de compilación específicas.

Para este ejemplo, suponga que ya hemos definido dos sabores de productos llamados `free` y de `paid` (más información sobre cómo definir sabores aquí).
Luego podemos agregar la dependencia de AdMob para el sabor `free` , y la biblioteca de Picasso para el `paid` como:

```
android {
    ...

    productFlavors {
        free {
            applicationId "com.example.app.free"
            versionName "1.0-free"
        }
        paid {
            applicationId "com.example.app.paid"
            versionName "1.0-paid"
        }
    }
}

...
dependencies {
    ...
    // Add AdMob only for free flavor
    freeCompile 'com.android.support:appcompat-v7:23.1.1'
    freeCompile 'com.google.android.gms:play-services-ads:8.4.0'
    freeCompile 'com.android.support:support-v4:23.1.1'

    // Add picasso only for paid flavor
    paidCompile 'com.squareup.picasso:picasso:2.5.2'
}
...
```

Añadiendo recursos específicos del sabor del producto.

Se pueden agregar recursos para un sabor de producto específico.

Para este ejemplo, suponga que ya hemos definido dos tipos de productos llamados `free` y de `paid` . Para agregar recursos específicos del sabor del producto, creamos carpetas de recursos adicionales junto con la carpeta `main/res` , a la que luego podemos agregar recursos como de costumbre. Para este ejemplo, definiremos una cadena, `status` , para cada sabor de producto:

*/ src / **main** /res/values/strings.xml*

```
<resources>
    <string name="status">Default</string>
</resources>
```

*/ src / **free** /res/values/strings.xml*

```
<resources>
    <string name="status">Free</string>
</resources>
```

*/ src / **paid** /res/values/strings.xml*

```
<resources>
    <string name="status">Paid</string>
</resources>
```

Las cadenas de `status` específicas del sabor del producto anularán el valor del `status` en el sabor `main` .

Definir y usar los campos de configuración de construcción

BuildConfigField

Gradle permite que `buildConfigField` líneas `buildConfigField` definan constantes. Estas constantes serán accesibles en tiempo de ejecución como campos estáticos de la clase `BuildConfig` . Esto se puede usar para crear sabores definiendo todos los campos dentro del bloque `defaultConfig` , y luego reemplazándolos para crear sabores individuales según sea necesario.

Este ejemplo define la fecha de compilación y marca la compilación para la producción en lugar de la prueba:

```
android {
    ...
    defaultConfig {
        ...
        // defining the build date
        buildConfigField "long", "BUILD_DATE", System.currentTimeMillis() + "L"
        // define whether this build is a production build
        buildConfigField "boolean", "IS_PRODUCTION", "false"
        // note that to define a string you need to escape it
        buildConfigField "String", "API_KEY", "\"my_api_key\""
    }

    productFlavors {
        prod {
            // override the productive flag for the flavor "prod"
            buildConfigField "boolean", "IS_PRODUCTION", "true"
            resValue 'string', 'app_name', 'My App Name'
        }
        dev {
```

```
            // inherit default fields
            resValue 'string', 'app_name', 'My App Name - Dev'
        }
    }
}
```

El <package_name> generado automáticamente. `BuildConfig` .java en la carpeta gen contiene los siguientes campos basados en la directiva anterior:

```
public class BuildConfig {
    // ... other generated fields ...
    public static final long BUILD_DATE = 1469504547000L;
    public static final boolean IS_PRODUCTION = false;
    public static final String API_KEY = "my_api_key";
}
```

Los campos definidos ahora se pueden usar dentro de la aplicación en tiempo de ejecución accediendo a la clase `BuildConfig` generada:

```
public void example() {
    // format the build date
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");
    String buildDate = dateFormat.format(new Date(BuildConfig.BUILD_DATE));
    Log.d("build date", buildDate);

    // do something depending whether this is a productive build
    if (BuildConfig.IS_PRODUCTION) {
        connectToProductionApiEndpoint();
    } else {
        connectToStagingApiEndpoint();
    }
}
```

Valorar

El `resValue` en los `productFlavors` crea un valor de recursos. Puede ser cualquier tipo de recurso (`string` , `dimen` , `color` , etc.). Esto es similar a definir un recurso en el archivo apropiado: por ejemplo, definir una cadena en un archivo `strings.xml` . La ventaja es que el definido en gradle se puede modificar en función de su productFlavor / buildVariant. Para acceder al valor, escriba el mismo código como si estuviera accediendo a una resolución desde el archivo de recursos:

```
getResources().getString(R.string.app_name)
```

Lo importante es que los recursos definidos de esta manera no pueden modificar los recursos existentes definidos en los archivos. Solo pueden crear nuevos valores de recursos.

Algunas bibliotecas (como la API de Android de Google Maps) requieren una clave API proporcionada en el manifiesto como una etiqueta de `meta-data` . Si se necesitan claves diferentes para la depuración y las compilaciones de producción, especifique un marcador de posición manifiesto completado por Gradle.

En su archivo `AndroidManifest.xml` :

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="${MAPS_API_KEY}"/>
```

Y luego establezca el campo correspondiente en su archivo `build.gradle` :

```
android {
    defaultConfig {
        ...
        // Your development key
        manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
    }

    productFlavors {
        prod {
            // Your production key
            manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
        }
    }
}
```

El sistema de compilación de Android genera una serie de campos automáticamente y los coloca en `BuildConfig.java` . Estos campos son:

Campo	Descripción
`DEBUG`	un `Boolean` indica si la aplicación está en modo de depuración o lanzamiento
`APPLICATION_ID`	una `String` contiene el ID de la aplicación (por ejemplo, `com.example.app`)
`BUILD_TYPE`	una `String` contiene el tipo de compilación de la aplicación (generalmente `debug` o `release`)
`FLAVOR`	una `String` contiene el sabor particular de la construcción
`VERSION_CODE`	un `int` contiene el número de versión (compilación). Esto es lo mismo que `versionCode` en `build.gradle` o `versionCode` en `AndroidManifest.xml`
`VERSION_NAME`	una `String` contiene el nombre de la versión (compilación). Este es el mismo como `versionName` en `build.gradle` o `versionName` en `AndroidManifest.xml`

Además de lo anterior, si ha definido múltiples dimensiones de sabor, entonces cada dimensión tendrá su propio valor. Por ejemplo, si tiene dos dimensiones de sabor para el `color` y el `size` , también tendrá las siguientes variables:

Campo	Descripción
`FLAVOR_color`	una `String` contiene el valor para el sabor 'color'.

Campo	Descripción
`FLAVOR_size`	una `String` contiene el valor para el sabor 'tamaño'.

Centralizando dependencias a través del archivo "dependencies.gradle"

Cuando se trabaja con proyectos de múltiples módulos, es útil centralizar las dependencias en una sola ubicación en lugar de tenerlos distribuidos en muchos archivos de compilación, especialmente para bibliotecas comunes como las bibliotecas de soporte de Android y las bibliotecas Firebase .

Una forma recomendada es separar los archivos de compilación de Gradle, con un `build.gradle` por módulo, así como uno en la raíz del proyecto y otro para las dependencias, por ejemplo:

```
root
  +- gradleScript/
  |     dependencies.gradle
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
```

Entonces, todas sus dependencias se pueden ubicar en `gradleScript/dependencies.gradle` :

```
ext {
    // Version
    supportVersion = '24.1.0'

    // Support Libraries dependencies
    supportDependencies = [
            design:            "com.android.support:design:${supportVersion}",
            recyclerView:      "com.android.support:recyclerview-v7:${supportVersion}",
            cardView:          "com.android.support:cardview-v7:${supportVersion}",
            appCompat:         "com.android.support:appcompat-v7:${supportVersion}",
            supportAnnotation: "com.android.support:support-annotations:${supportVersion}",
    ]

    firebaseVersion = '9.2.0';

    firebaseDependencies = [
            core:         "com.google.firebase:firebase-core:${firebaseVersion}",
            database:     "com.google.firebase:firebase-database:${firebaseVersion}",
            storage:      "com.google.firebase:firebase-storage:${firebaseVersion}",
            crash:        "com.google.firebase:firebase-crash:${firebaseVersion}",
            auth:         "com.google.firebase:firebase-auth:${firebaseVersion}",
            messaging:    "com.google.firebase:firebase-messaging:${firebaseVersion}",
            remoteConfig: "com.google.firebase:firebase-config:${firebaseVersion}",
            invites:      "com.google.firebase:firebase-invites:${firebaseVersion}",
            adMod:        "com.google.firebase:firebase-ads:${firebaseVersion}",
            appIndexing:  "com.google.android.gms:play-services-
appindexing:${firebaseVersion}",
    ];
}
```

Que luego se puede aplicar desde ese archivo en el archivo de nivel superior `build.gradle` así:

```
// Load dependencies
apply from: 'gradleScript/dependencies.gradle'
```

y en el `module1/build.gradle` como tal:

```
// Module build file
dependencies {
    // ...
    compile supportDependencies.appCompat
    compile supportDependencies.design
    compile firebaseDependencies.crash
}
```

Otro enfoque

Se puede lograr un enfoque menos detallado para centralizar las versiones de las dependencias de la biblioteca declarando el número de versión como una variable una vez, y usándolo en todas partes.

En el espacio de trabajo root `build.gradle` agregue esto:

```
ext.v = [
    supportVersion:'24.1.1',
]
```

Y en cada módulo que use la misma biblioteca agregue las bibliotecas necesarias

```
compile "com.android.support:support-v4:${v.supportVersion}"
compile "com.android.support:recyclerview-v7:${v.supportVersion}"
compile "com.android.support:design:${v.supportVersion}"
compile "com.android.support:support-annotations:${v.supportVersion}"
```

Estructura de directorio para recursos específicos de sabor

Diferentes tipos de compilaciones de aplicaciones pueden contener diferentes recursos. Para crear un recurso de sabor específico, cree un directorio con el nombre en minúsculas de su sabor en el directorio `src` y agregue sus recursos de la misma manera que lo haría normalmente.

Por ejemplo, si tuviera un `Development` sabor y quisiera proporcionar un ícono de `src/development/res/drawable-mdpi` distinto, crearía un directorio `src/development/res/drawable-mdpi` y dentro de ese directorio crearía un archivo `ic_launcher.png` con su ícono específico de desarrollo.

La estructura del directorio se verá así:

```
src/
  main/
    res/
```

```
      drawable-mdpi/
        ic_launcher.png  <-- the default launcher icon
  development/
    res/
      drawable-mdpi/
        ic_launcher.png  <-- the launcher icon used when the product flavor is 'Development'
```

(Por supuesto, en este caso, también crearías íconos para drawable-hdpi, drawable-xhdpi, *etc.*).

¿Por qué hay dos archivos build.gradle en un proyecto de Android Studio?

`<PROJECT_ROOT>\app\build.gradle` es específico para el **módulo de la aplicación** .

`<PROJECT_ROOT>\build.gradle` es un **"archivo de compilación de nivel superior"** donde puede agregar opciones de configuración comunes a todos los subproyectos / módulos.

Si usa otro módulo en su proyecto, como biblioteca local tendrá otro archivo `build.gradle` : `<PROJECT_ROOT>\module\build.gradle`

En el archivo de nivel superior puede especificar propiedades comunes como el bloque de buildscript o algunas propiedades comunes.

```
buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
       classpath 'com.android.tools.build:gradle:2.2.0'
       classpath 'com.google.gms:google-services:3.0.0'
    }
}

ext {
    compileSdkVersion = 23
    buildToolsVersion = "23.0.1"
}
```

En la `app\build.gradle` usted define solo las propiedades para el módulo:

```
apply plugin: 'com.android.application'


android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion
}

dependencies {
    //.....
}
```

Ejecutando un script de shell desde gradle

Un script de shell es una forma muy versátil de ampliar su compilación a básicamente cualquier cosa que se pueda imaginar.

Como ejemplo, aquí hay un script simple para compilar archivos protobuf y agregar los archivos java de resultados al directorio de origen para una compilación adicional:

```
def compilePb() {
    exec {
        // NOTICE: gradle will fail if there's an error in the protoc file...
        executable "../pbScript.sh"
    }
}

project.afterEvaluate {
    compilePb()
}
```

El script de shell 'pbScript.sh' para este ejemplo, ubicado en la carpeta raíz del proyecto:

```
#!/usr/bin/env bash
pp=/home/myself/my/proto

/usr/local/bin/protoc -I=$pp \
 --java_out=./src/main/java \
  --proto_path=$pp \
 $pp/my.proto \
 --proto_path=$pp \
 $pp/my_other.proto
```

Depurando tus errores de Gradle

El siguiente es un extracto de Gradle: ¿Qué es un valor de salida distinto de cero y cómo puedo solucionarlo? , verlo para la discusión completa.

Digamos que está desarrollando una aplicación y obtiene un error de Gradle que parece que, en general, se verá así.

```
:module:someTask FAILED
FAILURE: Build failed with an exception.
* What went wrong:
Execution failed for task ':module:someTask'.
> some message here...  finished with non-zero exit value X
* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get
more log output.
BUILD FAILED
Total time: Y.ZZ secs
```

Busca tu problema aquí en StackOverflow, y la gente dice que debes limpiar y reconstruir tu proyecto, o habilitar MultiDex , y cuando lo intentas, simplemente no está solucionando el problema.

Hay formas de obtener más información , pero la salida de Gradle en sí misma debería apuntar al

error real en las pocas líneas sobre ese mensaje entre: `module:someTask FAILED` y el último `:module:someOtherTask` que pasó. Por lo tanto, si hace una pregunta sobre su error, edite sus preguntas para incluir más contexto al error.

Entonces, obtienes un "valor de salida distinto de cero". Bueno, ese número es un buen indicador de lo que debes tratar de arreglar. Aquí hay algunos que ocurren con más frecuencia.

- `1` es solo un código de error general y el error es probable en la salida de Gradle
- `2` parece estar relacionado con la superposición de dependencias o la configuración errónea del proyecto.
- `3` parece ser por incluir demasiadas dependencias, o un problema de memoria.

Las soluciones generales para lo anterior (después de intentar limpiar y reconstruir el proyecto) son:

- `1` - Abordar el error que se menciona. En general, este es un error en tiempo de compilación, lo que significa que parte del código de su proyecto no es válido. Esto incluye tanto XML como Java para un proyecto de Android.
- `2` y `3` : muchas respuestas aquí le dicen que habilite multidex . Si bien puede solucionar el problema, es muy probable que sea una solución. Si no entiende por qué lo está utilizando (vea el enlace), probablemente no lo necesite. Las soluciones generales implican reducir su uso excesivo de las dependencias de la biblioteca (como todos los Servicios de Google Play, cuando solo necesita usar una biblioteca, como Mapas o Iniciar sesión, por ejemplo).

Especificar diferentes ID de aplicación para tipos de compilación y sabores de producto

Puede especificar diferentes ID de aplicación o nombres de paquetes para cada `buildType` o `productFlavor` utilizando el atributo de configuración **applicationIdSuffix:**

Ejemplo de sufijo del `applicationId` para cada `buildType` :

```
defaultConfig {
    applicationId "com.package.android"
    minSdkVersion 17
    targetSdkVersion 23
    versionCode 1
    versionName "1.0"
}

buildTypes {
    release {
        debuggable false
    }

    development {
        debuggable true
        applicationIdSuffix ".dev"
    }

    testing {
        debuggable true
```

```
        applicationIdSuffix ".qa"
    }
}
```

Nuestra `applicationIds` resultante sería ahora:

- com.package.android para el `release`
- com.package.android. **dev** para `development`
- com.package.android. **qa** para la `testing`

Esto también se puede hacer para `productFlavors` :

```
productFlavors {
    free {
        applicationIdSuffix ".free"
    }
    paid {
        applicationIdSuffix ".paid"
    }
}
```

Las `applicationIds` resultantes serían:

- com.package.android. **Gratis** para el sabor `free`
- com.package.android. **pagado** por el sabor `paid`

Firmar APK sin exponer la contraseña del keystore

Puede definir la configuración de firma para firmar el apk en el archivo `build.gradle` usando estas propiedades:

- `storeFile` : el archivo de almacén de claves
- `storePassword` : la contraseña del almacén de claves
- `keyAlias` : un nombre de alias de clave
- `keyPassword` : una contraseña de alias de clave

En muchos casos, es posible que deba evitar este tipo de información en el archivo `build.gradle` .

Método A: configure la firma de liberación utilizando un archivo keystore.properties

Es posible configurar `build.gradle` su aplicación para que lea la información de configuración de firma de un archivo de propiedades como `keystore.properties` .

Configurar la firma de esta manera es beneficioso porque:

- Su información de configuración de firma es independiente de su archivo `build.gradle`
- No tiene que intervenir durante el proceso de firma para proporcionar contraseñas para su

archivo de almacén de claves
- Puede excluir fácilmente el archivo `keystore.properties` del control de versiones

Primero, cree un archivo llamado `keystore.properties` en la raíz de su proyecto con contenido como este (reemplazando los valores con los suyos):

```
storeFile=keystore.jks
storePassword=storePassword
keyAlias=keyAlias
keyPassword=keyPassword
```

Ahora, en el archivo `build.gradle` su aplicación, configure el bloque de `signingConfigs` configuración de la siguiente manera:

```
android {
...

    signingConfigs {
        release {
            def propsFile = rootProject.file('keystore.properties')
            if (propsFile.exists()) {
                def props = new Properties()
                props.load(new FileInputStream(propsFile))
                storeFile = file(props['storeFile'])
                storePassword = props['storePassword']
                keyAlias = props['keyAlias']
                keyPassword = props['keyPassword']
            }
        }
    }
}
```

Eso es todo lo que hay en ello, **pero no olvide excluir tanto su archivo de almacén de claves como su archivo de `keystore.properties` del control de versiones** .

Un par de cosas a anotar:

- La ruta de `storeFile` especificada en el archivo `keystore.properties` debe ser relativa al archivo `build.gradle` su aplicación. Este ejemplo asume que el archivo de almacén de claves está en el mismo directorio que el archivo `build.gradle` la aplicación.
- Este ejemplo tiene el archivo `keystore.properties` en la raíz del proyecto. Si lo coloca en otro lugar, asegúrese de cambiar el valor en `rootProject.file('keystore.properties')` a su ubicación, en relación con la raíz de su proyecto.

Método B: utilizando una variable de entorno

Lo mismo se puede lograr también sin un archivo de propiedades, lo que hace que la contraseña sea más difícil de encontrar:

```
android {
```

```
  signingConfigs {
    release {
        storeFile file('/your/keystore/location/key')
        keyAlias 'your_alias'
        String ps = System.getenv("ps")
        if (ps == null) {
             throw new GradleException('missing ps env variable')
        }
        keyPassword ps
        storePassword ps
    }
}
```

La variable de entorno `"ps"` puede ser global, pero un enfoque más seguro puede ser agregándolo a la shell de Android Studio solamente.
En Linux, esto se puede hacer editando la `Desktop Entry` Android Studio

```
Exec=sh -c "export ps=myPassword123 ; /path/to/studio.sh"
```

Puede encontrar más detalles en este tema .

Versiones de sus compilaciones a través del archivo "version.properties"

Puedes usar Gradle para incrementar automáticamente la versión de tu paquete cada vez que lo construyas. Para ello, cree un archivo `version.properties` en el mismo directorio que su `build.gradle` con el siguiente contenido:

```
VERSION_MAJOR=0
VERSION_MINOR=1
VERSION_BUILD=1
```

(Cambiando los valores para mayor y menor como mejor le parezca). Luego, en tu `build.gradle` agrega el siguiente código a la sección de `android` :

```
// Read version information from local file and increment as appropriate
def versionPropsFile = file('version.properties')
if (versionPropsFile.canRead()) {
  def Properties versionProps = new Properties()

  versionProps.load(new FileInputStream(versionPropsFile))

  def versionMajor = versionProps['VERSION_MAJOR'].toInteger()
  def versionMinor = versionProps['VERSION_MINOR'].toInteger()
  def versionBuild = versionProps['VERSION_BUILD'].toInteger() + 1

  // Update the build number in the local file
  versionProps['VERSION_BUILD'] = versionBuild.toString()
  versionProps.store(versionPropsFile.newWriter(), null)

  defaultConfig {
    versionCode versionBuild
    versionName "${versionMajor}.${versionMinor}." + String.format("%05d", versionBuild)
  }
```

```
}
```

Se puede acceder a la información en Java como una cadena `BuildConfig.VERSION_NAME` para completar {major}. { `BuildConfig.VERSION_NAME` }. { `BuildConfig.VERSION_CODE` } y como un entero `BuildConfig.VERSION_CODE` solo para el número de compilación.

Cambiar el nombre del apk de salida y agregar el nombre de la versión:

Este es el código para cambiar el nombre del archivo de la aplicación de salida (.apk). El nombre se puede configurar asignando un valor diferente a `newName`

```
android {

    applicationVariants.all { variant ->
        def newName = "ApkName";
        variant.outputs.each { output ->
            def apk = output.outputFile;

            newName += "-v" + defaultConfig.versionName;
            if (variant.buildType.name == "release") {
                newName += "-release.apk";
            } else {
                newName += ".apk";
            }
            if (!output.zipAlign) {
                newName = newName.replace(".apk", "-unaligned.apk");
            }

            output.outputFile = new File(apk.parentFile, newName);
            logger.info("INFO: Set outputFile to "
                        + output.outputFile
                        + " for [" + output.name + "]");
        }
    }
}
```

Deshabilite la compresión de imágenes para un tamaño de archivo APK más pequeño

Si está optimizando todas las imágenes manualmente, desactive APT Cruncher para un tamaño de archivo APK más pequeño.

```
android {

    aaptOptions {
        cruncherEnabled = false
    }
}
```

Habilitar Proguard usando gradle

Para habilitar las configuraciones de Proguard para su aplicación, necesita habilitarla en su archivo de nivel de módulo. `minifyEnabled` establecer el valor de `minifyEnabled` en `true` .

```
buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
```

El código anterior aplicará sus configuraciones de Proguard contenidas en el SDK de Android predeterminado combinado con el archivo "proguard-rules.pro" en su módulo a su apk liberado.

Habilitar el soporte experimental del complemento NDK para Gradle y AndroidStudio

Habilite y configure el complemento experimental de Gradle para mejorar el soporte NDK de AndroidStudio. Comprueba que cumples los siguientes requisitos:

- Gradle 2.10 (para este ejemplo)
- Android NDK r10 o posterior
- Android SDK con herramientas de compilación v19.0.0 o posterior

Configurar el archivo MyApp / build.gradle

Edite la línea dependencies.classpath en build.gradle desde, por ejemplo,

```
classpath 'com.android.tools.build:gradle:2.1.2'
```

a

```
classpath 'com.android.tools.build:gradle-experimental:0.7.2'
```

(v0.7.2 era la última versión en el momento de la redacción. Verifique la última versión usted mismo y adapte su línea en consecuencia)

El archivo build.gradle debería verse similar a esto:

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle-experimental:0.7.2'
    }
}

allprojects {
    repositories {
        jcenter()
    }
}
```

```
task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Configurar el archivo MyApp / app / build.gradle

Edite el archivo build.gradle para que se vea similar al siguiente ejemplo. Sus números de versión pueden parecer diferentes.

```
apply plugin: 'com.android.model.application'

model {
    android {
        compileSdkVersion 19
        buildToolsVersion "24.0.1"

        defaultConfig {
            applicationId "com.example.mydomain.myapp"
            minSdkVersion.apiLevel 19
            targetSdkVersion.apiLevel 19
            versionCode 1
            versionName "1.0"
        }
        buildTypes {
            release {
                minifyEnabled false
                proguardFiles.add(file('proguard-android.txt'))
            }
        }
        ndk {
            moduleName "myLib"

            /* The following lines are examples of a some optional flags that
               you may set to configure your build environment
            */
            cppFlags.add("-I${file("path/to/my/includes/dir")}".toString())
            cppFlags.add("-std=c++11")
            ldLibs.addAll(['log', 'm'])
            stl = "c++_static"
            abiFilters.add("armeabi-v7a")
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

Sincronice y verifique que no haya errores en los archivos de Gradle antes de continuar.

Probar si el plugin está habilitado

Primero asegúrese de haber descargado el módulo NDK de Android. Luego cree una nueva aplicación en AndroidStudio y agregue lo siguiente al archivo ActivityMain:

```
public class MainActivity implements Activity {
    onCreate() {
        // Pregenerated code. Not important here
    }
    static {
        System.loadLibrary("myLib");
    }
    public static native String getString();
}
```

La parte `getString()` debe resaltarse en rojo diciendo que no se pudo encontrar la función JNI correspondiente. Mueva el mouse sobre la función de llamada hasta que aparezca una bombilla roja. Haga clic en la bombilla y seleccione `create function JNI_...` Esto debería generar un archivo myLib.c en el directorio myApp / app / src / main / jni con la llamada a la función JNI correcta. Debería verse similar a esto:

```
#include <jni.h>

JNIEXPORT jstring JNICALL
Java_com_example_mydomain_myapp_MainActivity_getString(JNIEnv *env, jobject instance)
{
    // TODO

    return (*env)->NewStringUTF(env, returnValue);
}
```

Si no se ve así, entonces el complemento no se ha configurado correctamente o el NDK no se ha descargado

Mostrar todas las tareas del proyecto Gradle

```
gradlew tasks -- show all tasks



Android tasks
-------------
androidDependencies - Displays the Android dependencies of the project.
signingReport - Displays the signing info for each variant.
sourceSets - Prints out all the source sets defined in this project.

Build tasks
-----------
assemble - Assembles all variants of all applications and secondary packages.
assembleAndroidTest - Assembles all the Test applications.
assembleDebug - Assembles all Debug builds.
assembleRelease - Assembles all Release builds.
build - Assembles and tests this project.
buildDependents - Assembles and tests this project and all projects that depend on it.
buildNeeded - Assembles and tests this project and all projects it depends on.
classes - Assembles main classes.
clean - Deletes the build directory.
```

```
compileDebugAndroidTestSources
compileDebugSources
compileDebugUnitTestSources
compileReleaseSources
compileReleaseUnitTestSources
extractDebugAnnotations - Extracts Android annotations for the debug variant into the archive
file
extractReleaseAnnotations - Extracts Android annotations for the release variant into the
archive file
jar - Assembles a jar archive containing the main classes.
mockableAndroidJar - Creates a version of android.jar that's suitable for unit tests.
testClasses - Assembles test classes.

Build Setup tasks
-----------------
init - Initializes a new Gradle build. [incubating]
wrapper - Generates Gradle wrapper files. [incubating]

Documentation tasks
-------------------
javadoc - Generates Javadoc API documentation for the main source code.

Help tasks
----------
buildEnvironment - Displays all buildscript dependencies declared in root project
'LeitnerBoxPro'.
components - Displays the components produced by root project 'LeitnerBoxPro'. [incubating]
dependencies - Displays all dependencies declared in root project 'LeitnerBoxPro'.
dependencyInsight - Displays the insight into a specific dependency in root project
'LeitnerBoxPro'.
help - Displays a help message.
model - Displays the configuration model of root project 'LeitnerBoxPro'. [incubating]
projects - Displays the sub-projects of root project 'LeitnerBoxPro'.
properties - Displays the properties of root project 'LeitnerBoxPro'.
tasks - Displays the tasks runnable from root project 'LeitnerBoxPro' (some of the displayed
tasks may belong to subprojects)
.

Install tasks
-------------
installDebug - Installs the Debug build.
installDebugAndroidTest - Installs the android (on device) tests for the Debug build.
uninstallAll - Uninstall all applications.
uninstallDebug - Uninstalls the Debug build.
uninstallDebugAndroidTest - Uninstalls the android (on device) tests for the Debug build.
uninstallRelease - Uninstalls the Release build.

Verification tasks
------------------
check - Runs all checks.
connectedAndroidTest - Installs and runs instrumentation tests for all flavors on connected
devices.
connectedCheck - Runs all device checks on currently connected devices.
connectedDebugAndroidTest - Installs and runs the tests for debug on connected devices.
deviceAndroidTest - Installs and runs instrumentation tests using all Device Providers.
deviceCheck - Runs all device checks using Device Providers and Test Servers.
lint - Runs lint on all variants.
lintDebug - Runs lint on the Debug build.
lintRelease - Runs lint on the Release build.
test - Run unit tests for all variants.
testDebugUnitTest - Run unit tests for the debug build.
```

```
testReleaseUnitTest - Run unit tests for the release build.

Other tasks
-----------
assembleDefault
clean
jarDebugClasses
jarReleaseClasses
transformResourcesWithMergeJavaResForDebugUnitTest
transformResourcesWithMergeJavaResForReleaseUnitTest
```

Eliminar "no alineado" apk automáticamente

Si no necesita archivos apk generados automáticamente con sufijo `unaligned` (que probablemente no necesite), puede agregar el siguiente código al archivo `build.gradle` :

```
// delete unaligned files
android.applicationVariants.all { variant ->
  variant.assemble.doLast {
    variant.outputs.each { output ->
        println "aligned " + output.outputFile
        println "unaligned " + output.packageApplication.outputFile

        File unaligned = output.packageApplication.outputFile;
        File aligned = output.outputFile
        if (!unaligned.getName().equalsIgnoreCase(aligned.getName())) {
            println "deleting " + unaligned.getName()
            unaligned.delete()
        }
    }
  }
}
```

Desde aqui

Ignorando la variante de construcción

Por algunas razones, es posible que desee ignorar las variantes de compilación. Por ejemplo: tiene un sabor de producto 'simulado' y lo usa solo para fines de depuración, como pruebas de unidad / instrumentación.

Ignoremos la variante **mockRelease** de nuestro proyecto. Abra el archivo **build.gradle** y escriba:

```
    // Remove mockRelease as it's not needed.
    android.variantFilter { variant ->
        if (variant.buildType.name.equals('release') &&
variant.getFlavors().get(0).name.equals('mock')) {
            variant.setIgnore(true);
        }
    }
```

Viendo arbol de dependencias

Usa las dependencias de la tarea. Dependiendo de cómo estén configurados los módulos, puede

ser `./gradlew dependencies` o ver las dependencias del uso de la aplicación del módulo `./gradlew :app:dependencies`

El siguiente ejemplo del archivo build.gradle

```
dependencies {
    compile 'com.android.support:design:23.2.1'
    compile 'com.android.support:cardview-v7:23.1.1'

    compile 'com.google.android.gms:play-services:6.5.87'
}
```

Producirá el siguiente gráfico:

```
Parallel execution is an incubating feature.
:app:dependencies

------------------------------------------------------------
Project :app
------------------------------------------------------------
. . .
_releaseApk - ## Internal use, do not manually configure ##
+--- com.android.support:design:23.2.1
|    +--- com.android.support:support-v4:23.2.1
|    |    \--- com.android.support:support-annotations:23.2.1
|    +--- com.android.support:appcompat-v7:23.2.1
|    |    +--- com.android.support:support-v4:23.2.1 (*)
|    |    +--- com.android.support:animated-vector-drawable:23.2.1
|    |    |    \--- com.android.support:support-vector-drawable:23.2.1
|    |    |         \--- com.android.support:support-v4:23.2.1 (*)
|    |    \--- com.android.support:support-vector-drawable:23.2.1 (*)
|    \--- com.android.support:recyclerview-v7:23.2.1
|         +--- com.android.support:support-v4:23.2.1 (*)
|         \--- com.android.support:support-annotations:23.2.1
+--- com.android.support:cardview-v7:23.1.1
\--- com.google.android.gms:play-services:6.5.87
     \--- com.android.support:support-v4:21.0.0 -> 23.2.1 (*)

. . .
```

Aquí puede ver que el proyecto incluye directamente `com.android.support:design` versión 23.2.1, que a su vez trae `com.android.support:support-v4` con la versión 23.2.1. Sin embargo, `com.google.android.gms:play-services` sí mismo depende del mismo `support-v4` pero con una versión anterior 21.0.0, que es un conflicto detectado por gradle.

`(*)` se utilizan cuando gradle se salta el subárbol porque esas dependencias ya estaban listadas anteriormente.

Use gradle.properties para central versionnumber / buildconfigurations

Puede definir la información de configuración central en

- un archivo de inclusión de Gradle separado. Centralización de dependencias a través del archivo "dependencies.gradle"
- un archivo de propiedades autónomas que versiona sus compilaciones a través del archivo

"version.properties"

o hazlo con el archivo raíz `gradle.properties`

la estructura del proyecto

```
root
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
  +- gradle.properties
```

configuración global para todos los submódulos en gradle.properties

```
# used for manifest
# todo increment for every release
appVersionCode=19
appVersionName=0.5.2.160726

# android tools settings
appCompileSdkVersion=23
appBuildToolsVersion=23.0.2
```

uso en un submódulo

```
apply plugin: 'com.android.application'
android {
    // appXXX are defined in gradle.properties
    compileSdkVersion = Integer.valueOf(appCompileSdkVersion)
    buildToolsVersion = appBuildToolsVersion

    defaultConfig {
        // appXXX are defined in gradle.properties
        versionCode = Long.valueOf(appVersionCode)
        versionName = appVersionName
    }
}

dependencies {
    ...
}
```

Nota: Si desea publicar su aplicación en la tienda de aplicaciones F-Droid, tiene que usar números mágicos en el archivo de gradle porque, de lo contrario, el robot f-droid no puede leer la versión actual para detectar / verificar los cambios de versión.

Mostrar información de firma

En algunas circunstancias (por ejemplo, la obtención de una clave API de Google) debe encontrar la huella digital del almacén de claves. Gradle tiene una tarea conveniente que muestra toda la información de firma, incluidas las huellas digitales del almacén de claves:

```
./gradlew signingReport
```

Esta es una salida de muestra:

```
:app:signingReport
Variant: release
Config: none
----------
Variant: debug
Config: debug
Store: /Users/user/.android/debug.keystore
Alias: AndroidDebugKey
MD5: 25:08:76:A9:7C:0C:19:35:99:02:7B:00:AA:1E:49:CA
SHA1: 26:BE:89:58:00:8C:5A:7D:A3:A9:D3:60:4A:30:53:7A:3D:4E:05:55
Valid until: Saturday 18 June 2044
----------
Variant: debugAndroidTest
Config: debug
Store: /Users/user/.android/debug.keystore
Alias: AndroidDebugKey
MD5: 25:08:76:A9:7C:0C:19:35:99:02:7B:00:AA:1E:49:CA
SHA1: 26:BE:89:58:00:8C:5A:7D:A3:A9:D3:60:4A:30:53:7A:3D:4E:05:55
Valid until: Saturday 18 June 2044
----------
Variant: debugUnitTest
Config: debug
Store: /Users/user/.android/debug.keystore
Alias: AndroidDebugKey
MD5: 25:08:76:A9:7C:0C:19:35:99:02:7B:00:AA:1E:49:CA
SHA1: 26:BE:89:58:00:8C:5A:7D:A3:A9:D3:60:4A:30:53:7A:3D:4E:05:55
Valid until: Saturday 18 June 2044
----------
Variant: releaseUnitTest
Config: none
----------
```

Definiendo tipos de compilación

Puede crear y configurar tipos de compilación en el archivo `build.gradle` nivel de `build.gradle` dentro del bloque `android {}` .

```
    android {
        ...
        defaultConfig {...}

        buildTypes {
            release {
                minifyEnabled true
                proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-
rules.pro'
            }

            debug {
                applicationIdSuffix ".debug"
            }
        }
    }
```

Lea Gradle para Android en línea: https://riptutorial.com/es/android/topic/95/gradle-para-android

Capítulo 121: GreenDAO

Introducción

GreenDAO es una biblioteca de mapeo de objetos relacionales para ayudar a los desarrolladores a usar bases de datos SQLite para el almacenamiento local persistente.

Examples

Métodos de ayuda para las consultas SELECT, INSERT, DELETE, UPDATE

Este ejemplo muestra una clase auxiliar que contiene métodos útiles cuando se ejecutan las consultas de datos. Cada método aquí utiliza Java Genérico para ser muy flexible.

```
public <T> List<T> selectElements(AbstractDao<T, ?> dao) {
    if (dao == null) {
        return null;
    }
    QueryBuilder<T> qb = dao.queryBuilder();
    return qb.list();
}

public <T> void insertElements(AbstractDao<T, ?> absDao, List<T> items) {
    if (items == null || items.size() == 0 || absDao == null) {
        return;
    }
    absDao.insertOrReplaceInTx(items);
}

public <T> T insertElement(AbstractDao<T, ?> absDao, T item) {
    if (item == null || absDao == null) {
        return null;
    }
    absDao.insertOrReplaceInTx(item);
    return item;
}

public <T> void updateElements(AbstractDao<T, ?> absDao, List<T> items) {
    if (items == null || items.size() == 0 || absDao == null) {
        return;
    }
    absDao.updateInTx(items);
}

public <T> T selectElementByCondition(AbstractDao<T, ?> absDao,
                                      WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    List<T> items = qb.list();
```

```
    return items != null && items.size() > 0 ? items.get(0) : null;
}

public <T> List<T> selectElementsByCondition(AbstractDao<T, ?> absDao,
                                             WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T> List<T> selectElementsByConditionAndSort(AbstractDao<T, ?> absDao,
                                                    Property sortProperty,
                                                    String sortStrategy,
                                                    WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    qb.orderCustom(sortProperty, sortStrategy);
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T> List<T> selectElementsByConditionAndSortWithNullHandling(AbstractDao<T, ?> absDao,
                                                                    Property sortProperty,
                                                                    boolean handleNulls,
                                                                    String sortStrategy,
                                                                    WhereCondition...
conditions) {
    if (!handleNulls) {
        return selectElementsByConditionAndSort(absDao, sortProperty, sortStrategy,
conditions);
    }
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    qb.orderRaw("(CASE WHEN " + "T." + sortProperty.columnName + " IS NULL then 1 ELSE 0
END)," + "T." + sortProperty.columnName + " " + sortStrategy);
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T, V extends Class> List<T> selectByJoin(AbstractDao<T, ?> absDao,
                                                 V className,
                                                 Property property, WhereCondition
whereCondition) {
    QueryBuilder<T> qb = absDao.queryBuilder();
    qb.join(className, property).where(whereCondition);
```

```
    return qb.list();
}

public <T> void deleteElementsByCondition(AbstractDao<T, ?> absDao,
                                          WhereCondition... conditions) {
    if (absDao == null) {
        return;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    List<T> list = qb.list();
    absDao.deleteInTx(list);
}

public <T> T deleteElement(DaoSession session, AbstractDao<T, ?> absDao, T object) {
    if (absDao == null) {
        return null;
    }
    absDao.delete(object);
    session.clear();
    return object;
}

public <T, V extends Class> void deleteByJoin(AbstractDao<T, ?> absDao,
                                              V className,
                                              Property property, WhereCondition
whereCondition) {
    QueryBuilder<T> qb = absDao.queryBuilder();
    qb.join(className, property).where(whereCondition);
    qb.buildDelete().executeDeleteWithoutDetachingEntities();
}

public <T> void deleteAllFromTable(AbstractDao<T, ?> absDao) {
    if (absDao == null) {
        return;
    }
    absDao.deleteAll();
}

public <T> long countElements(AbstractDao<T, ?> absDao) {
    if (absDao == null) {
        return 0;
    }
    return absDao.count();
}
```

Creación de una entidad con GreenDAO 3.X que tiene una clave primaria compuesta

Al crear un modelo para una tabla que tiene una clave primaria compuesta, se requiere trabajo adicional en el Objeto para que la Entidad modelo respete esas restricciones.

La siguiente tabla SQL de ejemplo y Entidad muestra la estructura para almacenar una revisión dejada por un cliente para un artículo en una tienda en línea. En este ejemplo, queremos que las columnas `customer_id` y `item_id` sean una clave primaria compuesta, permitiendo que solo exista una revisión entre un cliente específico y un artículo.

Tabla SQL

```
CREATE TABLE review (
    customer_id STRING NOT NULL,
    item_id STRING NOT NULL,
    star_rating INTEGER NOT NULL,
    content STRING,
    PRIMARY KEY (customer_id, item_id)
);
```

Por lo general, `@Unique` anotaciones `@Id` y `@Unique` sobre los campos respectivos en la clase de entidad, sin embargo, para una clave primaria compuesta hacemos lo siguiente:

1. Agregue la anotación `@Index` dentro de la anotación `@Entity` nivel de `@Entity` . La propiedad de valor contiene una lista delimitada por comas de los campos que conforman la clave. Use la propiedad `unique` como se muestra para imponer la singularidad en la clave.

2. GreenDAO requiere que cada Entidad tenga un objeto `long` o `Long` como clave principal. Aún necesitamos agregar esto a la clase Entidad, sin embargo, no necesitamos usarlo o preocuparnos de que esto afecte nuestra implementación. En el siguiente ejemplo se llama `localID`

Entidad

```
@Entity(indexes = { @Index(value = "customer_id,item_id", unique = true)})
public class Review {

    @Id(autoincrement = true)
    private Long localID;

    private String customer_id;
    private String item_id;

    @NotNull
    private Integer star_rating;

    private String content;

    public Review() {}
}
```

Empezando con GreenDao v3.X

Después de agregar la dependencia de la biblioteca GreenDao y el complemento Gradle, primero debemos crear un objeto de entidad.

Entidad

Una entidad es un objeto Java antiguo simple *(POJO)* que modela algunos datos en la base de datos. GreenDao usará esta clase para crear una tabla en la base de datos SQLite y generar automáticamente las clases auxiliares que podemos usar para acceder y almacenar datos sin tener que escribir sentencias de SQL.

```
@Entity
public class Users {

    @Id(autoincrement = true)
    private Long id;

    private String firstname;
    private String lastname;

    @Unique
    private String email;

    // Getters and setters for the fields...

}
```

Una sola vez configuración de GreenDao

Cada vez que se lanza una aplicación, GreenDao necesita ser inicializada. GreenDao sugiere mantener este código en una clase de aplicación o en algún lugar que solo se ejecutará una vez.

```
DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this, "mydatabase", null);
db = helper.getWritableDatabase();
DaoMaster daoMaster = new DaoMaster(db);
DaoSession daoSession = daoMaster.newSession();
```

Clases de ayuda de GreenDao

Después de que se crea el objeto de entidad, GreenDao crea automáticamente las clases auxiliares utilizadas para interactuar con la base de datos. Estos se denominan de forma similar al nombre del objeto de entidad que se creó, seguido de `Dao` y se recuperan del objeto `daoSession` .

```
UsersDao usersDao = daoSession.getUsersDao();
```

Muchas acciones típicas de la base de datos ahora se pueden realizar usando este objeto Dao con el objeto entidad.

Consulta

```
String email = "jdoe@example.com";
String firstname = "John";

// Single user query WHERE email matches "jdoe@example.com"
Users user = userDao.queryBuilder()
                .where(UsersDao.Properties.Email.eq(email)).build().unique();

// Multiple user query WHERE firstname = "John"
List<Users> user = userDao.queryBuilder()
                .where(UsersDao.Properties.Firstname.eq(firstname)).build().list();
```

Insertar

```
Users newUser = new User("John","Doe","jdoe@example.com");
usersDao.insert(newUser);
```

Actualizar

```
// Modify a previously retrieved user object and update
user.setLastname("Dole");
usersDao.update(user);
```

Borrar

```
// Delete a previously retrieved user object
usersDao.delete(user);
```

Lea GreenDAO en línea: https://riptutorial.com/es/android/topic/1345/greendao

Capítulo 122: GreenRobot EventBus

Sintaxis

- @Subscribe (threadMode = ThreadMode.POSTING) public void onEvent (EventClass event) {}

Parámetros

Modo hilo	Descripción
`ThreadMode.POSTING`	Se llamará en el mismo hilo en el que se publicó el evento. Este es el modo por defecto.
`ThreadMode.MAIN`	Se llamará en el hilo principal de la interfaz de usuario.
`ThreadMode.BACKGROUND`	Será llamado en un hilo de fondo. Si el hilo de publicación no es el hilo principal, se utilizará. Si se publica en el hilo principal, `EventBus` tiene un solo hilo de fondo que utilizará.
`ThreadMode.ASYNC`	Será llamado en su propio hilo.

Examples

Creando un objeto de evento

Para enviar y recibir eventos, primero necesitamos un objeto de evento. Los objetos de eventos son realmente POJOs simples.

```
public class ArbitaryEvent{
    public static final int TYPE_1 = 1;
    public static final int TYPE_2 = 2;
    private int eventType;
    public ArbitaryEvent(int eventType){
        this.eventType = eventType;
    }

    public int getEventType(){
        return eventType;
    }
}
```

Recibir eventos

Para recibir eventos necesita registrar su clase en el `EventBus` .

```
@Override
public void onStart() {
    super.onStart();
    EventBus.getDefault().register(this);
}

@Override
public void onStop() {
   EventBus.getDefault().unregister(this);
   super.onStop();
}
```

Y luego suscribirse a los eventos.

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void handleEvent(ArbitaryEvent event) {
    Toast.makeText(getActivity(), "Event type: "+event.getEventType(),
Toast.LENGTH_SHORT).show();
}
```

Enviando eventos

Enviar eventos es tan fácil como crear el objeto Evento y luego publicarlo.

```
EventBus.getDefault().post(new ArbitaryEvent(ArbitaryEvent.TYPE_1));
```

Pasando un evento simple

Lo primero que tenemos que hacer es agregar EventBus al archivo gradle de nuestro módulo:

```
dependencies {
    ...
    compile 'org.greenrobot:eventbus:3.0.0'
    ...
}
```

Ahora necesitamos crear un modelo para nuestro evento. Puede contener cualquier cosa que queramos transmitir. Por ahora solo haremos una clase vacía.

```
public class DeviceConnectedEvent
{
}
```

Ahora podemos agregar el código a nuestra `Activity` que se registrará en EventBus y suscribirse al evento.

```
public class MainActivity extends AppCompatActivity
{
    private EventBus _eventBus;

    @Override
    protected void onCreate (Bundle savedInstanceState)
```

```
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        _eventBus = EventBus.getDefault();
    }

    @Override
    protected void onStart ()
    {
        super.onStart();
        _eventBus.register(this);
    }

    @Override
    protected void onStop ()
    {
        _eventBus.unregister(this);
        super.onStop();
    }

    @Subscribe(threadMode = ThreadMode.MAIN)
    public void onDeviceConnected (final DeviceConnectedEvent event)
    {
        // Process event and update UI
    }
}
```

En esta `Activity` obtenemos una instancia de `EventBus` en el método `onCreate()` . Registramos / desregistramos para eventos en `onStart()` / `onStop()` . Es importante recordar cancelar el registro cuando su interlocutor pierda el alcance o podría perder su `Activity` .

Finalmente definimos el método que queremos llamar con el evento. La anotación `@Subscribe` le dice a EventBus qué métodos puede buscar para manejar eventos. `@Subscribe` tener al menos un método anotado con `@Subscribe` para registrarse en EventBus o se producirá una excepción. En la anotación definimos el modo hilo. Esto le dice a EventBus en qué hilo invocar el método. ¡Es una forma muy útil de pasar información de un subproceso en segundo plano al subproceso de la interfaz de usuario! Eso es exactamente lo que estamos haciendo aquí. `ThreadMode.MAIN` significa que este método se llamará en el hilo principal de la interfaz de usuario de Android, por lo que es seguro realizar cualquier manipulación de la interfaz de usuario que necesite. El nombre del método no importa. El único pensamiento, aparte de la anotación `@Subscribe` , que EventBus está buscando es el tipo de argumento. Siempre que el tipo coincida, se llamará cuando se publique un evento.

Lo último que tenemos que hacer para publicar un evento. Este código estará en nuestro `Service` .

```
EventBus.getDefault().post(new DeviceConnectedEvent());
```

¡Eso es todo al respecto! EventBus tomará ese DeviceConnectedEvent y examinará sus escuchas registrados, examinará los métodos que se han suscrito y encontrará los que toman un DeviceConnectedEvent como un argumento y los llamará al hilo en el que desean que se les llame.

Lea GreenRobot EventBus en línea: https://riptutorial.com/es/android/topic/3551/greenrobot-eventbus

Capítulo 123: Gson

Introducción

Gson es una biblioteca de Java que se puede usar para convertir objetos de Java en su representación JSON. Gson considera que ambos son objetivos de diseño muy importantes.

Características de Gson:

Proporcione `toJson()` simples `toJson()` y `fromJson()` para convertir objetos Java a JSON y viceversa

Permitir que los objetos no modificables preexistentes se conviertan ay desde JSON

Amplio soporte de Java Generics

Admite objetos complejos arbitrariamente (con jerarquías de herencia profundas y uso extensivo de tipos genéricos)

Sintaxis

- Excluder excluder ()
- FieldNamingStrategy fieldNamingStrategy ()
- <T> T fromJson (JsonElement json, Class <T> classOfT)
- <T> T fromJson (JsonElement json, tipo typeOfT)
- <T> T fromJson (lector JsonReader, tipo typeOfT)
- <T> T fromJson (Reader json, Class <T> classOfT)
- <T> T fromJson (Reader json, tipo typeOfT)
- <T> T fromJson (String json, Class <T> classOfT)
- <T> T fromJson (String json, Type typeOfT)
- <T> TypeAdapter <T> getAdapter (clase <T> tipo)
- <T> TypeAdapter <T> getAdapter (TypeToken <T> type)
- <T> TypeAdapter <T> getDelegateAdapter (TypeAdapterFactory skipPast, TypeToken <T> type)
- JsonReader newJsonReader (lector de lectores)
- JsonWriter newJsonWriter (escritor escritor)
- JsonElement toJsonTree (Object src)
- JsonElement toJsonTree (Object src, Type typeOfSrc)
- boolean serializeNulls ()
- booleano htmlSafe ()
- String toJson (JsonElement jsonElement)
- String toJson (Object src)
- String toJson (Object src, Type typeOfSrc)
- String toString ()
- void toJson (Object src, Type typeOfSrc, Appendable writer)

- void toJson (Object src, Type typeOfSrc, escritor JsonWriter)
- void toJson (JsonElement jsonElement, escritor anexable)
- void toJson (JsonElement jsonElement, escritor JsonWriter)
- void toJson (Object src, Appendable writer)

Examples

Analizando JSON con Gson

El ejemplo muestra el análisis de un objeto JSON utilizando la biblioteca Gson de Google .

Analizando objetos:

```
class Robot {
    //OPTIONAL - this annotation allows for the key to be different from the field name, and
can be omitted if key and field name are same . Also this is good coding practice as it
decouple your variable names with server keys name
    @SerializedName("version")
    private String version;

    @SerializedName("age")
    private int age;

    @SerializedName("robotName")
    private String name;

    // optional : Benefit it allows to set default values and retain them, even if key is
missing from Json response. Not required for primitive data types.

    public Robot{
       version = "";
       name = "";
    }


}
```

Luego, cuando sea necesario realizar un análisis, use lo siguiente:

```
String robotJson = "{
                        \"version\": \"JellyBean\",
                        \"age\": 3,
                        \"robotName\": \"Droid\"
                    }";

Gson gson = new Gson();
Robot robot = gson.fromJson(robotJson, Robot.class);
```

Analizando una lista:

Al recuperar una lista de objetos JSON, a menudo querrá analizarlos y convertirlos en objetos Java.

La cadena JSON que intentaremos convertir es la siguiente:

```
{
  "owned_dogs": [
    {
      "name": "Ron",
      "age": 12,
      "breed": "terrier"
    },
    {
      "name": "Bob",
      "age": 4,
      "breed": "bulldog"
    },
    {
      "name": "Johny",
      "age": 3,
      "breed": "golden retriever"
    }
  ]
}
```

Esta matriz JSON particular contiene tres objetos. En nuestro código Java, querremos asignar estos objetos a objetos `Dog` . Un objeto de perro se vería así:

```
private class Dog {
    public String name;
    public int age;

    @SerializedName("breed")
    public String breedName;
}
```

Para convertir la matriz JSON en un `Dog[]` :

```
Dog[] arrayOfDogs = gson.fromJson(jsonArrayString, Dog[].class);
```

Convertir un `Dog[]` en una cadena JSON:

```
String jsonArray = gson.toJson(arrayOfDogs, Dog[].class);
```

Para convertir la matriz JSON en un `ArrayList<Dog>` podemos hacer lo siguiente:

```
Type typeListOfDogs = new TypeToken<List<Dog>>(){}.getType();
List<Dog> listOfDogs = gson.fromJson(jsonArrayString, typeListOfDogs);
```

El objeto Type `typeListOfDogs` define el aspecto que tendría una lista de objetos `Dog` . GSON puede usar este tipo de objeto para asignar la matriz JSON a los valores correctos.

Alternativamente, la conversión de una `List<Dog>` a una matriz JSON se puede hacer de una manera similar.

```
String jsonArray = gson.toJson(listOfDogs, typeListOfDogs);
```

Analizar la propiedad JSON para enumerar con Gson

Si desea analizar una cadena para enumerar con Gson:

{"estado": "abierto"}

```
public enum Status {
    @SerializedName("open")
    OPEN,
    @SerializedName("waiting")
    WAITING,
    @SerializedName("confirm")
    CONFIRM,
    @SerializedName("ready")
    READY
}
```

Analizar una lista con Gson

Método 1

```
Gson gson = new Gson();
String json = "[ \"Adam\", \"John\", \"Mary\" ]";

Type type = new TypeToken<List<String>>(){}.getType();
List<String> members = gson.fromJson(json, type);
Log.v("Members", members.toString());
```

Esto es útil para la mayoría de las clases de contenedores genéricos, ya que no puede obtener la clase de un tipo parametrizado (es decir, no puede llamar a la `List<String>.class`).

Método 2

```
public class StringList extends ArrayList<String> { }

...

List<String> members = gson.fromJson(json, StringList.class);
```

Alternativamente, siempre puede subclasificar el tipo que desee y luego pasar esa clase. Sin embargo, esto no siempre es la mejor práctica, ya que le devolverá un objeto de tipo `StringList` ;

Serialización / Deserialización JSON con AutoValue y Gson

Importa en tu archivo root de gradle

```
classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
```

Importa en tu aplicación de Gradle

```
apt 'com.google.auto.value:auto-value:1.2'
```

```
apt 'com.ryanharter.auto.value:auto-value-gson:0.3.1'
provided 'com.jakewharton.auto.value:auto-value-annotations:1.2-update1'
provided 'org.glassfish:javax.annotation:10.0-b28'
```

Crear objeto con autovalue:

```
@AutoValue public abstract class SignIn {
    @SerializedName("signin_token") public abstract String signinToken();
    public abstract String username();

    public static TypeAdapter<SignIn> typeAdapter(Gson gson) {
        return new AutoValue_SignIn.GsonTypeAdapter(gson);
    }

    public static SignIn create(String signin, String username) {
        return new AutoValue_SignIn(signin, username);
    }
}
```

Crea tu convertidor Gson con tu GsonBuilder

```
Gson gson = new GsonBuilder()
                .registerTypeAdapterFactory(
                    new AutoValueGsonTypeAdapterFactory())
                .create());
```

Deserializar

```
String myJsonData = "{
    \"signin_token\": \"mySigninToken\",
    \"username\": \"myUsername\" }";
SignIn signInData = gson.fromJson(myJsonData, Signin.class);
```

Publicar por fascículos

```
Signin myData = SignIn.create("myTokenData", "myUsername");
String myJsonData = gson.toJson(myData);
```

Usar Gson es una excelente manera de simplificar el código de serialización y deserialización utilizando objetos POJO. El efecto secundario es que la reflexión es costosa en términos de rendimiento. Es por eso que el uso de AutoValue-Gson para generar CustomTypeAdapter evitará este costo de reflexión y se mantendrá muy simple de actualizar cuando ocurra un cambio de API.

Análisis de JSON a objetos de clase genéricos con Gson

Supongamos que tenemos una cadena JSON:

```
["first","second","third"]
```

Podemos analizar esta cadena JSON en una matriz de `String` :

```
Gson gson = new Gson();
String jsonArray = "[\"first\",\"second\",\"third\"]";
String[] strings = gson.fromJson(jsonArray, String[].class);
```

Pero si queremos analizarlo en un objeto `List<String>` , debemos usar `TypeToken` .

Aquí está la muestra:

```
Gson gson = new Gson();
String jsonArray = "[\"first\",\"second\",\"third\"]";
List<String> stringList = gson.fromJson(jsonArray, new TypeToken<List<String>>()
{}.getType());
```

Supongamos que tenemos dos clases a continuación:

```
public class Outer<T> {
    public int index;
    public T data;
}

public class Person {
    public String firstName;
    public String lastName;
}
```

y tenemos una cadena JSON que debe analizarse a un objeto `Outer<Person>` .

Este ejemplo muestra cómo analizar esta cadena JSON al objeto de clase genérico relacionado:

```
String json = "......";
Type userType = new TypeToken<Outer<Person>>(){}.getType();
Result<User> userResult = gson.fromJson(json,userType);
```

Si la cadena JSON se debe analizar a un objeto `Outer<List<Person>>` :

```
Type userListType = new TypeToken<Outer<List<Person>>>(){}.getType();
Result<List<User>> userListResult = gson.fromJson(json,userListType);
```

Añadiendo Gson a tu proyecto

```
dependencies {
    compile 'com.google.code.gson:gson:2.8.1'
}
```

Para usar la última versión de Gson

La siguiente línea compilará la última versión de gson library cada vez que compile, no tiene que cambiar la versión.

Pros: Puedes usar las últimas funciones, velocidad y menos errores.
Contras: Podría romper la compatibilidad con tu código.

```
compile 'com.google.code.gson:gson:+'
```

Usando Gson para cargar un archivo JSON desde el disco.

Esto cargará un archivo JSON desde el disco y lo convertirá al tipo dado.

```
public static <T> T getFile(String fileName, Class<T> type) throws FileNotFoundException {
    Gson gson = new GsonBuilder()
            .create();
    FileReader json = new FileReader(fileName);
    return gson.fromJson(json, type);
}
```

Agregar un convertidor personalizado a Gson

A veces necesita serializar o deserializar algunos campos en un formato deseado, por ejemplo, su backend puede usar el formato "YYYY-MM-dd HH: mm" para las fechas y desea que su POJOS use la clase DateTime en Joda Time.

Para convertir automáticamente estas cadenas en el objeto DateTimes, puede usar un convertidor personalizado.

```
/**
 * Gson serialiser/deserialiser for converting Joda {@link DateTime} objects.
 */
public class DateTimeConverter implements JsonSerializer<DateTime>, JsonDeserializer<DateTime>
{

    private final DateTimeFormatter dateTimeFormatter;

    @Inject
    public DateTimeConverter() {
        this.dateTimeFormatter = DateTimeFormat.forPattern("YYYY-MM-dd HH:mm");
    }

    @Override
    public JsonElement serialize(DateTime src, Type typeOfSrc, JsonSerializationContext
context) {
        return new JsonPrimitive(dateTimeFormatter.print(src));
    }

    @Override
    public DateTime deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext
context)
            throws JsonParseException {

        if (json.getAsString() == null || json.getAsString().isEmpty()) {
            return null;
        }

        return dateTimeFormatter.parseDateTime(json.getAsString());
    }
}
```

Para hacer que Gson use el convertidor recién creado, debe asignarlo al crear el objeto Gson:

```
    DateTimeConverter dateTimeConverter = new DateTimeConverter();
    Gson gson = new GsonBuilder().registerTypeAdapter(DateTime.class, dateTimeConverter)
            .create();

    String s = gson.toJson(DateTime.now());
    // this will show the date in the desired format
```

Para deserializar la fecha en ese formato, solo tiene que definir un campo en el formato DateTime:

```
public class SomePojo {
    private DateTime someDate;
}
```

Cuando Gson encuentra un campo de tipo DateTime, llamará a su convertidor para deserializar el campo.

Usando Gson como serializador con Retrofit

En primer lugar, debe agregar `GsonConverterFactory` a su archivo build.gradle

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

Luego, debe agregar la fábrica de convertidores al crear el Servicio de actualización:

```
Gson gson = new GsonBuilder().create();
new Retrofit.Builder()
        .baseUrl(someUrl)
        .addConverterFactory(GsonConverterFactory.create(gson))
        .build()
        .create(RetrofitService.class);
```

Puede agregar convertidores personalizados al crear el objeto Gson que está pasando a la fábrica. Le permite crear conversiones de tipo personalizado.

Analizar la matriz json a una clase genérica usando Gson

Supongamos que tenemos un json:

```
{
  "total_count": 132,
  "page_size": 2,
  "page_index": 1,
  "twitter_posts": [
    {
      "created_on": 1465935152,
      "tweet_id": 210462857140252672,
      "tweet": "Along with our new #Twitterbird, we've also updated our Display Guidelines",
      "url": "https://twitter.com/twitterapi/status/210462857140252672"
    },
    {
      "created_on": 1465995741,
```

```
      "tweet_id": 735128881808691200,
      "tweet": "Information on the upcoming changes to Tweets is now on the developer site",
      "url": "https://twitter.com/twitterapi/status/735128881808691200"
    }
  ]
}
```

Podemos analizar esta matriz en un objeto de Tweets personalizados (contenedor de listas de tweets) manualmente, pero es más fácil hacerlo con el método `fromJson` :

```
Gson gson = new Gson();
String jsonArray = "....";
Tweets tweets = gson.fromJson(jsonArray, Tweets.class);
```

Supongamos que tenemos dos clases a continuación:

```
class Tweets {
    @SerializedName("total_count")
    int totalCount;
    @SerializedName("page_size")
    int pageSize;
    @SerializedName("page_index")
    int pageIndex;
    // all you need to do it is just define List variable with correct name
    @SerializedName("twitter_posts")
    List<Tweet> tweets;
}

class Tweet {
    @SerializedName("created_on")
    long createdOn;
    @SerializedName("tweet_id")
    String tweetId;
    @SerializedName("tweet")
    String tweetBody;
    @SerializedName("url")
    String url;
}
```

y si solo necesita analizar una matriz json, puede usar este código en su análisis:

```
String tweetsJsonArray = "[{.....},{.....}]"
List<Tweet> tweets = gson.fromJson(tweetsJsonArray, new TypeToken<List<Tweet>>()
{}.getType());
```

Deserializador JSON personalizado utilizando Gson

Imagine que tiene todas las fechas en todas las respuestas en algún formato personalizado, por ejemplo `/Date(1465935152)/` y desea aplicar la regla general para deserializar todas las fechas Json a las instancias de `Date` Java. En este caso, debe implementar Json Deserializer personalizado.

Ejemplo de json:

```
{
  "id": 1,
  "created_on": "Date(1465935152)",
  "updated_on": "Date(1465968945)",
  "name": "Oleksandr"
}
```

Supongamos que tenemos esta clase a continuación:

```
class User {
    @SerializedName("id")
    long id;
    @SerializedName("created_on")
    Date createdOn;
    @SerializedName("updated_on")
    Date updatedOn;
    @SerializedName("name")
    String name;
}
```

Deserializador personalizado:

```
class DateDeSerializer implements JsonDeserializer<Date> {
    private static final String DATE_PREFIX = "/Date(";
    private static final String DATE_SUFFIX = ")/";

    @Override
    public Date deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext
context) throws JsonParseException {
        String dateString = json.getAsString();
        if (dateString.startsWith(DATE_PREFIX) && dateString.endsWith(DATE_SUFFIX)) {
            dateString = dateString.substring(DATE_PREFIX.length(), dateString.length() -
DATE_SUFFIX.length());
        } else {
            throw new JsonParseException("Wrong date format: " + dateString);
        }
        return new Date(Long.parseLong(dateString) - TimeZone.getDefault().getRawOffset());
    }
}
```

Y el uso:

```
Gson gson = new GsonBuilder()
                .registerTypeAdapter(Date.class, new DateDeSerializer())
                .create();
String json = "....";
User user = gson.fromJson(json, User.class);
```

Serializar y deserializar cadenas Jackson JSON con tipos de fecha

Esto también se aplica al caso en el que desea que la conversión de Gson Date sea compatible con Jackson, por ejemplo.

Jackson usualmente serializa Date a "milisegundos desde la época", mientras que Gson usa un formato legible como el `Aug 31, 2016 10:26:17` para representar la fecha. Esto lleva a

JsonSyntaxExceptions en Gson cuando intenta deserializar una fecha de formato Jackson.

Para evitar esto, puede agregar un serializador personalizado y un deserializador personalizado:

```
JsonSerializer<Date> ser = new JsonSerializer<Date>() {
    @Override
    public JsonElement serialize(Date src, Type typeOfSrc, JsonSerializationContext
                context) {
        return src == null ? null : new JsonPrimitive(src.getTime());
    }
};

JsonDeserializer<Date> deser = new JsonDeserializer<Date>() {
    @Override
    public Date deserialize(JsonElement json, Type typeOfT,
                JsonDeserializationContext context) throws JsonParseException {
        return json == null ? null : new Date(json.getAsLong());
    }
};

Gson gson = new GsonBuilder()
                .registerTypeAdapter(Date.class, ser)
                .registerTypeAdapter(Date.class, deser)
                .create();
```

Usando Gson con herencia

Gson no admite herencia fuera de la caja.

Digamos que tenemos la siguiente jerarquía de clases:

```
public class BaseClass {
    int a;

    public int getInt() {
        return a;
   }
}

public class DerivedClass1 extends BaseClass {
     int b;

     @Override
     public int getInt() {
         return b;
     }
 }

public class DerivedClass2 extends BaseClass {
    int c;

    @Override
    public int getInt() {
        return c;
    }
}
```

Y ahora queremos serializar una instancia de `DerivedClass1` a una cadena JSON

```
DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;

Gson gson = new Gson();
String derivedClass1Json = gson.toJson(derivedClass1);
```

Ahora, en otro lugar, recibimos esta cadena json y queremos deserializarla, pero en tiempo de compilación solo sabemos que se supone que es una instancia de `BaseClass` :

```
BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());
```

Pero GSON no sabe que `derivedClass1Json` fue originalmente una instancia de `DerivedClass1` , por lo que esto imprimirá 10.

¿Cómo resolver esto?

Necesita crear su propio `JsonDeserializer` , que maneja tales casos. La solución no está perfectamente limpia, pero no pude encontrar una mejor.

Primero, agregue el siguiente campo a su clase base

```
@SerializedName("type")
private String typeName;
```

E inicialízalo en el constructor de la clase base.

```
public BaseClass() {
    typeName = getClass().getName();
}
```

Ahora agregue la siguiente clase:

```
public class JsonDeserializerWithInheritance<T> implements JsonDeserializer<T> {

 @Override
 public T deserialize(
     JsonElement json, Type typeOfT, JsonDeserializationContext context)
     throws JsonParseException {
     JsonObject jsonObject = json.getAsJsonObject();
     JsonPrimitive classNamePrimitive = (JsonPrimitive) jsonObject.get("type");

     String className = classNamePrimitive.getAsString();

     Class<?> clazz;
     try {
     clazz = Class.forName(className);
     } catch (ClassNotFoundException e) {
     throw new JsonParseException(e.getMessage());
     }
```

```
    return context.deserialize(jsonObject, clazz);
 }
}
```

Todo lo que queda por hacer es conectar todo

```
GsonBuilder builder = new GsonBuilder();
 builder
 .registerTypeAdapter(BaseClass.class, new JsonDeserializerWithInheritance<BaseClass>());
 Gson gson = builder.create();
```

Y ahora, ejecutando el siguiente código-

```
 DerivedClass1 derivedClass1 = new DerivedClass1();
 derivedClass1.b = 5;
 derivedClass1.a = 10;
 String derivedClass1Json = gson.toJson(derivedClass1);

 BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
 System.out.println(maybeDerivedClass1.getInt());
```

Se imprimirá 5.

Lea Gson en línea: https://riptutorial.com/es/android/topic/4158/gson

Capítulo 124: Herramientas Atributos

Observaciones

Android tiene un espacio de nombres XML dedicado destinado a las herramientas para poder registrar información en un archivo XML.

El URI del espacio de nombres es:

`http://schemas.android.com/tools` y generalmente está vinculado a las `tools:` prefijo.

Examples

Atributos de diseño en tiempo de diseño

Estos atributos se utilizan cuando el diseño se representa en Android Studio, pero no tienen impacto en el tiempo de ejecución.

En general, puede usar cualquier atributo del marco de Android, simplemente utilizando las `tools:` espacio de nombres en lugar de `android:` espacio de nombres para la vista previa del diseño. Puede agregar el atributo `android:` namespace (que se usa en el tiempo de ejecución) y las `tools:` correspondientes `tools:` atributo (que anula el atributo de ejecución en la vista previa del diseño solamente).

Simplemente defina el espacio de nombres de las herramientas como se describe en la sección de comentarios.

Por ejemplo el atributo de `text` :

```
<EditText
      tools:text="My Text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
```

O el atributo de `visibility` para desactivar una vista para vista previa:

```
<LinearLayout
        android:id="@+id/ll1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:visibility="gone" />
```

O el atributo de `context` para asociar el diseño con actividad o fragmento.

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity" >
```

O el atributo `showIn` para ver e incluir una vista previa del diseño en otro diseño

```
<EditText xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/text"
    tools:showIn="@layout/activity_main" />
```

Lea Herramientas Atributos en línea: https://riptutorial.com/es/android/topic/1676/herramientas-atributos

Capítulo 125: Herramientas de informes de bloqueo

Observaciones

El mejor wiki completo está disponible aquí en github .

Examples

Tejido - Crashlytics

Fabric es una plataforma móvil modular que proporciona kits útiles que puede mezclar para construir su aplicación. **Crashlytics** es una herramienta de informe de **fallas** y problemas provista por Fabric que le permite hacer un seguimiento y monitorear sus aplicaciones en detalle.

Cómo configurar Fabric-Crashlytics

Paso 1: Cambia tu `build.gradle` :

Agrega el plugin repo y el plugin gradle:

```
 buildscript {
  repositories {
    maven { url 'https://maven.fabric.io/public' }
  }

  dependencies {
    // The Fabric Gradle plugin uses an open ended version to react
    // quickly to Android tooling updates
    classpath 'io.fabric.tools:gradle:1.+'
  }
}
```

Aplicar el plugin:

```
apply plugin: 'com.android.application'
//Put Fabric plugin after Android plugin
apply plugin: 'io.fabric'
```

Añadir el repositorio de tela:

```
repositories {
  maven { url 'https://maven.fabric.io/public' }
}
```

Agregue el kit Crashlyrics:

```
dependencies {

  compile('com.crashlytics.sdk.android:crashlytics:2.6.6@aar') {
    transitive = true;
  }
}
```

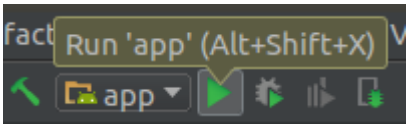
Paso 2: agregue su **clave de API** y el permiso de **INTERNET** en `AndroidManifest.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
  <application
     ... >

      <meta-data
          android:name="io.fabric.ApiKey"
          android:value="25eeca3bb31cd41577e097cabd1ab9eee9da151d"
          />

  </application>

  <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

Paso 3: inicie el kit en tiempo de ejecución en su código, por ejemplo:

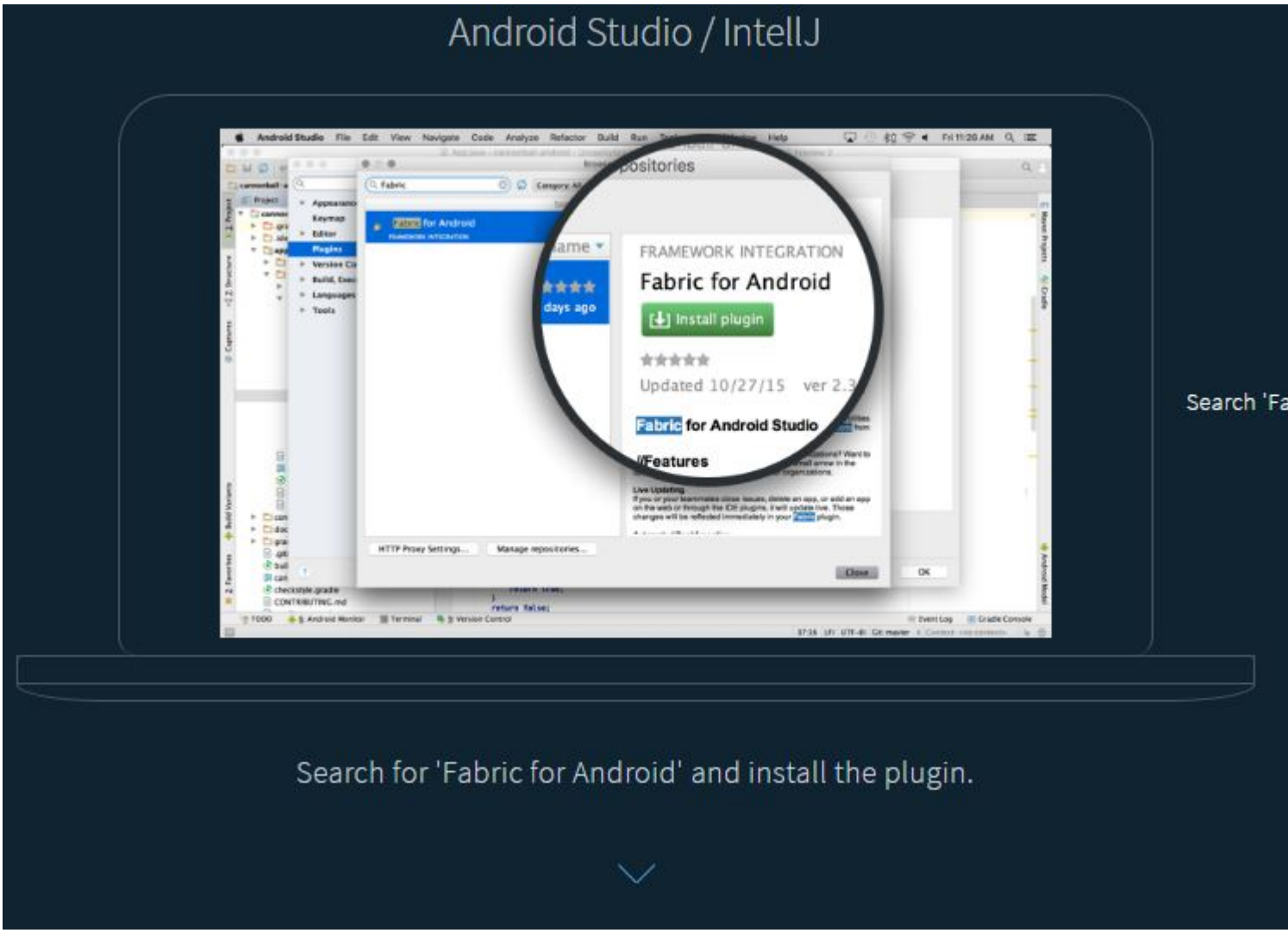
```
public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      //Init the KIT
      Fabric.with(this, new Crashlytics());

      setContentView(R.layout.activity_main);
    }
}
```

Paso 4: Proyecto de construcción. Para construir y ejecutar:



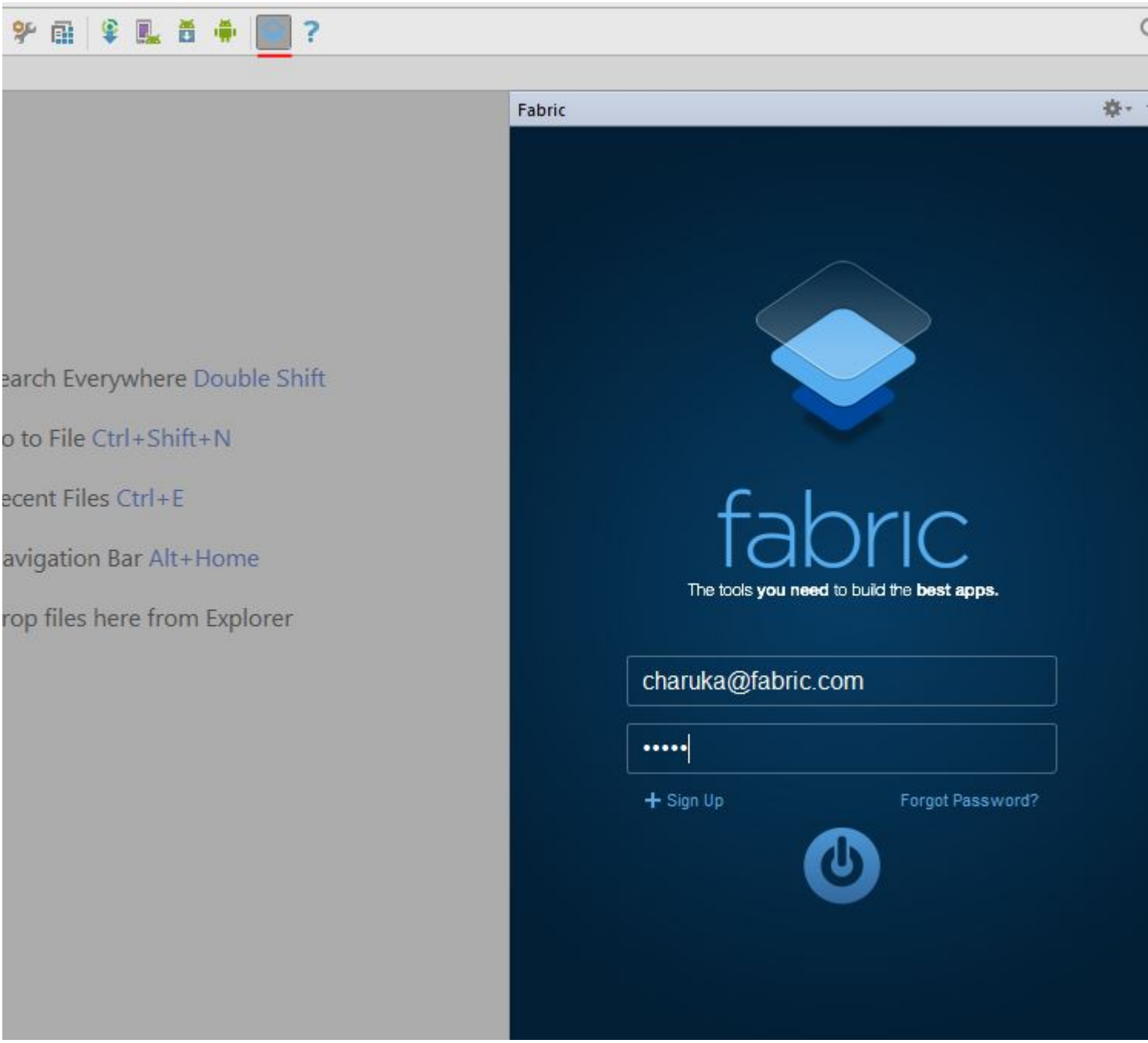
Usando el plugin IDE de tela

Los kits se pueden instalar utilizando el complemento Fabric IDE para Android Studio o IntelliJ siguiendo este enlace.



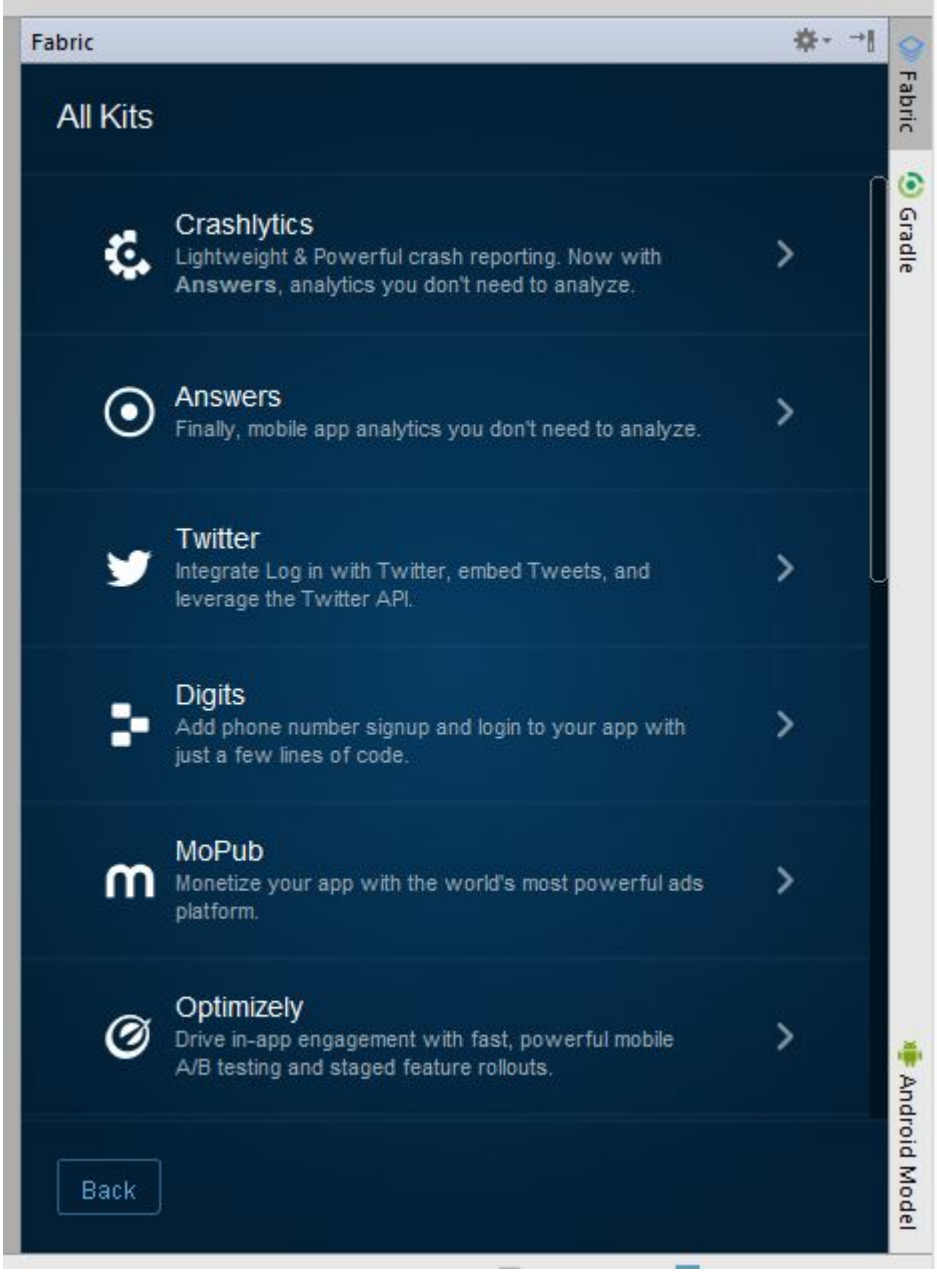
Después de instalar el complemento, **reinicie** Android Studio e **inicie sesión** con su cuenta utilizando **Android Studio** .

(tecla corta> `CTRL + L`)

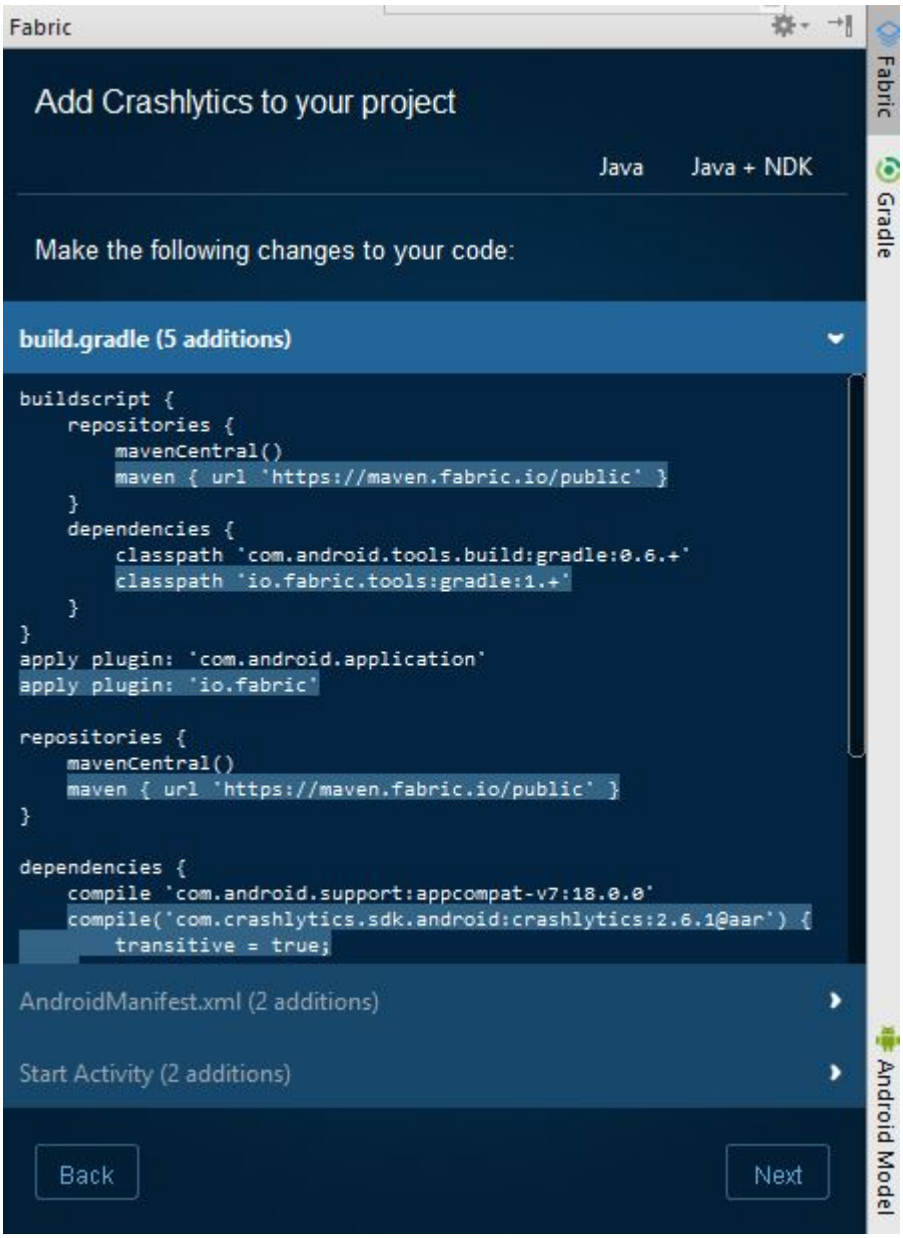


Luego mostrará los proyectos que tiene / el proyecto que abrió, seleccione el que necesita y haga clic en siguiente ... a continuación.

Seleccione el kit que desea agregar, para su ejemplo es **Crashlytics** :



Luego presione `Install` . No es necesario que lo agregue manualmente esta vez, como el **complemento gradle** anterior, en su lugar, se construirá para usted.



¡Hecho!

Informe de Accidentes con ACRA

Paso 1: Agregue la dependencia del último ACRA AAR a su aplicación gradle (build.gradle).

Paso 2: en su clase de aplicación (la clase que extiende la aplicación; si no la crea) Agregue una anotación `@ReportsCrashes` y anule el método `attachBaseContext()` .

Paso 3: Inicialice la clase ACRA en su clase de aplicación

```
@ReportsCrashes(
    formUri = "Your choice of backend",
    reportType = REPORT_TYPES(JSON/FORM),
    httpMethod = HTTP_METHOD(POST/PUT),
    formUriBasicAuthLogin = "AUTH_USERNAME",
    formUriBasicAuthPassword = "AUTH_PASSWORD,
    customReportContent = {
            ReportField.USER_APP_START_DATE,
```

```
            ReportField.USER_CRASH_DATE,
            ReportField.APP_VERSION_CODE,
            ReportField.APP_VERSION_NAME,
            ReportField.ANDROID_VERSION,
            ReportField.DEVICE_ID,
            ReportField.BUILD,
            ReportField.BRAND,
            ReportField.DEVICE_FEATURES,
            ReportField.PACKAGE_NAME,
            ReportField.REPORT_ID,
            ReportField.STACK_TRACE,
    },
    mode = NOTIFICATION_TYPE(TOAST,DIALOG,NOTIFICATION)
    resToastText = R.string.crash_text_toast)

    public class MyApplication extends Application {
     @Override
            protected void attachBaseContext(Context base) {
                super.attachBaseContext(base);
                // Initialization of ACRA
                ACRA.init(this);
            }
    }
```

Donde AUTH_USERNAME y AUTH_PASSWORD son las credenciales de tus backends deseados.

Paso 4: definir la clase de aplicación en AndroidManifest.xml

```
 <application
        android:name=".MyApplication">
        <service></service>
        <activity></activity>
        <receiver></receiver>
 </application>
```

Paso 5: asegúrese de tener permiso de `internet` para recibir el informe de la aplicación bloqueada

```
<uses-permission android:name="android.permission.INTERNET"/>
```

En caso de que desee enviar el informe silencioso al backend, simplemente use el método a continuación para lograrlo.

```
ACRA.getErrorReporter().handleSilentException(e);
```

Forzar un choque de prueba con tela

Agregue un botón que puede tocar para desencadenar un bloqueo. Pegue este código en su diseño donde le gustaría que aparezca el botón.

```
<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="Force Crash!"
```

```
    android:onClick="forceCrash"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true" />
```

Lanzar un RuntimeException

```
public void forceCrash(View view) {
    throw new RuntimeException("This is a crash");
}
```

Ejecute su aplicación y toque el nuevo botón para provocar un bloqueo. En un minuto o dos, debería poder ver el bloqueo en su panel de control de Crashlytics y recibir un correo.

Captura bloqueos utilizando Sherlock

Sherlock captura todos tus choques y los informa como una notificación. Cuando toca la notificación, se abre una actividad con todos los detalles del fallo junto con la información del dispositivo y la aplicación

¿Cómo integrar Sherlock con tu aplicación?

Solo necesita agregar Sherlock como una dependencia de Gradle en su proyecto.

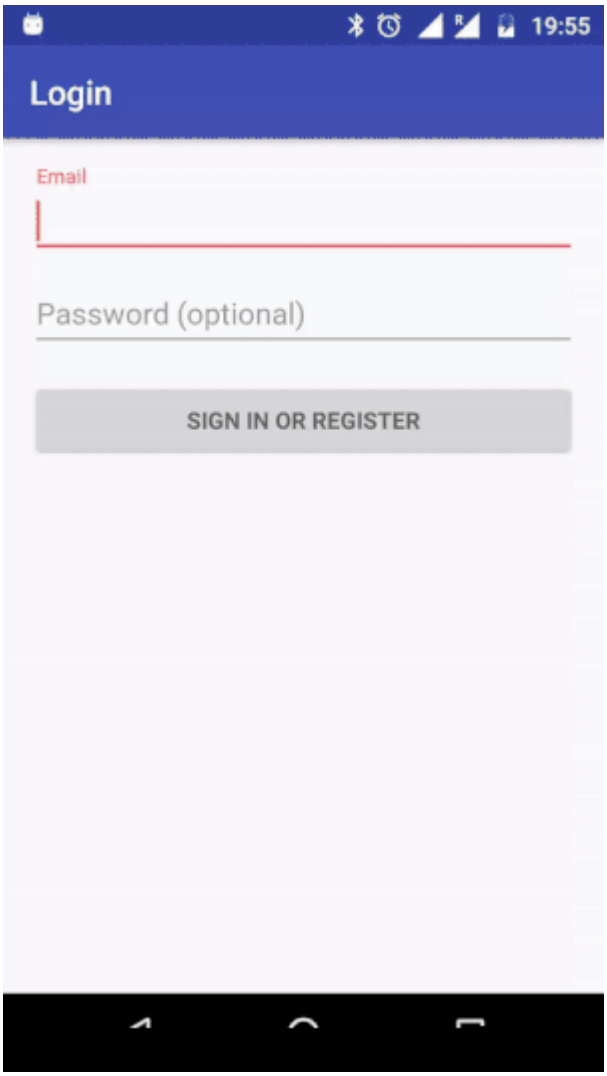
```
dependencies {
    compile('com.github.ajitsing:sherlock:1.0.1@aar') {
        transitive = true
    }
}
```

Después de sincronizar tu estudio de Android, inicializa Sherlock en tu clase de Aplicación.

```
package com.singhajit.login;

import android.app.Application;

import com.singhajit.sherlock.core.Sherlock;

public class SampleApp extends Application {
  @Override
  public void onCreate() {
    super.onCreate();
    Sherlock.init(this);
  }
}
```

Eso es todo lo que necesitas hacer. También Sherlock hace mucho más que informar un accidente. Para ver todas sus características, eche un vistazo a este artículo .

Manifestación



Lea Herramientas de informes de bloqueo en línea:
https://riptutorial.com/es/android/topic/3871/herramientas-de-informes-de-bloqueo

Capítulo 126: Hilandero

Examples

Añadiendo un spinner a tu actividad.

En /res/values/strings.xml:

```
<string-array name="spinner_options">
    <item>Option 1</item>
    <item>Option 2</item>
    <item>Option 3</item>
</string-array>
```

En maquetación XML:

```
<Spinner
    android:id="@+id/spinnerName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_options" />
```

En Actividad:

```
Spinner spinnerName = (Spinner) findViewById(R.id.spinnerName);
spinnerName.setOnItemSelectedListener(new OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
         String chosenOption = (String) parent.getItemAtPosition(position);
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {}
});
```

Ejemplo de Spinner básico

Spinner Es un tipo de entrada desplegable. En primer lugar en el diseño

```
  <Spinner
    android:id="@+id/spinner"     <!-- id to refer this spinner from JAVA-->
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

</Spinner>
```

Ahora, en segundo lugar, rellene los valores en el selector. Existen principalmente dos formas de completar los valores en el `spinner` .

1. Desde el mismo XML, cree un **array.xml** en el directorio de **valores** bajo **res** . Crear esta `array`

```
  <string-array name="defaultValue">
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
</string-array>
```

Ahora agregue esta línea en sppiner XML

```
            android:entries="@array/defaultValue"
```

2. También puede agregar valores a través de JAVA

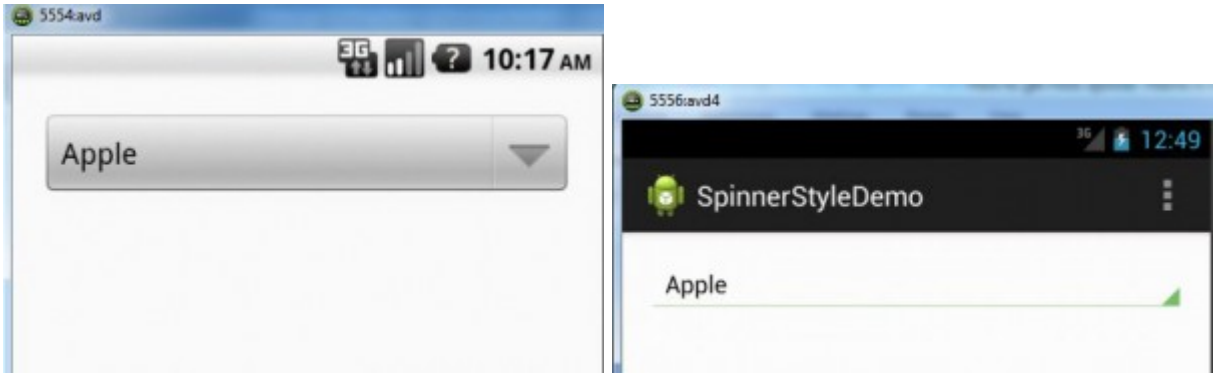
si está utilizando en la `activity` cityArea = (Spinner) findViewById (R.id.cityArea); más si está utilizando en el `fragment`

```
cityArea = (Spinner) findViewById(R.id.cityArea);
```

Ahora crea una `arrayList` de `Strings`

```
ArrayList<String> area = new ArrayList<>();
//add values in area arrayList
cityArea.setAdapter(new ArrayAdapter<String>(context
                            , android.R.layout.simple_list_item_1, area));
```

Esto se verá como



De acuerdo a la versión del dispositivo Android, se renderizará estilo.

Los siguientes son algunos de los temas por defecto

Si una aplicación no solicita explícitamente un tema en su manifiesto, el Sistema Android determinará el tema predeterminado basado en el targetSdkVersion de la aplicación para mantener las expectativas originales de la aplicación:

Versión de Android SDK	Tema predeterminado
Versión <11	@android: estilo / tema
Versión entre 11 y 13.	@android: style / Theme.Holo

Versión de Android SDK	Tema predeterminado
14 y más alto	@android: style / Theme.DeviceDefault

`Spinner` se puede personalizar fácilmente con la ayuda de XML, por ejemplo

```
 android:background="@drawable/spinner_background"

 android:layout_margin="16dp"

 android:padding="16dp"
```

Crea un fondo personalizado en XML y úsalo.

Obtenga fácilmente la posición y otros detalles del elemento seleccionado en el girador

```
cityArea.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            areaNo = position;
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {

        }
    });
```

Cambia el color del texto del elemento seleccionado en spinner.

Esto se puede hacer de dos maneras en XML

```
<item android:state_activated="true" android:color="@color/red"/>
```

Esto cambiará el color del elemento seleccionado en la ventana emergente.

y desde JAVA haga esto (en el setOnItemSelectedListener (...))

```
 @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long
id) {
                 ((TextView) parent.getChildAt(0)).setTextColor(0x00000000);
// similarly change `background color` etc.
            }
```

Lea Hilandero en línea: https://riptutorial.com/es/android/topic/3459/hilandero

Capítulo 127: Hilo

Examples

Ejemplo de hilo con su descripción

Al iniciar una aplicación se ejecuta primer hilo principal. Este hilo principal maneja todo el concepto de UI de la aplicación. Si queremos ejecutar durante mucho tiempo la tarea en la que no necesitamos la interfaz de usuario, usamos un hilo para ejecutar esa tarea en segundo plano.

Aquí está el ejemplo de Hilo que describe golpe:

```
new Thread(new Runnable() {
    public void run() {
        for(int i = 1; i < 5;i++) {
            System.out.println(i);
        }
    }
}).start();
```

Podemos crear un hilo creando el objeto de Hilo que tiene el método `Thread.run()` para ejecutar el hilo. Aquí, el método `start()` llama al método `run()` .

También podemos ejecutar los múltiples subprocesos de forma independiente, lo que se conoce como MultiThreading. Este subproceso también tiene la funcionalidad de suspensión por la cual el subproceso que se está ejecutando actualmente está en suspensión (detener temporalmente la ejecución) durante el número de tiempo especificado. Pero sleep lanza la InterruptedException Por lo tanto, tenemos que manejarlo usando try / catch como este.

```
try{Thread.sleep(500);}catch(InterruptedException e){System.out.println(e);}
```

Actualización de la interfaz de usuario desde un hilo de fondo

Es común utilizar un subproceso de fondo para realizar operaciones de red o tareas de larga ejecución, y luego actualizar la interfaz de usuario con los resultados cuando sea necesario.

Esto plantea un problema, ya que solo el hilo principal puede actualizar la interfaz de usuario.

La solución es usar el método `runOnUiThread()` , ya que le permite iniciar la ejecución de código en el subproceso de la interfaz de usuario desde un subproceso de fondo.

En este sencillo ejemplo, un Thread se inicia cuando se crea la Actividad, se ejecuta hasta que el número mágico de `42` se genera aleatoriamente, y luego utiliza el método `runOnUiThread()` para actualizar la interfaz de usuario una vez que se cumple esta condición.

```
public class MainActivity extends AppCompatActivity {
```

```
    TextView mTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mTextView = (TextView) findViewById(R.id.my_text_view);

        new Thread(new Runnable() {
            @Override
            public void run() {
                while (true) {
                    //do stuff....
                    Random r = new Random();
                    if (r.nextInt(100) == 42) {
                       break;
                    }
                }

                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        mTextView.setText("Ready Player One");
                    }
                });
            }
        }).start();
    }
}
```

Lea Hilo en línea: https://riptutorial.com/es/android/topic/7131/hilo

Capítulo 128: Hojas inferiores

Introducción

Una hoja inferior es una hoja que se desliza hacia arriba desde el borde inferior de la pantalla.

Observaciones

Las hojas inferiores se deslizan hacia arriba desde la parte inferior de la pantalla para revelar más contenido.
Se agregaron a la biblioteca de soporte de Android en la versión v23.2.0.

Examples

BottomSheetBehavior como los mapas de Google

2.1.x

Este ejemplo depende de la biblioteca de soporte 23.4.0. +.

BottomSheetBehavior se caracteriza por:

1. Dos barras de herramientas con animaciones que responden a los movimientos de la hoja inferior.
2. Un FAB que se oculta cuando está cerca de la "barra de herramientas modal" (la que aparece cuando se desliza hacia arriba).
3. Una imagen de fondo detrás de la hoja inferior con algún tipo de efecto de paralaje.
4. Un título (TextView) en la barra de herramientas que aparece cuando la hoja inferior lo alcanza.
5. La barra de notificación satus puede convertir su fondo a transparente o a todo color.
6. Un comportamiento personalizado de la hoja inferior con un estado "ancla".

Ahora vamos a revisarlos uno por uno:

Barras de herramientas
Cuando abres esa vista en Google Maps, puedes ver una barra de herramientas en la que puedes buscar, es la única que no estoy haciendo exactamente como Google Maps, porque quería hacerlo más genérico. De todos modos, la `AppBarLayout ToolBar` está dentro de un `AppBarLayout` y se ocultó cuando comenzó a arrastrar la hoja inferior y aparece de nuevo cuando la hoja inferior llega al estado `COLLAPSED` .
Para lograrlo necesitas:

- cree un `Behavior` y `AppBarLayout.ScrollingViewBehavior` desde

`AppBarLayout.ScrollingViewBehavior`

- anular los métodos `layoutDependsOn` y `onDependentViewChanged` . Al hacerlo escucharás movimientos de la parte inferior.
- cree algunos métodos para ocultar y mostrar la barra de herramientas AppBarLayout / ToolBar.

Así es como lo hice para la primera barra de herramientas o ActionBar:

```
@Override
public boolean layoutDependsOn(CoordinatorLayout parent, View child, View dependency) {
    return dependency instanceof NestedScrollView;
}

@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, View child,
                                      View dependency) {

    if (mChild == null) {
        initValues(child, dependency);
        return false;
    }

    float dVerticalScroll = dependency.getY() - mPreviousY;
    mPreviousY = dependency.getY();

    //going up
    if (dVerticalScroll <= 0 && !hidden) {
        dismissAppBar(child);
        return true;
    }

    return false;
}

private void initValues(final View child, View dependency) {

    mChild = child;
    mInitialY = child.getY();

    BottomSheetBehaviorGoogleMapsLike bottomSheetBehavior =
BottomSheetBehaviorGoogleMapsLike.from(dependency);
    bottomSheetBehavior.addBottomSheetCallback(new
BottomSheetBehaviorGoogleMapsLike.BottomSheetCallback() {
        @Override
        public void onStateChanged(@NonNull View bottomSheet,
@BottomSheetBehaviorGoogleMapsLike.State int newState) {
            if (newState == BottomSheetBehaviorGoogleMapsLike.STATE_COLLAPSED ||
                    newState == BottomSheetBehaviorGoogleMapsLike.STATE_HIDDEN)
                showAppBar(child);
        }

        @Override
        public void onSlide(@NonNull View bottomSheet, float slideOffset) {

        }
    });
}

private void dismissAppBar(View child){
```

```
    hidden = true;
    AppBarLayout appBarLayout = (AppBarLayout)child;
    mToolbarAnimation =
appBarLayout.animate().setDuration(mContext.getResources().getInteger(android.R.integer.config_shortAni

    mToolbarAnimation.y(-(mChild.getHeight()+25)).start();
}

private void showAppBar(View child) {
    hidden = false;
    AppBarLayout appBarLayout = (AppBarLayout)child;
    mToolbarAnimation =
appBarLayout.animate().setDuration(mContext.getResources().getInteger(android.R.integer.config_mediumAn

    mToolbarAnimation.y(mInitialY).start();
}
```

Aquí está el archivo completo si lo necesitas

La segunda barra de herramientas o barra de herramientas "Modal":
Debe anular los mismos métodos, pero en este tiene que cuidar más conductas:

- mostrar / ocultar la barra de herramientas con animaciones
- Cambiar color de barra de estado / fondo
- mostrar / ocultar el título de la hoja de fondo en la barra de herramientas
- cerrar la parte inferior de la hoja o enviarla al estado contraído

El código para este es un poco extenso, así que dejaré el enlace

El fab

Este también es un comportamiento personalizado, pero se extiende desde `FloatingActionButton.Behavior` . En `onDependentViewChanged` , debe mirar cuando llegue a "offSet" o señalar dónde desea ocultarlo. En mi caso, quiero ocultarlo cuando está cerca de la segunda barra de herramientas, así que entro en el FAB padre (un CoordinatorLayout) buscando el AppBarLayout que contiene la ToolBar, luego uso la posición ToolBar como `OffSet` :

```
@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, FloatingActionButton child,
View dependency) {

    if (offset == 0)
        setOffsetValue(parent);

    if (dependency.getY() <=0)
        return false;

    if (child.getY() <= (offset + child.getHeight()) && child.getVisibility() == View.VISIBLE)
        child.hide();
    else if (child.getY() > offset && child.getVisibility() != View.VISIBLE)
        child.show();

    return false;
```

```
}
```

Completa enlace de comportamiento FAB personalizado

La imagen detrás del BottomSheet con efecto de paralaje :
Al igual que los demás, es un comportamiento personalizado, lo único "complicado" en este es el pequeño algoritmo que mantiene la imagen anclada en la Hoja Inferior y evita el colapso de la imagen como el efecto de paralaje predeterminado:

```
@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, View child,
                                      View dependency) {

    if (mYmultiplier == 0) {
        initValues(child, dependency);
        return true;
    }

    float dVerticalScroll = dependency.getY() - mPreviousY;
    mPreviousY = dependency.getY();

    //going up
    if (dVerticalScroll <= 0 && child.getY() <= 0) {
        child.setY(0);
        return true;
    }

    //going down
    if (dVerticalScroll >= 0 && dependency.getY() <= mImageHeight)
        return false;

    child.setY( (int)(child.getY() + (dVerticalScroll * mYmultiplier) ) );

    return true;
}
```

El archivo completo para la imagen de fondo con efecto de paralaje.

Ahora, para el final: **el comportamiento personalizado de BottomSheet**
Para alcanzar los 3 pasos, primero que hay que entender que por defecto BottomSheetBehavior tiene 5 estados: `STATE_DRAGGING`, `STATE_SETTLING`, `STATE_EXPANDED`, `STATE_COLLAPSED`, `STATE_HIDDEN` y para el comportamiento de Google Maps es necesario agregar un estado intermedio entre colapsado y ampliado: `STATE_ANCHOR_POINT` .
Intenté extender el comportamiento predeterminado de bottomSheetBehavior sin éxito, así que simplemente copié todo el código y modifiqué lo que necesito.
Para lograr lo que estoy hablando, siga los siguientes pasos:

1. Cree una clase de Java y extiéndala desde `CoordinatorLayout.Behavior<V>`

2. Copie el código de `BottomSheetBehavior` archivo de `BottomSheetBehavior` predeterminado a su nuevo.

3. Modifique el método `clampViewPositionVertical` con el siguiente código:

```
@Override
public int clampViewPositionVertical(View child, int top, int dy) {
    return constrain(top, mMinOffset, mHideable ? mParentHeight : mMaxOffset);
}
int constrain(int amount, int low, int high) {
    return amount < low ? low : (amount > high ? high : amount);
}
```

4. Añadir un nuevo estado

   public static final int STATE_ANCHOR_POINT = X;

5. Modifique los siguientes métodos: `onLayoutChild` , `onStopNestedScroll` , `BottomSheetBehavior<V> from(V view)` y `setState` (opcional)

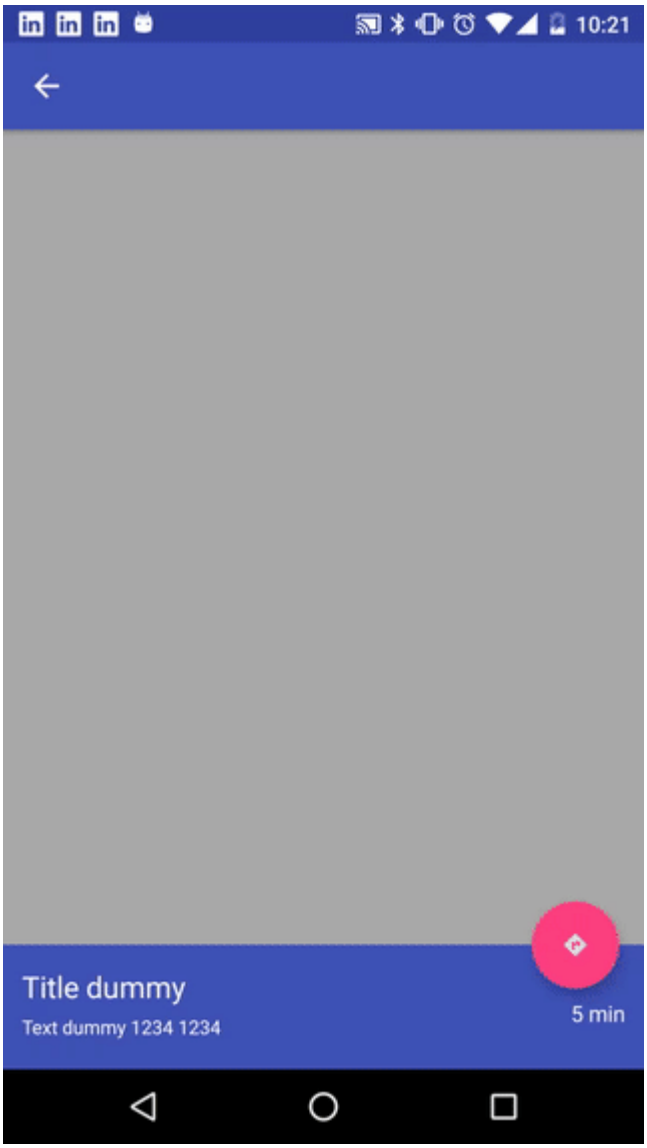
```
public boolean onLayoutChild(CoordinatorLayout parent, V child, int layoutDirection) {
    // First let the parent lay it out
    if (mState != STATE_DRAGGING && mState != STATE_SETTLING) {
        if (ViewCompat.getFitsSystemWindows(parent) &&
                !ViewCompat.getFitsSystemWindows(child)) {
            ViewCompat.setFitsSystemWindows(child, true);
        }
        parent.onLayoutChild(child, layoutDirection);
    }
    // Offset the bottom sheet
    mParentHeight = parent.getHeight();
    mMinOffset = Math.max(0, mParentHeight - child.getHeight());
    mMaxOffset = Math.max(mParentHeight - mPeekHeight, mMinOffset);

    //if (mState == STATE_EXPANDED) {
    //    ViewCompat.offsetTopAndBottom(child, mMinOffset);
    //} else if (mHideable && mState == STATE_HIDDEN...
    if (mState == STATE_ANCHOR_POINT) {
        ViewCompat.offsetTopAndBottom(child, mAnchorPoint);
    } else if (mState == STATE_EXPANDED) {
        ViewCompat.offsetTopAndBottom(child, mMinOffset);
    } else if (mHideable && mState == STATE_HIDDEN) {
        ViewCompat.offsetTopAndBottom(child, mParentHeight);
    } else if (mState == STATE_COLLAPSED) {
        ViewCompat.offsetTopAndBottom(child, mMaxOffset);
    }
    if (mViewDragHelper == null) {
        mViewDragHelper = ViewDragHelper.create(parent, mDragCallback);
    }
    mViewRef = new WeakReference<>(child);
    mNestedScrollingChildRef = new WeakReference<>(findScrollingChild(child));
    return true;
}
```

```
public void onStopNestedScroll(CoordinatorLayout coordinatorLayout, V child, View target) {
    if (child.getTop() == mMinOffset) {
        setStateInternal(STATE_EXPANDED);
        return;
    }
    if (target != mNestedScrollingChildRef.get() || !mNestedScrolled) {
        return;
    }
    int top;
    int targetState;
    if (mLastNestedScrollDy > 0) {
        //top = mMinOffset;
        //targetState = STATE_EXPANDED;
        int currentTop = child.getTop();
        if (currentTop > mAnchorPoint) {
            top = mAnchorPoint;
            targetState = STATE_ANCHOR_POINT;
        }
        else {
            top = mMinOffset;
            targetState = STATE_EXPANDED;
        }
    } else if (mHideable && shouldHide(child, getYVelocity())) {
        top = mParentHeight;
        targetState = STATE_HIDDEN;
    } else if (mLastNestedScrollDy == 0) {
        int currentTop = child.getTop();
        if (Math.abs(currentTop - mMinOffset) < Math.abs(currentTop - mMaxOffset)) {
            top = mMinOffset;
            targetState = STATE_EXPANDED;
        } else {
            top = mMaxOffset;
            targetState = STATE_COLLAPSED;
        }
    } else {
        //top = mMaxOffset;
        //targetState = STATE_COLLAPSED;
        int currentTop = child.getTop();
        if (currentTop > mAnchorPoint) {
            top = mMaxOffset;
            targetState = STATE_COLLAPSED;
        }
        else {
            top = mAnchorPoint;
            targetState = STATE_ANCHOR_POINT;
        }
    }
    if (mViewDragHelper.smoothSlideViewTo(child, child.getLeft(), top)) {
        setStateInternal(STATE_SETTLING);
        ViewCompat.postOnAnimation(child, new SettleRunnable(child, targetState));
    } else {
        setStateInternal(targetState);
    }
    mNestedScrolled = false;
}

public final void setState(@State int state) {
    if (state == mState) {
        return;
    }
```

```
    if (mViewRef == null) {
        // The view is not laid out yet; modify mState and let onLayoutChild handle it later
        /**
         * New behavior (added: state == STATE_ANCHOR_POINT ||)
         */
        if (state == STATE_COLLAPSED || state == STATE_EXPANDED ||
                state == STATE_ANCHOR_POINT ||
                (mHideable && state == STATE_HIDDEN)) {
            mState = state;
        }
        return;
    }
    V child = mViewRef.get();
    if (child == null) {
        return;
    }
    int top;
    if (state == STATE_COLLAPSED) {
        top = mMaxOffset;
    } else if (state == STATE_ANCHOR_POINT) {
        top = mAnchorPoint;
    } else if (state == STATE_EXPANDED) {
        top = mMinOffset;
    } else if (mHideable && state == STATE_HIDDEN) {
        top = mParentHeight;
    } else {
        throw new IllegalArgumentException("Illegal state argument: " + state);
    }
    setStateInternal(STATE_SETTLING);
    if (mViewDragHelper.smoothSlideViewTo(child, child.getLeft(), top)) {
        ViewCompat.postOnAnimation(child, new SettleRunnable(child, state));
    }
}


public static <V extends View> BottomSheetBehaviorGoogleMapsLike<V> from(V view) {
    ViewGroup.LayoutParams params = view.getLayoutParams();
    if (!(params instanceof CoordinatorLayout.LayoutParams)) {
        throw new IllegalArgumentException("The view is not a child of CoordinatorLayout");
    }
    CoordinatorLayout.Behavior behavior = ((CoordinatorLayout.LayoutParams) params)
            .getBehavior();
    if (!(behavior instanceof BottomSheetBehaviorGoogleMapsLike)) {
        throw new IllegalArgumentException(
                "The view is not associated with BottomSheetBehaviorGoogleMapsLike");
    }
    return (BottomSheetBehaviorGoogleMapsLike<V>) behavior;
}
```

Enlace a todo el proyecto donde se pueden ver todos los comportamientos personalizados.

Y aquí es como se ve:
El



]

Configuración rápida

Asegúrese de que la siguiente dependencia se agregue al archivo build.gradle de su aplicación en las dependencias:

```
compile 'com.android.support:design:25.3.1'
```

Luego puedes usar la hoja inferior usando estas opciones:

- `BottomSheetBehavior` para ser utilizado con `CoordinatorLayout`
- `BottomSheetDialog` que es un diálogo con un comportamiento de la hoja inferior
- `BottomSheetDialogFragment` que es una extensión de `DialogFragment` , que crea un `BottomSheetDialog` lugar de un diálogo estándar.

Hojas inferiores persistentes

Puede lograr una Hoja de abajo persistente adjuntando una `BottomSheetBehavior` de `BottomSheetBehavior` a una vista de niño de un `CoordinatorLayout` :

```
<android.support.design.widget.CoordinatorLayout >

    <!-- .....   -->

    <LinearLayout
       android:id="@+id/bottom_sheet"
       android:elevation="4dp"
       android:minHeight="120dp"
       app:behavior_peekHeight="120dp"
       ...
       app:layout_behavior="android.support.design.widget.BottomSheetBehavior">

           <!-- .....   -->

       </LinearLayout>

</android.support.design.widget.CoordinatorLayout>
```

Luego, en tu código puedes crear una referencia usando:

```
 // The View with the BottomSheetBehavior
 View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
 BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

Puede establecer el estado de su BottomSheetBehavior utilizando el método setState () :

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

Puedes usar uno de estos estados:

- `STATE_COLLAPSED` : este estado colapsado es el predeterminado y muestra solo una parte del diseño en la parte inferior. La altura se puede controlar con el atributo `app:behavior_peekHeight` (predeterminado en 0)
- `STATE_EXPANDED` : el estado totalmente expandido de la hoja inferior, donde puede verse toda la hoja inferior (si su altura es menor que la que contiene el `CoordinatorLayout`) o la totalidad del `CoordinatorLayout` se llena
- `STATE_HIDDEN` : deshabilitado de forma predeterminada (y habilitado con la `app:behavior_hideable` atributo de `app:behavior_hideable` ocultable), lo que permite a los usuarios deslizarse hacia abajo en la hoja inferior para ocultar completamente la hoja inferior

Si desea recibir devoluciones de llamadas de cambios de estado, puede agregar una `BottomSheetCallback` :

```
mBottomSheetBehavior.setBottomSheetCallback(new BottomSheetCallback() {
    @Override
    public void onStateChanged(@NonNull View bottomSheet, int newState) {
      // React to state change
    }
      @Override
      public void onSlide(@NonNull View bottomSheet, float slideOffset) {
```

```
        // React to dragging events
    }
});
```

Hojas inferiores modales con BottomSheetDialogFragment

Puede realizar hojas `BottomSheetDialogFragment` modales utilizando un `BottomSheetDialogFragment` .

El `BottomSheetDialogFragment` es una hoja de fondo modal.
Esta es una versión de `DialogFragment` que muestra una hoja inferior usando `BottomSheetDialog` lugar de un diálogo flotante.

Solo define el fragmento:

```
public class MyBottomSheetDialogFragment extends BottomSheetDialogFragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        return inflater.inflate(R.layout.my_fragment_bottom_sheet, container);
    }
}
```

Luego usa este código para mostrar el fragmento:

```
MyBottomSheetDialogFragment mySheetDialog = new MyBottomSheetDialogFragment();
FragmentManager fm = getSupportFragmentManager();
mySheetDialog.show(fm, "modalSheetDialog");
```

Este fragmento creará un `BottomSheetDialog` .

Hojas de fondo modales con BottomSheetDialog

El `BottomSheetDialog` es un diálogo con el estilo de una hoja inferior

Solo usa:

```
//Create a new BottomSheetDialog
BottomSheetDialog dialog = new BottomSheetDialog(context);
//Inflate the layout R.layout.my_dialog_layout
dialog.setContentView(R.layout.my_dialog_layout);
//Show the dialog
dialog.show();
```

En este caso, no es necesario adjuntar un comportamiento BottomSheet.

Abra BottomSheet DialogFragment en modo Expandido de forma predeterminada.

BottomSheet DialogFragment se abre en `STATE_COLLAPSED` de forma predeterminada. Se puede forzar para abrir `STATE_EXPANDED` y ocupar la pantalla completa del dispositivo con la ayuda de la

siguiente plantilla de código.

@NonNull @Override public Dialog onCreateDialog (Bundle savedInstanceState) {

```
    BottomSheetDialog dialog = (BottomSheetDialog) super.onCreateDialog(savedInstanceState);

    dialog.setOnShowListener(new DialogInterface.OnShowListener() {
        @Override
        public void onShow(DialogInterface dialog) {
            BottomSheetDialog d = (BottomSheetDialog) dialog;

            FrameLayout bottomSheet = (FrameLayout)
d.findViewById(android.support.design.R.id.design_bottom_sheet);

BottomSheetBehavior.from(bottomSheet).setState(BottomSheetBehavior.STATE_EXPANDED);
        }
    });

    // Do something with your dialog like setContentView() or whatever
    return dialog;
}
```

Aunque la animación del diálogo es ligeramente perceptible, pero la tarea de abrir DialogFragment en pantalla completa es muy buena.

Lea Hojas inferiores en línea: https://riptutorial.com/es/android/topic/5702/hojas-inferiores

Capítulo 129: HttpURLConnection

Sintaxis

- desconexión abstracta del vacío ()
- Resumen booleano usingProxy ()
- estático booleano getFollowRedirects ()
- static void setFollowRedirects (conjunto booleano)
- Cadena getHeaderField (int n)
- Cadena getHeaderFieldKey (int n)
- Cadena getRequestMethod ()
- Cadena getResponseMessage ()
- int getResponseCode ()
- long getHeaderFieldDate (nombre de cadena, valor predeterminado largo)
- booleano getInstanceFollowRedirects ()
- Permiso getPermission ()
- InputStream getErrorStream ()
- void setChunkedStreamingMode (int chunklen)
- void setFixedLengthStreamingMode (int contentLength)
- void setFixedLengthStreamingMode (long contentLength)
- void setInstanceFollowRedirects (boolean followRedirects)
- void setRequestMethod (método String)

Observaciones

`HttpURLConnection` es el cliente HTTP estándar para Android, utilizado para enviar y recibir datos a través de la web. Es una implementación concreta de URLConnection para HTTP (RFC 2616).

Examples

Creando una conexión HttpURLC

Para crear un nuevo HTTP HTTP Client `HttpURLConnection` , llame a `openConnection()` en una instancia de URL. Dado que `openConnection()` devuelve un `URLConnection` , debe emitir explícitamente el valor devuelto.

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
// do something with the connection
```

Si está creando una nueva `URL` , también tiene que manejar las excepciones asociadas con el análisis de URL.

```
try {
```

```
    URL url = new URL("http://example.com");
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    // do something with the connection
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

Una vez que se ha leído el cuerpo de la respuesta y ya no se requiere la conexión, la conexión debe cerrarse llamando a `disconnect()` .

Aquí hay un ejemplo:

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
try {
    // do something with the connection
} finally {
    connection.disconnect();
}
```

Enviando una solicitud HTTP GET

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

try {
    BufferedReader br = new BufferedReader(new
InputStreamReader(connection.getInputStream()));

    // read the input stream
    // in this case, I simply read the first line of the stream
    String line = br.readLine();
    Log.d("HTTP-GET", line);

} finally {
    connection.disconnect();
}
```

Tenga en cuenta que las excepciones no se manejan en el ejemplo anterior. Un ejemplo completo, incluido el manejo de excepciones (trivial), sería:

```
URL url;
HttpURLConnection connection = null;

try {
    url = new URL("http://example.com");
    connection = (HttpURLConnection) url.openConnection();
    BufferedReader br = new BufferedReader(new
InputStreamReader(connection.getInputStream()));

    // read the input stream
    // in this case, I simply read the first line of the stream
    String line = br.readLine();
    Log.d("HTTP-GET", line);

} catch (IOException e) {
```

```
    e.printStackTrace();
} finally {
    if (connection != null) {
        connection.disconnect();
    }
}
```

Leyendo el cuerpo de una solicitud HTTP GET

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

try {
    BufferedReader br = new BufferedReader(new
InputStreamReader(connection.getInputStream()));

    // use a string builder to bufferize the response body
    // read from the input strea.
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line).append('\n');
    }

    // use the string builder directly,
    // or convert it into a String
    String body = sb.toString();

    Log.d("HTTP-GET", body);

} finally {
    connection.disconnect();
}
```

Tenga en cuenta que las excepciones no se manejan en el ejemplo anterior.

Utilice HttpURLConnection para multipart / form-data

Crear una clase personalizada para llamar a la solicitud HttpURLConnection multipart / form-data

MultipartUtility.java

```
public class MultipartUtility {
        private final String boundary;
        private static final String LINE_FEED = "\r\n";
        private HttpURLConnection httpConn;
        private String charset;
        private OutputStream outputStream;
        private PrintWriter writer;

        /**
         * This constructor initializes a new HTTP POST request with content type
         * is set to multipart/form-data
         *
         * @param requestURL
         * @param charset
```

```
     * @throws IOException
     */
    public MultipartUtility(String requestURL, String charset)
            throws IOException {
        this.charset = charset;

        // creates a unique boundary based on time stamp
        boundary = "===" + System.currentTimeMillis() + "===";
        URL url = new URL(requestURL);
        httpConn = (HttpURLConnection) url.openConnection();
        httpConn.setUseCaches(false);
        httpConn.setDoOutput(true);    // indicates POST method
        httpConn.setDoInput(true);
        httpConn.setRequestProperty("Content-Type",
                "multipart/form-data; boundary=" + boundary);
        outputStream = httpConn.getOutputStream();
        writer = new PrintWriter(new OutputStreamWriter(outputStream, charset),
                true);
    }

    /**
     * Adds a form field to the request
     *
     * @param name  field name
     * @param value field value
     */
    public void addFormField(String name, String value) {
        writer.append("--" + boundary).append(LINE_FEED);
        writer.append("Content-Disposition: form-data; name=\"" + name + "\"")
                .append(LINE_FEED);
        writer.append("Content-Type: text/plain; charset=" + charset).append(
                LINE_FEED);
        writer.append(LINE_FEED);
        writer.append(value).append(LINE_FEED);
        writer.flush();
    }

    /**
     * Adds a upload file section to the request
     *
     * @param fieldName  name attribute in <input type="file" name="..." />
     * @param uploadFile a File to be uploaded
     * @throws IOException
     */
    public void addFilePart(String fieldName, File uploadFile)
            throws IOException {
        String fileName = uploadFile.getName();
        writer.append("--" + boundary).append(LINE_FEED);
        writer.append(
                "Content-Disposition: form-data; name=\"" + fieldName
                        + "\"; filename=\"" + fileName + "\"")
                .append(LINE_FEED);
        writer.append(
                "Content-Type: "
                        + URLConnection.guessContentTypeFromName(fileName))
                .append(LINE_FEED);
        writer.append("Content-Transfer-Encoding: binary").append(LINE_FEED);
        writer.append(LINE_FEED);
        writer.flush();

        FileInputStream inputStream = new FileInputStream(uploadFile);
```

```
            byte[] buffer = new byte[4096];
            int bytesRead = -1;
            while ((bytesRead = inputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
            }
            outputStream.flush();
            inputStream.close();
            writer.append(LINE_FEED);
            writer.flush();
        }

        /**
         * Adds a header field to the request.
         *
         * @param name  - name of the header field
         * @param value - value of the header field
         */
        public void addHeaderField(String name, String value) {
            writer.append(name + ": " + value).append(LINE_FEED);
            writer.flush();
        }

        /**
         * Completes the request and receives response from the server.
         *
         * @return a list of Strings as response in case the server returned
         * status OK, otherwise an exception is thrown.
         * @throws IOException
         */
        public List<String> finish() throws IOException {
            List<String> response = new ArrayList<String>();
            writer.append(LINE_FEED).flush();
            writer.append("--" + boundary + "--").append(LINE_FEED);
            writer.close();

            // checks server's status code first
            int status = httpConn.getResponseCode();
            if (status == HttpURLConnection.HTTP_OK) {
                BufferedReader reader = new BufferedReader(new InputStreamReader(
                        httpConn.getInputStream()));
                String line = null;
                while ((line = reader.readLine()) != null) {
                    response.add(line);
                }
                reader.close();
                httpConn.disconnect();
            } else {
                throw new IOException("Server returned non-OK status: " + status);
            }
            return response;
        }
    }
```

Úsalo (modo asíncrono)

```
    MultipartUtility multipart = new MultipartUtility(requestURL, charset);

    // In your case you are not adding form data so ignore this
                /*This is to add parameter values */
                for (int i = 0; i < myFormDataArray.size(); i++) {
```

```
                    multipart.addFormField(myFormDataArray.get(i).getParamName(),
                            myFormDataArray.get(i).getParamValue());
                }


//add your file here.
                /*This is to add file content*/
                for (int i = 0; i < myFileArray.size(); i++) {
                    multipart.addFilePart(myFileArray.getParamName(),
                            new File(myFileArray.getFileName()));
                }

                List<String> response = multipart.finish();
                Debug.e(TAG, "SERVER REPLIED:");
                for (String line : response) {
                    Debug.e(TAG, "Upload Files Response:::" + line);
// get your server response here.
                    responseString = line;
                }
```

Enviando una solicitud HTTP POST con parámetros

Utilice un HashMap para almacenar los parámetros que deben enviarse al servidor a través de los parámetros POST:

```
HashMap<String, String> params;
```

Una vez que los `params` HashMap `params` completos, cree el StringBuilder que se usará para enviarlos al servidor:

```
    StringBuilder sbParams = new StringBuilder();
    int i = 0;
    for (String key : params.keySet()) {
        try {
            if (i != 0){
                sbParams.append("&");
            }
            sbParams.append(key).append("=")
                    .append(URLEncoder.encode(params.get(key), "UTF-8"));

        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        i++;
    }
```

Luego, cree HttpURLConnection, abra la conexión y envíe los parámetros POST:

```
try{
    String url = "http://www.example.com/test.php";
    URL urlObj = new URL(url);
    HttpURLConnection conn = (HttpURLConnection) urlObj.openConnection();
    conn.setDoOutput(true);
    conn.setRequestMethod("POST");
    conn.setRequestProperty("Accept-Charset", "UTF-8");
```

```
    conn.setReadTimeout(10000);
    conn.setConnectTimeout(15000);

    conn.connect();

    String paramsString = sbParams.toString();

    DataOutputStream wr = new DataOutputStream(conn.getOutputStream());
    wr.writeBytes(paramsString);
    wr.flush();
    wr.close();
} catch (IOException e) {
  e.printStackTrace();
}
```

Luego recibe el resultado que el servidor devuelve:

```
try {
  InputStream in = new BufferedInputStream(conn.getInputStream());
  BufferedReader reader = new BufferedReader(new InputStreamReader(in));
  StringBuilder result = new StringBuilder();
  String line;
  while ((line = reader.readLine()) != null) {
    result.append(line);
  }

  Log.d("test", "result from server: " + result.toString());

} catch (IOException e) {
  e.printStackTrace();
} finally {
    if (conn != null) {
        conn.disconnect();
    }
}
```

Cargar archivo (POST) utilizando HttpURLConnection

Muy a menudo es necesario enviar / cargar un archivo a un servidor remoto, por ejemplo, una imagen, video, audio o una copia de seguridad de la base de datos de la aplicación a un servidor privado remoto. Suponiendo que el servidor está esperando una solicitud POST con el contenido, aquí hay un ejemplo simple de cómo completar esta tarea en Android.

Las cargas de archivos se envían utilizando solicitudes POST `multipart/form-data` . Es muy fácil de implementar:

```
URL url = new URL(postTarget);
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

String auth = "Bearer " + oauthToken;
connection.setRequestProperty("Authorization", basicAuth);

String boundary = UUID.randomUUID().toString();
connection.setRequestMethod("POST");
connection.setDoOutput(true);
connection.setRequestProperty("Content-Type", "multipart/form-data;boundary=" + boundary);
```

```
DataOutputStream request = new DataOutputStream(uc.getOutputStream());

request.writeBytes("--" + boundary + "\r\n");
request.writeBytes("Content-Disposition: form-data; name=\"description\"\r\n\r\n");
request.writeBytes(fileDescription + "\r\n");

request.writeBytes("--" + boundary + "\r\n");
request.writeBytes("Content-Disposition: form-data; name=\"file\"; filename=\"" +
file.fileName + "\"\r\n\r\n");
request.write(FileUtils.readFileToByteArray(file));
request.writeBytes("\r\n");

request.writeBytes("--" + boundary + "--\r\n");
request.flush();
int respCode = connection.getResponseCode();

switch(respCode) {
    case 200:
        //all went ok - read response
        ...
        break;
    case 301:
    case 302:
    case 307:
        //handle redirect - for example, re-post to the new location
        ...
        break;
    ...
    default:
        //do something sensible
}
```

Por supuesto, las excepciones deberán ser capturadas o declaradas como lanzadas. Un par de puntos a tener en cuenta sobre este código:

1. `postTarget` es la URL de destino de la POST; `oauthToken` es el token de autenticación; `fileDescription` es la descripción del archivo, que se envía como el valor de la `description` del campo; `file` es el archivo que se va a enviar (es de tipo `java.io.File` : si tiene la ruta del archivo, puede usar un `new File(filePath)` .
2. Establece el encabezado de `Authorization` para una autenticación oAuth.
3. Utiliza Apache Common `FileUtil` para leer el archivo en una matriz de bytes: si ya tiene el contenido del archivo en una matriz de bytes o de alguna otra forma en la memoria, entonces no es necesario leerlo.

Una clase HttpURLConnection multipropósito para manejar todos los tipos de solicitudes HTTP

La siguiente clase se puede usar como una clase única que puede manejar `GET` , `POST` , `PUT` , `PATCH` y otras solicitudes:

```
class APIResponseObject{
    int responseCode;
    String response;
```

```
    APIResponseObject(int responseCode,String response)
    {
         this.responseCode = responseCode;
         this.response = response;
    }
}

public class APIAccessTask extends AsyncTask<String,Void,APIResponseObject> {
    URL requestUrl;
    Context context;
    HttpURLConnection urlConnection;
    List<Pair<String,String>> postData, headerData;
    String method;
    int responseCode = HttpURLConnection.HTTP_OK;

    interface OnCompleteListener{
        void onComplete(APIResponseObject result);
    }

    public OnCompleteListener delegate = null;

    APIAccessTask(Context context, String requestUrl, String method, OnCompleteListener
delegate){
        this.context = context;
        this.delegate = delegate;
        this.method = method;
        try {
            this.requestUrl = new URL(requestUrl);
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
    }

    APIAccessTask(Context context, String requestUrl, String method, List<Pair<String,String>>
postData, OnCompleteListener delegate){
        this(context, requestUrl, method, delegate);
        this.postData = postData;
    }

    APIAccessTask(Context context, String requestUrl, String method, List<Pair<String,String>>
postData,
            List<Pair<String,String>> headerData, OnCompleteListener delegate ){
        this(context, requestUrl,method,postData,delegate);
        this.headerData = headerData;
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected APIResponseObject doInBackground(String... params) {
        Log.d("debug", "url = "+ requestUrl);
        try {
            urlConnection = (HttpURLConnection) requestUrl.openConnection();

            if(headerData != null) {
                for (Pair pair : headerData) {
```

```
urlConnection.setRequestProperty(pair.first.toString(),pair.second.toString());
                }
            }

            urlConnection.setDoInput(true);
            urlConnection.setChunkedStreamingMode(0);
            urlConnection.setRequestMethod(method);
            urlConnection.connect();

            StringBuilder sb = new StringBuilder();

            if(!(method.equals("GET"))) {
                OutputStream out = new BufferedOutputStream(urlConnection.getOutputStream());
                BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out, "UTF-
8"));
                writer.write(getPostDataString(postData));
                writer.flush();
                writer.close();
                out.close();
            }

            urlConnection.connect();
            responseCode = urlConnection.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK) {
                InputStream in = new BufferedInputStream(urlConnection.getInputStream());
                BufferedReader reader = new BufferedReader(new InputStreamReader(in, "UTF-
8"));
                String line;

                while ((line = reader.readLine()) != null) {
                    sb.append(line);
                }
            }

            return new APIResponseObject(responseCode, sb.toString());
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
        return null;
    }

    @Override
    protected void onPostExecute(APIResponseObject result) {
        delegate.onComplete(result);
        super.onPostExecute(result);
    }

    private String getPostDataString(List<Pair<String, String>> params) throws
UnsupportedEncodingException {
        StringBuilder result = new StringBuilder();
        boolean first = true;
        for(Pair<String,String> pair : params){
            if (first)
                first = false;
            else
                result.append("&");

            result.append(URLEncoder.encode(pair.first,"UTF-8"));
            result.append("=");
            result.append(URLEncoder.encode(pair.second, "UTF-8"));
```

```
        }
        return result.toString();
    }
}
```

Uso

Use cualquiera de los constructores dados de la clase dependiendo de si necesita enviar datos `POST` o cualquier encabezado adicional.

Se `onComplete()` método `onComplete()` cuando se complete la `onComplete()` datos. Los datos se devuelven como un objeto de la clase `APIResponseObject` , que tiene un código de estado que indica el código de estado HTTP de la solicitud y una cadena que contiene la respuesta. Puede analizar esta respuesta en su clase, es decir, XML o JSON.

Llame a `execute()` en el objeto de la clase para ejecutar la solicitud, como se muestra en el siguiente ejemplo:

```
class MainClass {
    String url = "https://example.com./api/v1/ex";
    String method = "POST";
    List<Pair<String,String>> postData = new ArrayList<>();

    postData.add(new Pair<>("email","whatever");
    postData.add(new Pair<>("password", "whatever");

    new APIAccessTask(MainActivity.this, url, method, postData,
            new APIAccessTask.OnCompleteListener() {
        @Override
        public void onComplete(APIResponseObject result) {
            if (result.responseCode == HttpURLConnection.HTTP_OK) {
                String str = result.response;
                // Do your XML/JSON parsing here
            }
        }
    }).execute();
}
```

Lea HttpURLConnection en línea: https://riptutorial.com/es/android/topic/781/httpurlconnection

Capítulo 130: Huella digital API en Android

Observaciones

ver también

Proyecto de muestra Github

Desarrollador de Android blogspot

Sitio de desarrollador de Android

Examples

Añadiendo el escáner de huellas dactilares en la aplicación de Android

Android es compatible con la API de huellas dactilares de Android 6.0 (Marshmallow) SDK 23

> Para usar esta función en su aplicación, primero agregue el permiso USE_FINGERPRINT en su manifiesto.

```
<uses-permission
        android:name="android.permission.USE_FINGERPRINT" />
```

Aquí el procedimiento a seguir.

> Primero debe crear una clave simétrica en el Key Store de Android usando KeyGenerator, que solo se puede usar después de que el usuario se haya autenticado con la huella digital y pasar un KeyGenParameterSpec.

```
KeyPairGenerator.getInstance(KeyProperties.KEY_ALGORITHM_EC, "AndroidKeyStore");
keyPairGenerator.initialize(
        new KeyGenParameterSpec.Builder(KEY_NAME,
                KeyProperties.PURPOSE_SIGN)
                .setDigests(KeyProperties.DIGEST_SHA256)
                .setAlgorithmParameterSpec(new ECGenParameterSpec("secp256r1"))
                .setUserAuthenticationRequired(true)
                .build());
keyPairGenerator.generateKeyPair();
```

> Al establecer KeyGenParameterSpec.Builder.setUserAuthenticationRequired en true, puede permitir el uso de la clave solo después de que el usuario la autentifique, incluso cuando se autentica con la huella digital del usuario.

```
KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
keyStore.load(null);
PublicKey publicKey =
        keyStore.getCertificate(MainActivity.KEY_NAME).getPublicKey();
```

```
KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
keyStore.load(null);
PrivateKey key = (PrivateKey) keyStore.getKey(KEY_NAME, null);
```

> Luego comience a escuchar una huella dactilar en el sensor de huellas dactilares llamando a FingerprintManager.authenticate con un cifrado inicializado con la clave simétrica creada. O, alternativamente, puede recurrir a la contraseña verificada del lado del servidor como autenticador.

Crea e inicializa `FingerprintManger` desde `fingerprintManger.class`

```
getContext().getSystemService(FingerprintManager.class)
```

Para autenticar el uso de la Api de `FingerprintManger` y crear una subclase usando

`FingerprintManager.AuthenticationCallback` e invalida los métodos

```
onAuthenticationError
onAuthenticationHelp
onAuthenticationSucceeded
onAuthenticationFailed
```

Para comenzar

Para comenzar, escuchar el método de autentificación de llamada de evento fingerPrint con crypto

```
fingerprintManager
              .authenticate(cryptoObject, mCancellationSignal, 0 , this, null);
```

Cancelar

para dejar de escuchar la llamada del escáner

```
android.os.CancellationSignal;
```

> Una vez que se verifica la huella digital (o contraseña), se devuelve la devolución de llamada FingerprintManager.AuthenticationCallback # onAuthenticationSucceeded ().

```
@Override

public void onAuthenticationSucceeded(AuthenticationResult result) {

            }
```

Cómo utilizar la API de huellas digitales de Android para guardar las contraseñas de los usuarios

Esta clase auxiliar de ejemplo interactúa con el administrador de huellas digitales y realiza el

cifrado y descifrado de la contraseña. Tenga en cuenta que el método utilizado para el cifrado en este ejemplo es AES. Esta no es la única forma de cifrar y existen otros ejemplos . En este ejemplo, los datos se cifran y descifran de la siguiente manera:

Cifrado

1. El usuario le da al ayudante la contraseña no encriptada deseada.
2. El usuario debe proporcionar huella digital.
3. Una vez autenticado, el ayudante obtiene una clave del `KeyStore` y cifra la contraseña mediante un `Cipher` .
4. La contraseña y la sal IV (la IV se vuelve a crear para cada cifrado y no se reutiliza) se guardan en preferencias compartidas para usarlas más adelante en el proceso de descifrado.

Descifrado:

1. El usuario solicita descifrar la contraseña.
2. El usuario debe proporcionar huella digital.
3. El ayudante construye un `Cipher` utilizando el IV y una vez que el usuario se autentica, KeyStore obtiene una clave del `KeyStore` y descifra la contraseña.

```
public class FingerPrintAuthHelper {

    private static final String FINGER_PRINT_HELPER = "FingerPrintAuthHelper";
    private static final String ENCRYPTED_PASS_SHARED_PREF_KEY =
"ENCRYPTED_PASS_SHARED_PREF_KEY";
    private static final String LAST_USED_IV_SHARED_PREF_KEY = "LAST_USED_IV_SHARED_PREF_KEY";
    private static final String MY_APP_ALIAS = "MY_APP_ALIAS";

    private KeyguardManager keyguardManager;
    private FingerprintManager fingerprintManager;

    private final Context context;
    private KeyStore keyStore;
    private KeyGenerator keyGenerator;

    private String lastError;


    public interface Callback {
        void onSuccess(String savedPass);

        void onFailure(String message);

        void onHelp(int helpCode, String helpString);
    }

    public FingerPrintAuthHelper(Context context) {
        this.context = context;
    }

    public String getLastError() {
        return lastError;
    }

    @TargetApi(Build.VERSION_CODES.M)
```

```
    public boolean init() {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
            setError("This Android version does not support fingerprint authentication");
            return false;
        }

        keyguardManager = (KeyguardManager) context.getSystemService(KEYGUARD_SERVICE);
        fingerprintManager = (FingerprintManager)
context.getSystemService(FINGERPRINT_SERVICE);

        if (!keyguardManager.isKeyguardSecure()) {
            setError("User hasn't enabled Lock Screen");
            return false;
        }

        if (!hasPermission()) {
            setError("User hasn't granted permission to use Fingerprint");
            return false;
        }

        if (!fingerprintManager.hasEnrolledFingerprints()) {
            setError("User hasn't registered any fingerprints");
            return false;
        }

        if (!initKeyStore()) {
            return false;
        }
        return false;
    }

    @Nullable
    @RequiresApi(api = Build.VERSION_CODES.M)
    private Cipher createCipher(int mode) throws NoSuchPaddingException,
NoSuchAlgorithmException, UnrecoverableKeyException, KeyStoreException, InvalidKeyException,
InvalidAlgorithmParameterException {
        Cipher cipher = Cipher.getInstance(KeyProperties.KEY_ALGORITHM_AES + "/" +
                KeyProperties.BLOCK_MODE_CBC + "/" +
                KeyProperties.ENCRYPTION_PADDING_PKCS7);

        Key key = keyStore.getKey(MY_APP_ALIAS, null);
        if (key == null) {
            return null;
        }
        if(mode == Cipher.ENCRYPT_MODE) {
            cipher.init(mode, key);
            byte[] iv = cipher.getIV();
            saveIv(iv);
        } else {
            byte[] lastIv = getLastIv();
            cipher.init(mode, key, new IvParameterSpec(lastIv));
        }
        return cipher;
    }

    @NonNull
    @RequiresApi(api = Build.VERSION_CODES.M)
    private KeyGenParameterSpec createKeyGenParameterSpec() {
        return new KeyGenParameterSpec.Builder(MY_APP_ALIAS, KeyProperties.PURPOSE_ENCRYPT |
KeyProperties.PURPOSE_DECRYPT)
                        .setBlockModes(KeyProperties.BLOCK_MODE_CBC)
```

```
                        .setUserAuthenticationRequired(true)
                        .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_PKCS7)
                        .build();
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    private boolean initKeyStore() {
        try {
            keyStore = KeyStore.getInstance("AndroidKeyStore");
            keyGenerator = KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_AES,
"AndroidKeyStore");
            keyStore.load(null);
            if (getLastIv() == null) {
                KeyGenParameterSpec keyGeneratorSpec = createKeyGenParameterSpec();
                keyGenerator.init(keyGeneratorSpec);
                keyGenerator.generateKey();
            }
        } catch (Throwable t) {
            setError("Failed init of keyStore & keyGenerator: " + t.getMessage());
            return false;
        }
        return true;
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    private void authenticate(CancellationSignal cancellationSignal,
FingerPrintAuthenticationListener authListener, int mode) {
        try {
            if (hasPermission()) {
                Cipher cipher = createCipher(mode);
                FingerprintManager.CryptoObject crypto = new
FingerprintManager.CryptoObject(cipher);
                fingerprintManager.authenticate(crypto, cancellationSignal, 0, authListener,
null);
            } else {
                authListener.getCallback().onFailure("User hasn't granted permission to use
Fingerprint");
            }
        } catch (Throwable t) {
            authListener.getCallback().onFailure("An error occurred: " + t.getMessage());
        }
    }

    private String getSavedEncryptedPassword() {
        SharedPreferences sharedPreferences = getSharedPreferences();
        if (sharedPreferences != null) {
            return sharedPreferences.getString(ENCRYPTED_PASS_SHARED_PREF_KEY, null);
        }
        return null;
    }

    private void saveEncryptedPassword(String encryptedPassword) {
        SharedPreferences.Editor edit = getSharedPreferences().edit();
        edit.putString(ENCRYPTED_PASS_SHARED_PREF_KEY, encryptedPassword);
        edit.commit();
    }

    private byte[] getLastIv() {
        SharedPreferences sharedPreferences = getSharedPreferences();
        if (sharedPreferences != null) {
            String ivString = sharedPreferences.getString(LAST_USED_IV_SHARED_PREF_KEY, null);
```

```
            if (ivString != null) {
                return decodeBytes(ivString);
            }
        }
        return null;
    }

    private void saveIv(byte[] iv) {
        SharedPreferences.Editor edit = getSharedPreferences().edit();
        String string = encodeBytes(iv);
        edit.putString(LAST_USED_IV_SHARED_PREF_KEY, string);
        edit.commit();
    }

    private SharedPreferences getSharedPreferences() {
        return context.getSharedPreferences(FINGER_PRINT_HELPER, 0);
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    private boolean hasPermission() {
        return ActivityCompat.checkSelfPermission(context,
Manifest.permission.USE_FINGERPRINT) == PackageManager.PERMISSION_GRANTED;
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    public void savePassword(@NonNull String password, CancellationSignal cancellationSignal,
Callback callback) {
        authenticate(cancellationSignal, new FingerPrintEncryptPasswordListener(callback,
password), Cipher.ENCRYPT_MODE);
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    public void getPassword(CancellationSignal cancellationSignal, Callback callback) {
        authenticate(cancellationSignal, new FingerPrintDecryptPasswordListener(callback),
Cipher.DECRYPT_MODE);
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    public boolean encryptPassword(Cipher cipher, String password) {
        try {
            // Encrypt the text
            if(password.isEmpty()) {
                setError("Password is empty");
                return false;
            }

            if (cipher == null) {
                setError("Could not create cipher");
                return false;
            }

            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            CipherOutputStream cipherOutputStream = new CipherOutputStream(outputStream,
cipher);
            byte[] bytes = password.getBytes(Charset.defaultCharset());
            cipherOutputStream.write(bytes);
            cipherOutputStream.flush();
            cipherOutputStream.close();
            saveEncryptedPassword(encodeBytes(outputStream.toByteArray()));
        } catch (Throwable t) {
```

```
            setError("Encryption failed " + t.getMessage());
            return false;
        }

        return true;
    }

    private byte[] decodeBytes(String s) {
        final int len = s.length();

        // "111" is not a valid hex encoding.
        if( len%2 != 0 )
            throw new IllegalArgumentException("hexBinary needs to be even-length: "+s);

        byte[] out = new byte[len/2];

        for( int i=0; i<len; i+=2 ) {
            int h = hexToBin(s.charAt(i  ));
            int l = hexToBin(s.charAt(i+1));
            if( h==-1 || l==-1 )
                throw new IllegalArgumentException("contains illegal character for hexBinary:
"+s);

            out[i/2] = (byte)(h*16+l);
        }

        return out;
    }

    private static int hexToBin( char ch ) {
        if( '0'<=ch && ch<='9' )    return ch-'0';
        if( 'A'<=ch && ch<='F' )    return ch-'A'+10;
        if( 'a'<=ch && ch<='f' )    return ch-'a'+10;
        return -1;
    }

    private static final char[] hexCode = "0123456789ABCDEF".toCharArray();

    public String encodeBytes(byte[] data) {
        StringBuilder r = new StringBuilder(data.length*2);
        for ( byte b : data) {
            r.append(hexCode[(b >> 4) & 0xF]);
            r.append(hexCode[(b & 0xF)]);
        }
        return r.toString();
    }

    @NonNull
    private String decipher(Cipher cipher) throws IOException, IllegalBlockSizeException,
BadPaddingException {
        String retVal = null;
        String savedEncryptedPassword = getSavedEncryptedPassword();
        if (savedEncryptedPassword != null) {
            byte[] decodedPassword = decodeBytes(savedEncryptedPassword);
            CipherInputStream cipherInputStream = new CipherInputStream(new
ByteArrayInputStream(decodedPassword), cipher);

            ArrayList<Byte> values = new ArrayList<>();
            int nextByte;
            while ((nextByte = cipherInputStream.read()) != -1) {
                values.add((byte) nextByte);
```

```
            }
            cipherInputStream.close();

            byte[] bytes = new byte[values.size()];
            for (int i = 0; i < values.size(); i++) {
                bytes[i] = values.get(i).byteValue();
            }

            retVal = new String(bytes, Charset.defaultCharset());
        }
        return retVal;
    }

    private void setError(String error) {
        lastError = error;
        Log.w(FINGER_PRINT_HELPER, lastError);
    }

    @RequiresApi(Build.VERSION_CODES.M)
    protected class FingerPrintAuthenticationListener extends
FingerprintManager.AuthenticationCallback {

        protected final Callback callback;

        public FingerPrintAuthenticationListener(@NonNull Callback callback) {
            this.callback = callback;
        }

        public void onAuthenticationError(int errorCode, CharSequence errString) {
            callback.onFailure("Authentication error [" + errorCode + "] " + errString);
        }

        /**
         * Called when a recoverable error has been encountered during authentication. The
help
         * string is provided to give the user guidance for what went wrong, such as
         * "Sensor dirty, please clean it."
         * @param helpCode An integer identifying the error message
         * @param helpString A human-readable string that can be shown in UI
         */
        public void onAuthenticationHelp(int helpCode, CharSequence helpString) {
            callback.onHelp(helpCode, helpString.toString());
        }

        /**
         * Called when a fingerprint is recognized.
         * @param result An object containing authentication-related data
         */
        public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result)
{
        }

        /**
         * Called when a fingerprint is valid but not recognized.
         */
        public void onAuthenticationFailed() {
            callback.onFailure("Authentication failed");
        }

        public @NonNull
        Callback getCallback() {
```

```
            return callback;
        }

    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    private class FingerPrintEncryptPasswordListener extends FingerPrintAuthenticationListener
{

        private final String password;

        public FingerPrintEncryptPasswordListener(Callback callback, String password) {
            super(callback);
            this.password = password;
        }

        public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result)
{
            Cipher cipher = result.getCryptoObject().getCipher();
            try {
                if (encryptPassword(cipher, password)) {
                    callback.onSuccess("Encrypted");
                } else {
                    callback.onFailure("Encryption failed");
                }

            } catch (Exception e) {
                callback.onFailure("Encryption failed " + e.getMessage());
            }
        }
    }

    @RequiresApi(Build.VERSION_CODES.M)
    protected class FingerPrintDecryptPasswordListener extends
FingerPrintAuthenticationListener {

        public FingerPrintDecryptPasswordListener(@NonNull Callback callback) {
            super(callback);
        }

        public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result)
{
            Cipher cipher = result.getCryptoObject().getCipher();
            try {
                String savedPass = decipher(cipher);
                if (savedPass != null) {
                    callback.onSuccess(savedPass);
                } else {
                    callback.onFailure("Failed deciphering");
                }

            } catch (Exception e) {
                callback.onFailure("Deciphering failed " + e.getMessage());
            }
        }
    }
}
```

Esta actividad a continuación es un ejemplo muy básico de cómo obtener una contraseña guardada por el usuario e interactuar con el ayudante.

```
public class MainActivity extends AppCompatActivity {

    private TextView passwordTextView;
    private FingerPrintAuthHelper fingerPrintAuthHelper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        passwordTextView = (TextView) findViewById(R.id.password);
        errorTextView = (TextView) findViewById(R.id.error);

        View savePasswordButton = findViewById(R.id.set_password_button);
        savePasswordButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    fingerPrintAuthHelper.savePassword(passwordTextView.getText().toString(),
new CancellationSignal(), getAuthListener(false));
                }
            }
        });

        View getPasswordButton = findViewById(R.id.get_password_button);
        getPasswordButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    fingerPrintAuthHelper.getPassword(new CancellationSignal(),
getAuthListener(true));
                }
            }
        });
    }

    // Start the finger print helper. In case this fails show error to user
    private void startFingerPrintAuthHelper() {
        fingerPrintAuthHelper = new FingerPrintAuthHelper(this);
        if (!fingerPrintAuthHelper.init()) {
            errorTextView.setText(fingerPrintAuthHelper.getLastError());
        }
    }

    @NonNull
    private FingerPrintAuthHelper.Callback getAuthListener(final boolean isGetPass) {
        return new FingerPrintAuthHelper.Callback() {
            @Override
            public void onSuccess(String result) {
                if (isGetPass) {
                    errorTextView.setText("Success!!! Pass = " + result);
                } else {
                    errorTextView.setText("Encrypted pass = " + result);
                }
            }

            @Override
            public void onFailure(String message) {
                errorTextView.setText("Failed - " + message);
            }

            @Override
            public void onHelp(int helpCode, String helpString) {
```

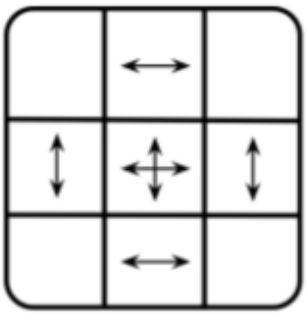
```
                errorTextView.setText("Help needed - " + helpString);
            }
        };
    }
}
```

Lea Huella digital API en Android en línea: https://riptutorial.com/es/android/topic/7523/huella-digital-api-en-android

Capítulo 131: Imágenes de 9 parches

Observaciones

Un archivo de imagen de *9 parches* es un archivo especialmente formateado para que Android sepa qué áreas / porciones de la imagen pueden o no pueden escalarse. Divide tu imagen en una cuadrícula de 3x3. Las esquinas permanecen sin escalar, los lados se escalan en una dirección y el centro se escala en ambas dimensiones.



Una imagen de Nine Patch (9-Patch) es un mapa de bits que tiene un borde ancho de un solo píxel alrededor de toda la imagen. Ignorando los 4 píxeles en las esquinas de la imagen. Este borde proporciona metadatos para el mapa de bits en sí. Los límites están marcados con líneas negras sólidas.

Una imagen de Nine Patch se almacena con la extensión `.9.png` .

El borde superior indica áreas que se extienden horizontalmente. El borde izquierdo indica áreas que se extienden verticalmente.

El borde inferior indica el relleno horizontalmente. El borde derecho indica relleno verticalmente.

Los bordes de relleno se utilizan generalmente para determinar dónde se va a dibujar el texto.

Google proporciona una excelente herramienta que simplifica *enormemente* la creación de estos archivos.

Ubicado en el SDK de Android: `android-sdk\tools\lib\draw9patch.jar`

Examples

Esquinas redondeadas basicas

La clave para estirar correctamente está en el borde superior e izquierdo.

El borde superior controla el estiramiento horizontal y el borde izquierdo controla el estiramiento

vertical.

Este ejemplo crea esquinas redondeadas adecuadas para una tostada.



Las partes de la imagen que están debajo del *borde superior* y a la derecha del *borde izquierdo* se expandirán para llenar todo el espacio no utilizado.

Este ejemplo se extenderá a todas las combinaciones de tamaños, como se muestra a continuación:



Hilandero basico

El `Spinner` se puede reajustar según sus propios requisitos de estilo utilizando un parche de Nueve.

Como ejemplo, vea este parche de nueve:



Como puedes ver, tiene 3 áreas extremadamente pequeñas de estiramiento marcadas.

El borde superior solo tiene la izquierda del icono marcado. Eso indica que quiero que el lado izquierdo (transparencia completa) del dibujable llene la vista del `Spinner` hasta que se alcance el ícono.

El borde izquierdo ha marcado segmentos transparentes en la parte superior e inferior del icono marcado. Eso indica que tanto la parte superior como la inferior se expandirán al tamaño de la vista del `Spinner` . Esto dejará el propio icono centrado verticalmente.

Usando la imagen sin metadatos de Nine Patch:



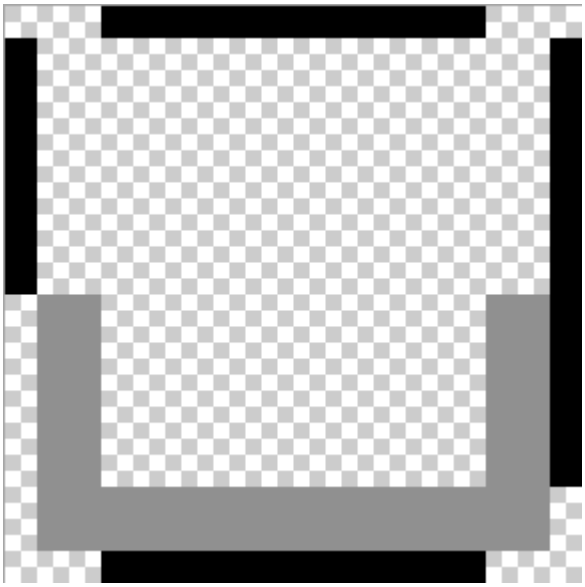
Usando la imagen con metadatos de Nine Patch:



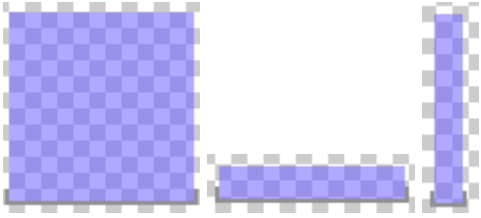
Líneas de relleno opcionales.

Las imágenes de nueve parches permiten la definición opcional de las líneas de relleno en la imagen. Las líneas de relleno son las líneas de la derecha y en la parte inferior.

Si una Vista establece la imagen de 9 parches como fondo, las líneas de relleno se utilizan para definir el espacio para el contenido de la Vista (por ejemplo, la entrada de texto en un `EditText`). Si las líneas de relleno no están definidas, se usan las líneas izquierda y superior en su lugar.



El área de contenido de la imagen estirada se ve así:



Lea Imágenes de 9 parches en línea: https://riptutorial.com/es/android/topic/461/imagenes-de-9-parches

Capítulo 132: ImageView

Introducción

ImageView (`android.widget.ImageView`) es una vista para mostrar y manipular recursos de imágenes, como Drawables y Bitmaps.

Algunos efectos, discutidos en este tema, pueden aplicarse a la imagen. La fuente de la imagen se puede configurar en un archivo XML (carpeta de `layout`) o programáticamente en un código Java.

Sintaxis

- El método `setImageResource(int resId)` establece un dibujo como el contenido de este `ImageView` .
- **Uso:** `imageView.setImageResource(R.drawable.anyImage)`

Parámetros

Parámetro	Descripción
`resId`	el nombre de su archivo de imagen en la carpeta `res` (generalmente en una carpeta `drawable`)

Examples

Establecer recurso de imagen

```
<ImageView
 android:id="@+id/imgExample"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 ...
 />
```

establece un dibujable como contenido de `ImageView` usando el atributo XML:

```
android:src="@drawable/android2"
```

establecer un dibujo programable:

```
 ImageView imgExample = (ImageView) findViewById(R.id.imgExample);
 imgExample.setImageResource(R.drawable.android2);
```

Establecer alfa

"alfa" se utiliza para especificar la opacidad de una imagen.

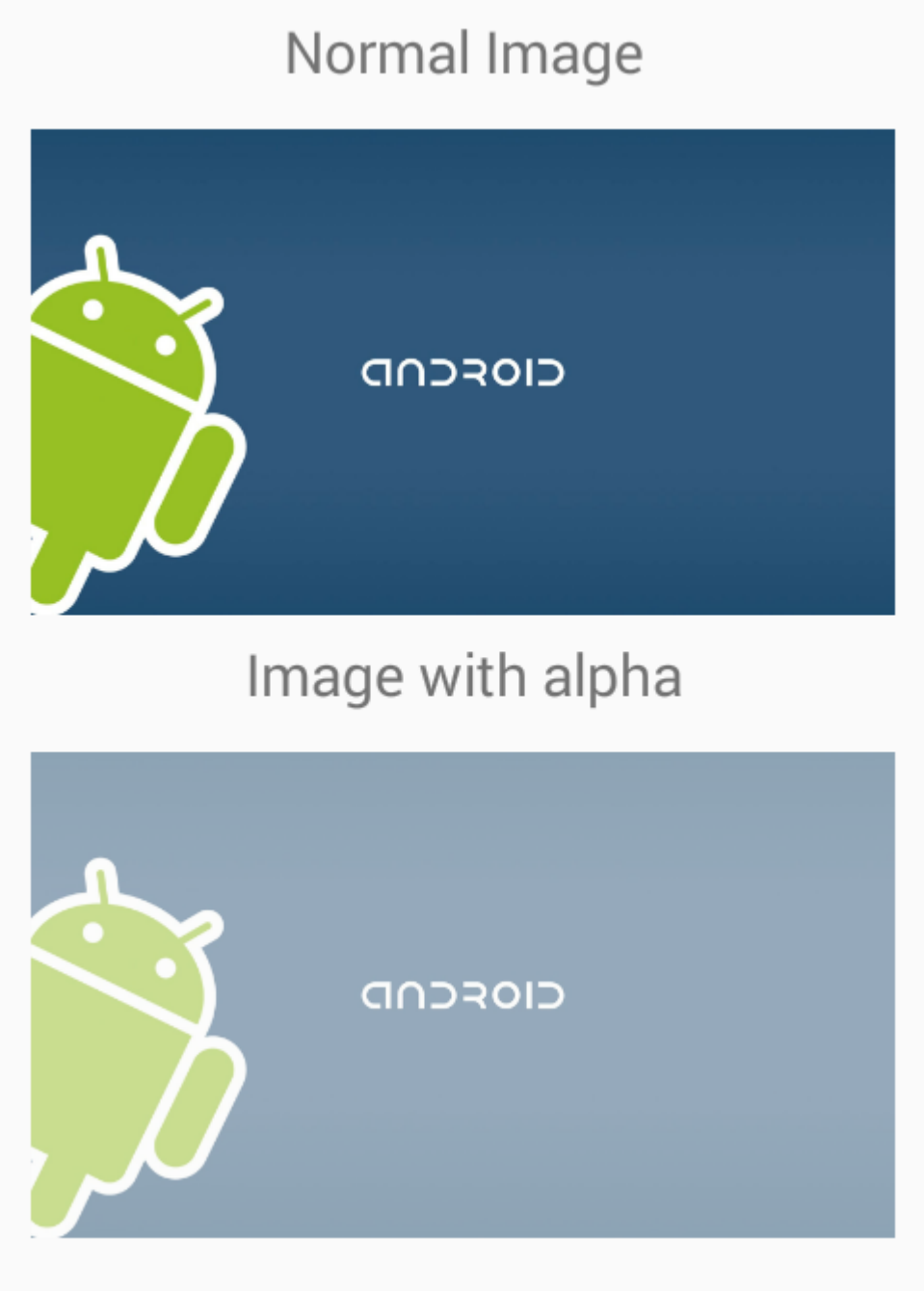
establece alfa usando el atributo XML:

```
android:alpha="0.5"
```

Nota: toma el valor flotante de 0 (transparente) a 1 (totalmente visible)

Establecer alfa programáticamente:

```
imgExample.setAlpha(0.5f);
```



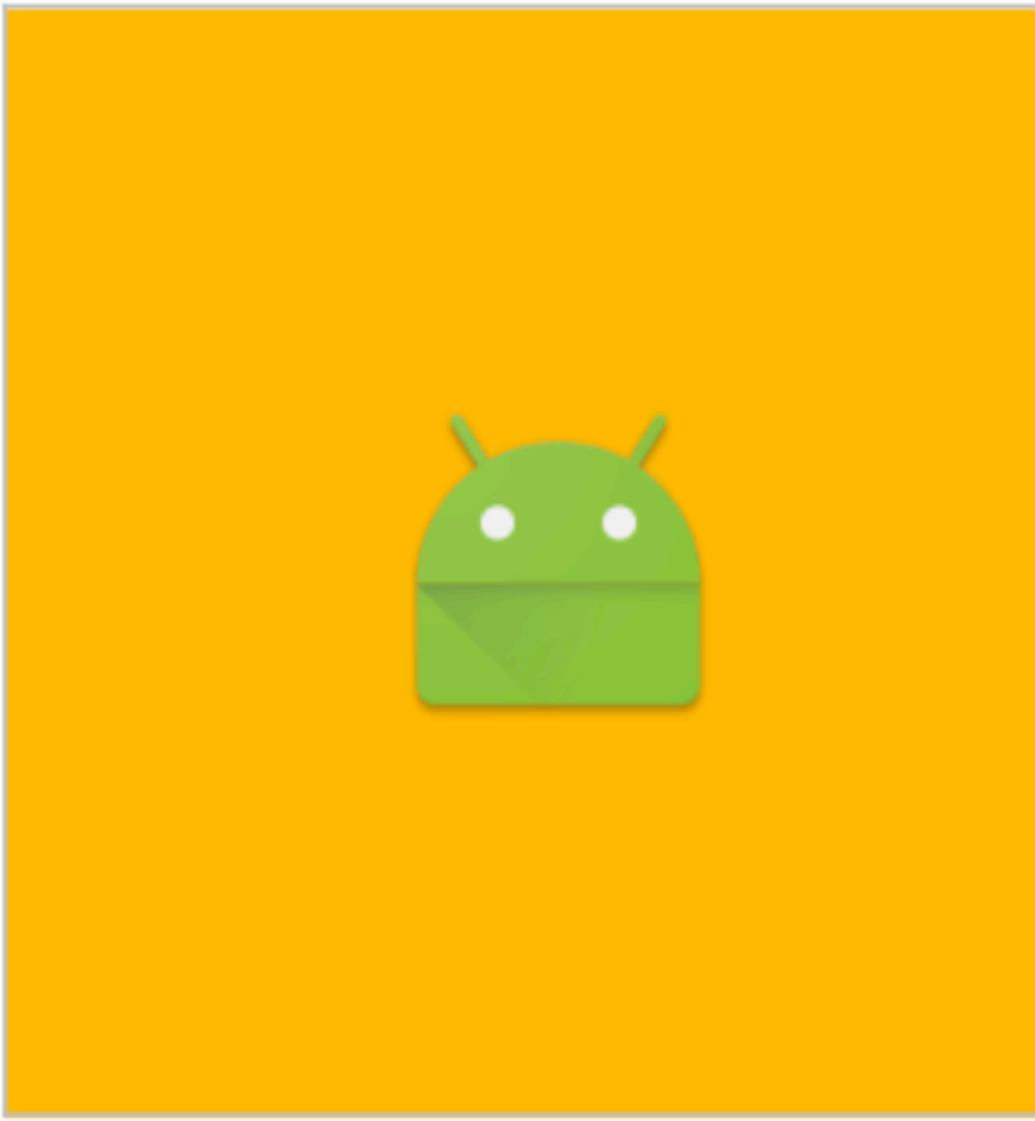
ImageView ScaleType - Centro

Es posible que la imagen contenida en ImageView no se ajuste al tamaño exacto dado al contenedor. En ese caso, el marco le permite cambiar el tamaño de la imagen de varias maneras.

Centrar

```
    <ImageView android:layout_width="20dp"
           android:layout_height="20dp"
           android:src="@mipmap/ic_launcher"
           android:id="@+id/imageView"
           android:scaleType="center"
           android:background="@android:color/holo_orange_light"/>
```

Esto no cambiará el tamaño de la imagen y la centrará dentro del contenedor *(Naranja = contenedor)*

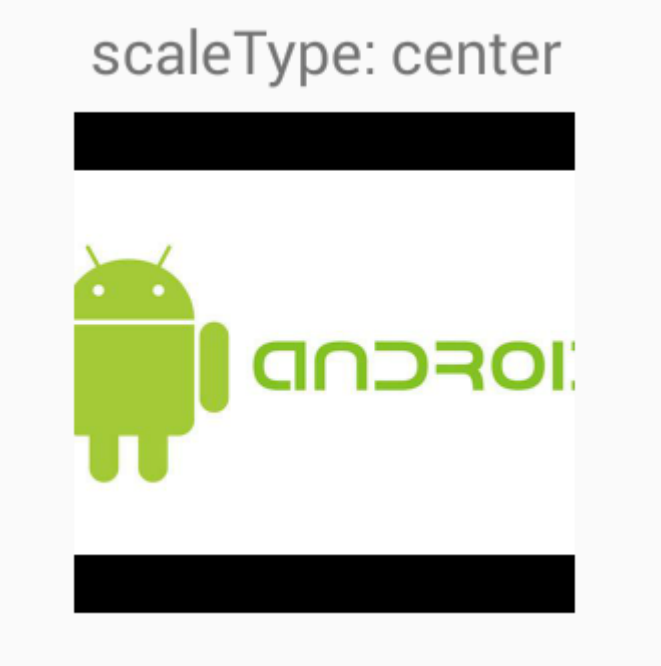


cuadrado que tiene un fondo negro y queremos mostrar un dibujo rectangular en un fondo blanco en `ImageView` .

```
<ImageView
 android:id="@+id/imgExample"
 android:layout_width="200dp"
 android:layout_height="200dp"
 android:background="#000"
 android:src="@drawable/android2"
 android:scaleType="..."/>
```

scaleType debe ser uno de los siguientes valores:

1. `center` : `center` la imagen en la vista, pero no realiza ninguna escala.



2. `centerCrop` : escala la imagen de manera uniforme (mantiene la relación de aspecto de la imagen) de modo que ambas dimensiones (ancho y alto) de la imagen sean iguales o mayores que la dimensión correspondiente de la vista (menos el relleno). La imagen se centra entonces en la vista.



3. `centerInside` : escala la imagen uniformemente (mantiene la relación de aspecto de la imagen) de modo que ambas dimensiones (ancho y alto) de la imagen sean iguales o menores que la dimensión correspondiente de la vista (menos el relleno). La imagen se centra entonces en la vista.



4. `matrix` : Escala usando la matriz de la imagen al dibujar.



5. `fitXY` : Escala la imagen usando FILL .



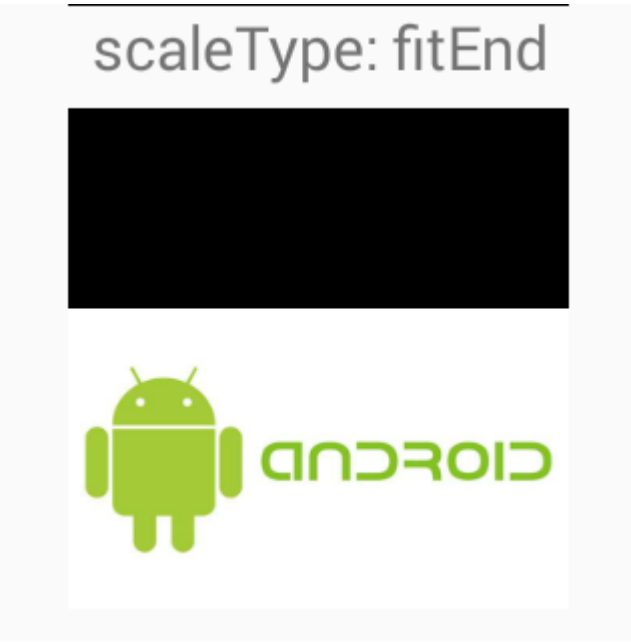
6. `fitStart` : escala la imagen usando START .



7. `fitCenter` : escala la imagen usando CENTRO .



8. `fitEnd` : Escala la imagen usando END .



Establecer tinte

Establecer un color de tintado para la imagen. Por defecto, el tinte se `SRC_ATOP` usando el modo `SRC_ATOP` .

establecer tinte usando el atributo XML:

```
android:tint="#009c38"
```

Nota: debe ser un valor de color, en forma de `"#rgb"` , `"#argb"` , `"#rrggbb"` o `"#aarrggbb"` .

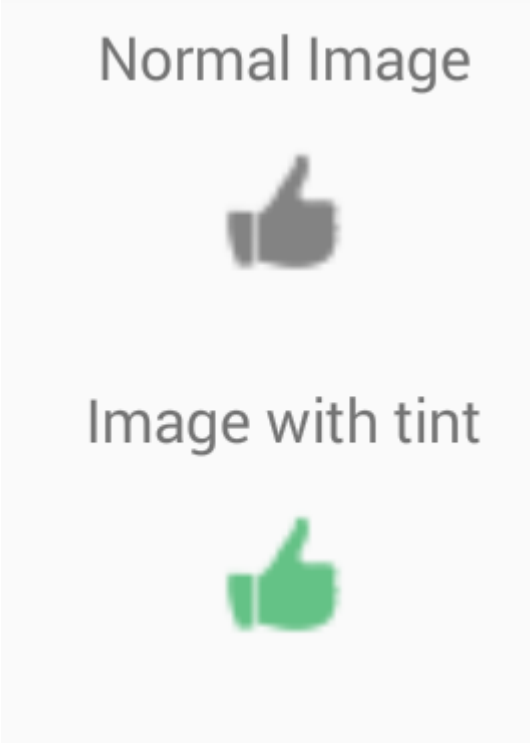
establecer tinte programáticamente:

```
imgExample.setColorFilter(Color.argb(255, 0, 156, 38));
```

y puedes borrar este filtro de color:

```
imgExample.clearColorFilter();
```

Ejemplo:



MLRoundedImageView.java

Copia y pega la siguiente clase en tu paquete:

```
public class MLRoundedImageView extends ImageView {

    public MLRoundedImageView(Context context) {
        super(context);
    }

    public MLRoundedImageView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public MLRoundedImageView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    @Override
    protected void onDraw(Canvas canvas) {

        Drawable drawable = getDrawable();

        if (drawable == null) {
            return;
        }

        if (getWidth() == 0 || getHeight() == 0) {
            return;
        }
        Bitmap b = ((BitmapDrawable) drawable).getBitmap();
        Bitmap bitmap = b.copy(Bitmap.Config.ARGB_8888, true);

        int w = getWidth(), h = getHeight();

        Bitmap roundBitmap = getCroppedBitmap(bitmap, w);
```

```
        canvas.drawBitmap(roundBitmap, 0, 0, null);

    }

    public static Bitmap getCroppedBitmap(Bitmap bmp, int radius) {
        Bitmap sbmp;

        if (bmp.getWidth() != radius || bmp.getHeight() != radius) {
            float smallest = Math.min(bmp.getWidth(), bmp.getHeight());
            float factor = smallest / radius;
            sbmp = Bitmap.createScaledBitmap(bmp, (int)(bmp.getWidth() / factor),
(int)(bmp.getHeight() / factor), false);
        } else {
            sbmp = bmp;
        }

        Bitmap output = Bitmap.createBitmap(radius, radius,
                Config.ARGB_8888);
        Canvas canvas = new Canvas(output);

        final int color = 0xffa19774;
        final Paint paint = new Paint();
        final Rect rect = new Rect(0, 0, radius, radius);

        paint.setAntiAlias(true);
        paint.setFilterBitmap(true);
        paint.setDither(true);
        canvas.drawARGB(0, 0, 0, 0);
        paint.setColor(Color.parseColor("#BAB399"));
        canvas.drawCircle(radius / 2 + 0.7f,
                radius / 2 + 0.7f, radius / 2 + 0.1f, paint);
        paint.setXfermode(new PorterDuffXfermode(Mode.SRC_IN));
        canvas.drawBitmap(sbmp, rect, rect, paint);

        return output;
    }
}
```

Utilice esta clase en XML con el nombre del paquete en lugar de `ImageView`

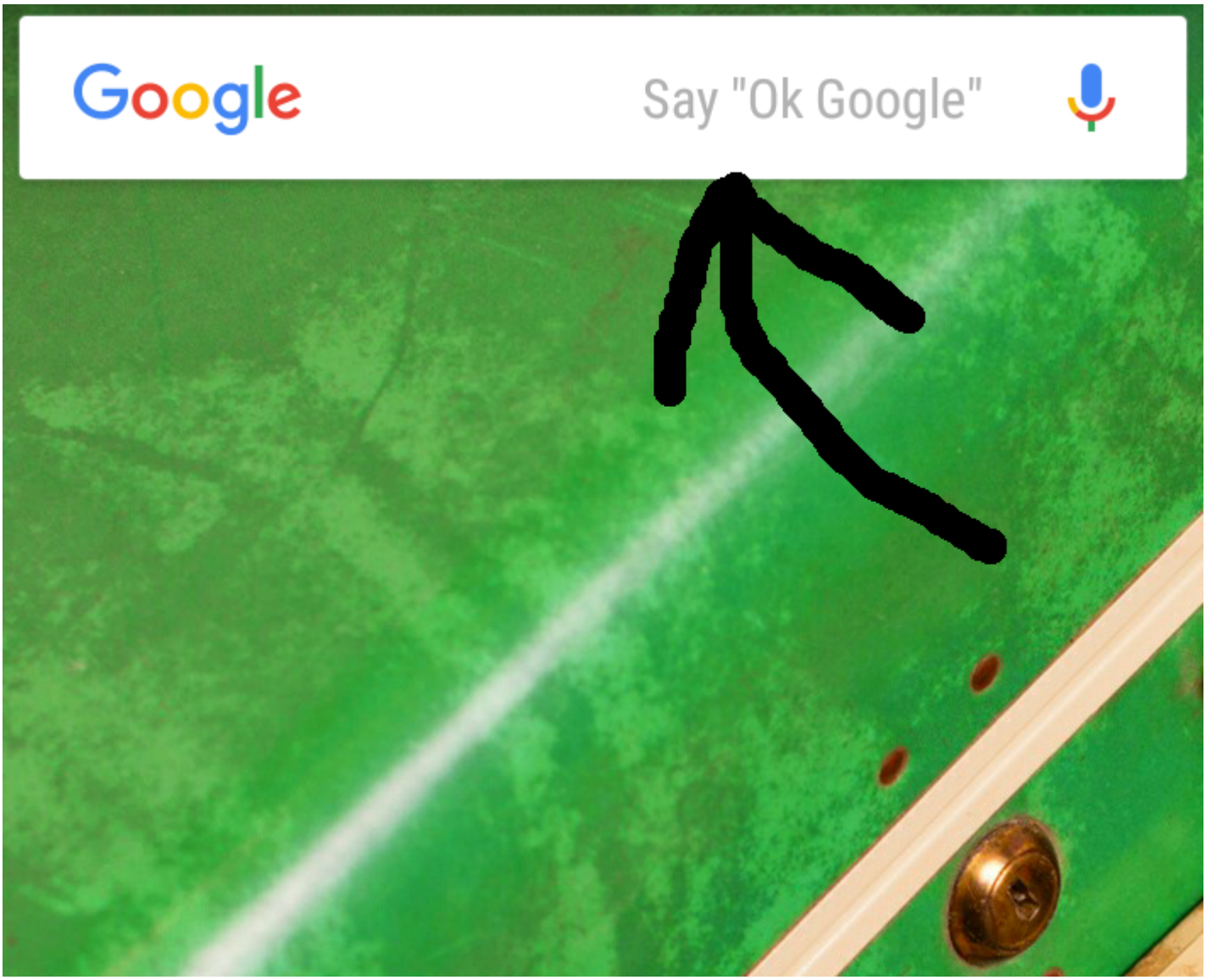
```
<com.androidbuts.example.MLRoundedImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />
```

Lea ImageView en línea: https://riptutorial.com/es/android/topic/4709/imageview

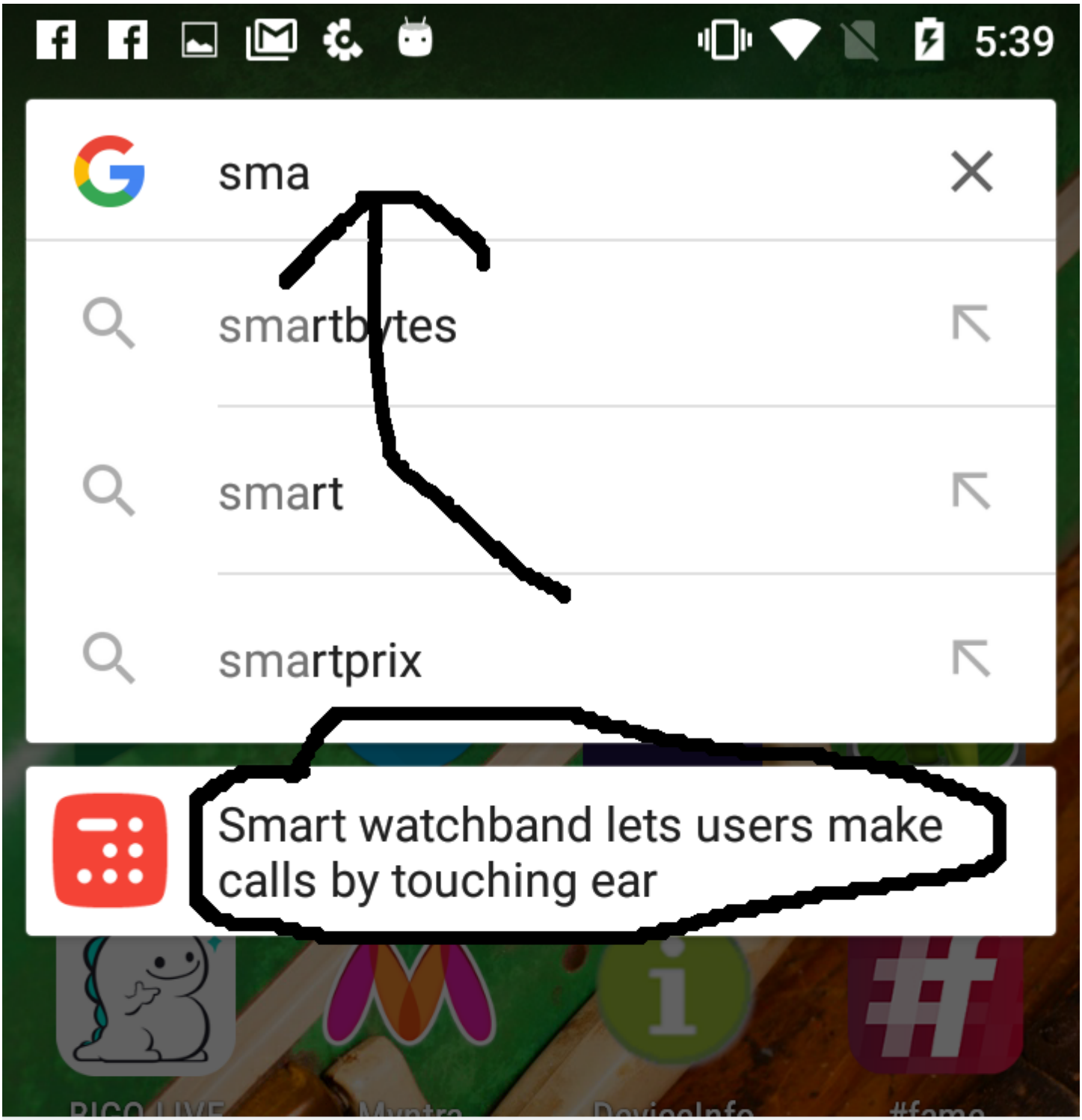
Capítulo 133: Indexación de la aplicación Firebase

Observaciones

- Cuando opta por implementar la indexación de aplicaciones, puede encontrar muchos blogs, documentación que lo puede confundir, en este caso, le sugiero que se limite a los documentos oficiales proporcionados por Firebase-Google. Incluso si desea utilizar un tercero para hacer esto, primero intente seguir esta documentación porque le dará una idea clara de cómo funcionan las cosas.
- Google tardará alrededor de 24 horas en indexar tu contenido. Sé paciente. Puedes hacer pruebas para que todo esté bien de tu lado.
- El primer ejemplo le permite admitir la URL HTTP de su sitio web para redirigir en su aplicación. Esto funcionará, por ejemplo, si ha buscado una consulta en la búsqueda de Google, los resultados muestran una de las URL de su sitio web, cuyos enlaces de aplicaciones están presentes en su aplicación que ya está instalada. Al hacer clic en esta URL, lo redireccionará directamente en la pantalla de la aplicación correspondiente a ese resultado de búsqueda. Eso es lo que he descubierto para esto.
- La adición de la API de AppIndexing indexa su contenido y se utiliza en las finalizaciones automáticas en la barra de búsqueda de Google. Tomemos un ejemplo de la aplicación inShorts para cada página, hay un título y una pequeña descripción. Después de leer 2 o 3 titulares, cierre la aplicación y muévase a google searchBar.



Intente ingresar el título por el que acaba de pasar; recibirá una sugerencia de la página de la aplicación con ese título como título. Esto es diferente de las sugerencias de aplicaciones que obtienes al buscar aplicaciones. Esto sucede porque ha escrito el código de la API de AppIndexing para esta página en particular y el título es el mismo que ha inicializado en `onCreate()` .


5:39
sma
smartbytes
smart
smartprix
Smart watchband lets users make calls by touching ear

Especifique la acción de intención de ACTION_VIEW para que se pueda acceder al filtro de intención desde la Búsqueda de Google.

<datos> Agregue una o más etiquetas, donde cada etiqueta representa un formato URI que se resuelve en la actividad. Como mínimo, la etiqueta debe incluir el atributo android: scheme. Puede agregar atributos adicionales para refinar aún más el tipo de URI que acepta la actividad. Por ejemplo, es posible que tenga varias actividades que acepten URI similares, pero que difieran simplemente en función del nombre de la ruta. En este caso, use el atributo android: path o sus variantes (pathPattern o pathPrefix) para diferenciar qué actividad debe abrir el sistema para diferentes rutas de URI.

<category> Incluir la categoría BROWSABLE. La categoría BROWSABLE es obligatoria para que se pueda acceder al filtro de intención desde un navegador web. Sin él, hacer clic en un enlace en un navegador no puede resolver su aplicación. La categoría POR DEFECTO es opcional, pero recomendada. Sin esta categoría, la actividad puede iniciarse solo con una intención explícita, utilizando el nombre del componente de su aplicación.

Paso 4: - Manejar URLS entrantes

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_schedule);
       onNewIntent(getIntent());
  }

   protected void onNewIntent(Intent intent) {
   String action = intent.getAction();
   Uri data = intent.getData();
   if (Intent.ACTION_VIEW.equals(action) && data != null) {
       articleId = data.getLastPathSegment();
       TextView linkText = (TextView)findViewById(R.id.link);
       linkText.setText(data.toString());
   }
```

}

Paso 5: - Puedes probar esto usando el comando Debug Bridge de Android o las configuraciones de estudio. Comando Adb: - Inicie su aplicación y luego ejecute este comando: -

```
adb shell am start -a android.intent.action.VIEW -d "{URL}" < package name >
```

Configuraciones de Android Studio: - **Android studio> Construir> Editar configuración> Opciones de lanzamiento> seleccionar URL> luego escriba su URL aquí> Aplicar** y probar. Ejecute su aplicación. Si la ventana “Ejecutar” muestra un error, debe verificar su formato de URL con su los applinks mencionados en el manifiesto, de lo contrario, se ejecutarán con éxito, y se redirigirán a la página mencionada su URL si se especifica.

Añadir API de AppIndexing

Para agregar esto al proyecto, puede encontrar documentos oficiales fácilmente, pero en este

ejemplo voy a resaltar algunas de las áreas clave que deben ser atendidas.

Paso 1: - Añadir servicio de google

```
dependencies {
      ...
      compile 'com.google.android.gms:play-services-appindexing:9.4.0'
      ...
    }
```

Paso 2: - Importar clases

```
import com.google.android.gms.appindexing.Action;
import com.google.android.gms.appindexing.AppIndex;
import com.google.android.gms.common.api.GoogleApiClient;
```

Paso 3: - Agregar llamadas API de indexación de aplicaciones

```
private GoogleApiClient mClient;
private Uri mUrl;
private String mTitle;
private String mDescription;

//If you know the values that to be indexed then you can initialize these variables in
onCreate()
@Override
protected void onCreate(Bundle savedInstanceState) {
mClient = new GoogleApiClient.Builder(this).addApi(AppIndex.API).build();
mUrl = "http://examplepetstore.com/dogs/standard-poodle";
mTitle = "Standard Poodle";
mDescription = "The Standard Poodle stands at least 18 inches at the withers";
}

//If your data is coming from a network request, then initialize these value in onResponse()
and make checks for NPE so that your code won't fall apart.

//setting title and description for App Indexing
mUrl = Uri.parse("android-app://com.famelive/https/m.fame.live/vod/" +model.getId());
mTitle =   model.getTitle();
mDescription = model.getDescription();

mClient.connect();
AppIndex.AppIndexApi.start(mClient, getAction());

@Override
protected void onStop() {
if (mTitle != null && mDescription != null && mUrl != null) //if your response fails then
check whether these are initialized or not
   if (getAction() != null) {
       AppIndex.AppIndexApi.end(mClient, getAction());
       mClient.disconnect();
    }
 super.onStop();
}

 public Action getAction() {
   Thing object = new Thing.Builder()
       .setName(mTitle)
```

```
        .setDescription(mDescription)
        .setUrl(mUrl)
        .build();

 return new Action.Builder(Action.TYPE_WATCH)
        .setObject(object)
        .setActionStatus(Action.STATUS_TYPE_COMPLETED)
        .build();
}
```

Para probar esto, simplemente siga el paso 4 en las Observaciones que se dan a continuación.

Lea Indexación de la aplicación Firebase en línea:
https://riptutorial.com/es/android/topic/5957/indexacion-de-la-aplicacion-firebase

Capítulo 134: Instalando aplicaciones con ADB

Examples

Instalar una aplicación

Escribe el siguiente comando en tu terminal:

```
adb install [-rtsdg] <file>
```

Tenga en cuenta que debe pasar un archivo que está en su computadora y no en su dispositivo.

Si agrega `-r` al final, se sobrescribirán los apks existentes en conflicto. De lo contrario, el comando se cerrará con un error.

`-g` otorgará inmediatamente todos los permisos de tiempo de ejecución.

`-d` permite degradar el código de la versión (solo se puede aplicar en paquetes que se pueden depurar).

Use `-s` para instalar la aplicación en la tarjeta SD externa.

`-t` Permitirá utilizar aplicaciones de prueba.

Desinstalar una aplicación

Escriba el siguiente comando en su terminal para desinstalar una aplicación con un nombre de paquete provisto:

```
adb uninstall <packagename>
```

Instalar todo el archivo apk en el directorio

Windows:

```
for %f in (C:\your_app_path\*.apk) do adb install "%f"
```

Linux:

```
for f in *.apk ; do adb install "$f" ; done
```

Lea Instalando aplicaciones con ADB en línea:
https://riptutorial.com/es/android/topic/5301/instalando-aplicaciones-con-adb

Capítulo 135: Integración de Android Paypal Gateway

Observaciones

Paypal nos proporciona su propia biblioteca para el pago, por lo que ahora es mucho más seguro y fácil de implementar en nuestra aplicación. A continuación se presentan los pasos importantes a realizar.

Examples

Configure paypal en su código de Android

1) Primero, visite el sitio web de Paypal Developer y cree una aplicación.

2) Ahora abra su archivo de manifiesto y otorgue los permisos a continuación

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

3) Y algunas actividades y servicios requeridos

```
 <service
        android:name="com.paypal.android.sdk.payments.PayPalService"
        android:exported="false" />
    <activity android:name="com.paypal.android.sdk.payments.PaymentActivity" />
    <activity android:name="com.paypal.android.sdk.payments.LoginActivity" />
    <activity android:name="com.paypal.android.sdk.payments.PaymentMethodActivity" />
    <activity android:name="com.paypal.android.sdk.payments.PaymentConfirmActivity" />
    <activity android:name="com.paypal.android.sdk.payments.PayPalFuturePaymentActivity" />
    <activity android:name="com.paypal.android.sdk.payments.FuturePaymentConsentActivity" />
    <activity android:name="com.paypal.android.sdk.payments.FuturePaymentInfoActivity" />
    <activity
        android:name="io.card.payment.CardIOActivity"
        android:configChanges="keyboardHidden|orientation" />
    <activity android:name="io.card.payment.DataEntryActivity" />
```

4) Abra su clase de actividad y configure la configuración para su aplicación

```
//set the environment for production/sandbox/no netowrk
 private static final String CONFIG_ENVIRONMENT = PayPalConfiguration.ENVIRONMENT_PRODUCTION;
```

5) Ahora configure el ID de cliente desde la cuenta de desarrollador de Paypal. Cadena final estática privada CONFIG_CLIENT_ID = "PONER SU ID DE CLIENTE"; 6) Dentro del método onCreate, llame al servicio Paypal: Intención intención = nueva Intención (esto, PayPalService.class); intent.putExtra (PayPalService.EXTRA_PAYPAL_CONFIGURATION, config); startService (intención);

7) Ahora está listo para realizar un pago con solo presionar un botón, llamar a la Actividad de pago-

```
PayPalPayment thingToBuy = new PayPalPayment(new BigDecimal(1),"USD", "androidhub4you.com",
                                PayPalPayment.PAYMENT_INTENT_SALE);
                            Intent intent = new Intent(MainActivity.this,
PaymentActivity.class);
                            intent.putExtra(PaymentActivity.EXTRA_PAYMENT, thingToBuy);

                            startActivityForResult(intent, REQUEST_PAYPAL_PAYMENT);
```

8) Y finalmente de onActivityResult obtener la respuesta de pago-

```
 @Override
           protected void onActivityResult(int requestCode, int resultCode, Intent data) {
               if (requestCode == REQUEST_PAYPAL_PAYMENT) {
                   if (resultCode == Activity.RESULT_OK) {
                       PaymentConfirmation confirm = data
                               .getParcelableExtra(PaymentActivity.EXTRA_RESULT_CONFIRMATION);
                       if (confirm != null) {
                           try {
                           System.out.println("Responseeee"+confirm);
                               Log.i("paymentExample", confirm.toJSONObject().toString());

                               JSONObject jsonObj=new
JSONObject(confirm.toJSONObject().toString());

                               String
paymentId=jsonObj.getJSONObject("response").getString("id");
                               System.out.println("payment id:-==="+paymentId);
                               Toast.makeText(getApplicationContext(), paymentId,
Toast.LENGTH_LONG).show();
                           } catch (JSONException e) {
                               Log.e("paymentExample", "an extremely unlikely failure
occurred: ", e);
                           }
                       }
                   } else if (resultCode == Activity.RESULT_CANCELED) {
                       Log.i("paymentExample", "The user canceled.");
                   } else if (resultCode == PaymentActivity.RESULT_EXTRAS_INVALID) {
                       Log.i("paymentExample", "An invalid Payment was submitted. Please see
the docs.");
                   }
               }


           }
```

Lea Integración de Android Paypal Gateway en línea:
https://riptutorial.com/es/android/topic/5895/integracion-de-android-paypal-gateway

Capítulo 136: Integración de inicio de sesión de Google en Android

Introducción

Este tema se basa en Cómo integrar el inicio de sesión de Google, en las aplicaciones de Android

Examples

Integración de google Auth en tu proyecto. (Obtener un archivo de configuración)

Primero obtenga el archivo de configuración para iniciar sesión desde

Abrir enlace a continuación

[https://developers.google.com/identity/sign-in/android/start-integrating [...]]

haga clic en obtener un archivo de configuración

- Ingrese el nombre de la aplicación y el nombre del paquete y haga clic en elegir y configurar servicios
- proporcionar SHA1 Habilitar google SIGNIN y generar archivos de configuración

Descargue el archivo de configuración y coloque el archivo en la aplicación / carpeta de su proyecto

1. Agregue la dependencia a su build.gradle a nivel de proyecto:

   classpath 'com.google.gms: google-services: 3.0.0'

2. Agregue el complemento a su build.gradle a nivel de aplicación: (parte inferior)

   aplique el complemento: 'com.google.gms.google-services'

3. Añade esta dependencia a tu aplicación.

   dependencias {compile 'com.google.android.gms: play-services-auth: 9.8.0'}

Implementación de código de Google SignIn

- En el método onCreate de su actividad de inicio de sesión, configure el inicio de sesión de Google para solicitar los datos de usuario requeridos por su aplicación.

```
 GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
```

```
        .requestEmail()
        .build();
```

- cree un objeto GoogleApiClient con acceso a la API de inicio de sesión de Google y las opciones que especificó.

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
        .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
        .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
        .build();
```

- Ahora, cuando el usuario haga clic en el botón de inicio de sesión de Google, llame a esta función.

```
 private void signIn() {
 Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
 startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

- implementar OnActivityResult para obtener la respuesta.

```
 @Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    // Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
        handleSignInResult(result);
    }
}
```

- Último paso Manejar el resultado y obtener datos de usuario

```
 private void handleSignInResult(GoogleSignInResult result) {
 Log.d(TAG, "handleSignInResult:" + result.isSuccess());
 if (result.isSuccess()) {
     // Signed in successfully, show authenticated UI.
     GoogleSignInAccount acct = result.getSignInAccount();
     mStatusTextView.setText(getString(R.string.signed_in_fmt, acct.getDisplayName()));
     updateUI(true);
 } else {
     // Signed out, show unauthenticated UI.
     updateUI(false);
 }
}
```

Lea Integración de inicio de sesión de Google en Android en línea:
https://riptutorial.com/es/android/topic/9960/integracion-de-inicio-de-sesion-de-google-en-android

Capítulo 137: Integrar el inicio de sesión de Google

Sintaxis

- newInstance (): para crear una instancia única de Google Helper
- initGoogleSignIn () - Para inicializar el inicio de sesión de Google
- getGoogleAccountDetails (): para iniciar sesión en los detalles de la cuenta
- signOut () - Para cerrar sesión de usuario
- getGoogleClient (): para utilizar GoogleApiClient

Parámetros

Parámetro	Detalle
ETIQUETA	Una cadena utilizada durante el registro
GoogleSignInHelper	Una referencia estática para ayudante.
AppCompatActivity	Una referencia de actividad
GoogleApiClient	Una referencia de GoogleAPIClient
RC_SIGN_IN	Un entero representa una constante de resultado de actividad
isLoggingOut	Un booleano para comprobar si la tarea de cierre de sesión se está ejecutando o no

Examples

Google Inicia sesión con la clase de ayuda

Agregue a continuación a su `build.gradle` fuera de la etiqueta de `android` :

```
// Apply plug-in to app.
apply plugin: 'com.google.gms.google-services'
```

Agregue la siguiente clase de ayuda a su paquete util:

```
/**
 * Created by Andy
 */
public class GoogleSignInHelper implements GoogleApiClient.OnConnectionFailedListener,
        GoogleApiClient.ConnectionCallbacks {
```

```
    private static final String TAG = GoogleSignInHelper.class.getSimpleName();

    private static GoogleSignInHelper googleSignInHelper;
    private AppCompatActivity mActivity;
    private GoogleApiClient mGoogleApiClient;
    public static final int RC_SIGN_IN = 9001;
    private boolean isLoggingOut = false;

    public static GoogleSignInHelper newInstance(AppCompatActivity mActivity) {
        if (googleSignInHelper == null) {
            googleSignInHelper = new GoogleSignInHelper(mActivity, fireBaseAuthHelper);
        }
        return googleSignInHelper;
    }

    public GoogleSignInHelper(AppCompatActivity mActivity) {
        this.mActivity = mActivity;
        initGoogleSignIn();
    }


    private void initGoogleSignIn() {
        // [START config_sign_in]
        // Configure Google Sign In
        GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                .requestIdToken(mActivity.getString(R.string.default_web_client_id))
                .requestEmail()
                .build();
        // [END config_sign_in]

        mGoogleApiClient = new GoogleApiClient.Builder(mActivity)
                .enableAutoManage(mActivity /* FragmentActivity */, this /*
OnConnectionFailedListener */)
                .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
                .addConnectionCallbacks(this)
                .build();

    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
        // An unresolvable error has occurred and Google APIs (including Sign-In) will not
        // be available.
        Log.d(TAG, "onConnectionFailed:" + connectionResult);
        Toast.makeText(mActivity, "Google Play Services error.", Toast.LENGTH_SHORT).show();
    }

    public void getGoogleAccountDetails(GoogleSignInResult result) {
        // Google Sign In was successful, authenticate with FireBase
        GoogleSignInAccount account = result.getSignInAccount();
        // You are now logged into Google
    }
    public void signOut() {

        if (mGoogleApiClient.isConnected()) {

            // Google sign out
            Auth.GoogleSignInApi.signOut(mGoogleApiClient).setResultCallback(
                    new ResultCallback<Status>() {
                        @Override
```

```
                        public void onResult(@NonNull Status status) {
                            isLoggingOut = false;
                        }
                    });
        } else {
            isLoggingOut = true;
        }
    }

    public GoogleApiClient getGoogleClient() {
        return mGoogleApiClient;
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        Log.w(TAG, "onConnected");
        if (isLoggingOut) {
            signOut();
        }
    }

    @Override
    public void onConnectionSuspended(int i) {
        Log.w(TAG, "onConnectionSuspended");
    }
}
```

Agregue el siguiente código a su `OnActivityResult` en el archivo de actividad:

```
 // [START onactivityresult]
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        // Result returned from launching the Intent from
GoogleSignInApi.getSignInIntent(...);
        if (requestCode == GoogleSignInHelper.RC_SIGN_IN) {
            GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
            if (result.isSuccess()) {
                googleSignInHelper.getGoogleAccountDetails(result);
            } else {
                // Google Sign In failed, update UI appropriately
                // [START_EXCLUDE]
                Log.d(TAG, "signInWith Google failed");
                // [END_EXCLUDE]
            }
        }
    }
    // [END onactivityresult]

 // [START signin]
    public void signIn() {
        Intent signInIntent =
Auth.GoogleSignInApi.getSignInIntent(googleSignInHelper.getGoogleClient());
        startActivityForResult(signInIntent, GoogleSignInHelper.RC_SIGN_IN);
    }

    // [END signin]
```

Lea Integrar el inicio de sesión de Google en línea:
https://riptutorial.com/es/android/topic/2837/integrar-el-inicio-de-sesion-de-google

Capítulo 138: Integrar OpenCV en Android Studio

Observaciones

Las bibliotecas Open CV se pueden encontrar en la web utilizando un motor de búsqueda.

Las **Gotchas** :

- Si baja su plataforma de destino debajo de KitKat, algunas de las bibliotecas de OpenCV ya no funcionarán, específicamente las clases relacionadas con *org.opencv.android.Camera2Renderer* y otras clases relacionadas. Probablemente pueda evitar esto simplemente eliminando los archivos apropiados de OpenCV .java.
- Si eleva su plataforma de destino a Lollipop o más arriba, es posible que mi ejemplo de carga de un archivo no funcione porque el uso de rutas de archivos absolutas está mal visto. Por lo tanto, es posible que tenga que cambiar el ejemplo para cargar un archivo de la galería o en otro lugar. Hay numerosos ejemplos flotando alrededor.

Examples

Instrucciones

Probado con AS v1.4.1 pero también debería funcionar con versiones más nuevas.

1. Cree un nuevo proyecto de Android Studio utilizando el asistente de proyectos (Menú: / Archivo / Nuevo proyecto):

   - Llámalo " **cvtest1** "
   - Factor de forma: **API 19, Android 4.4 (KitKat)**
   - **Actividad en blanco** llamada **MainActivity**

   Debe tener un directorio *cvtest1* donde se almacena este proyecto. (La barra de título de Android Studio muestra dónde está cvtest1 cuando abres el proyecto)

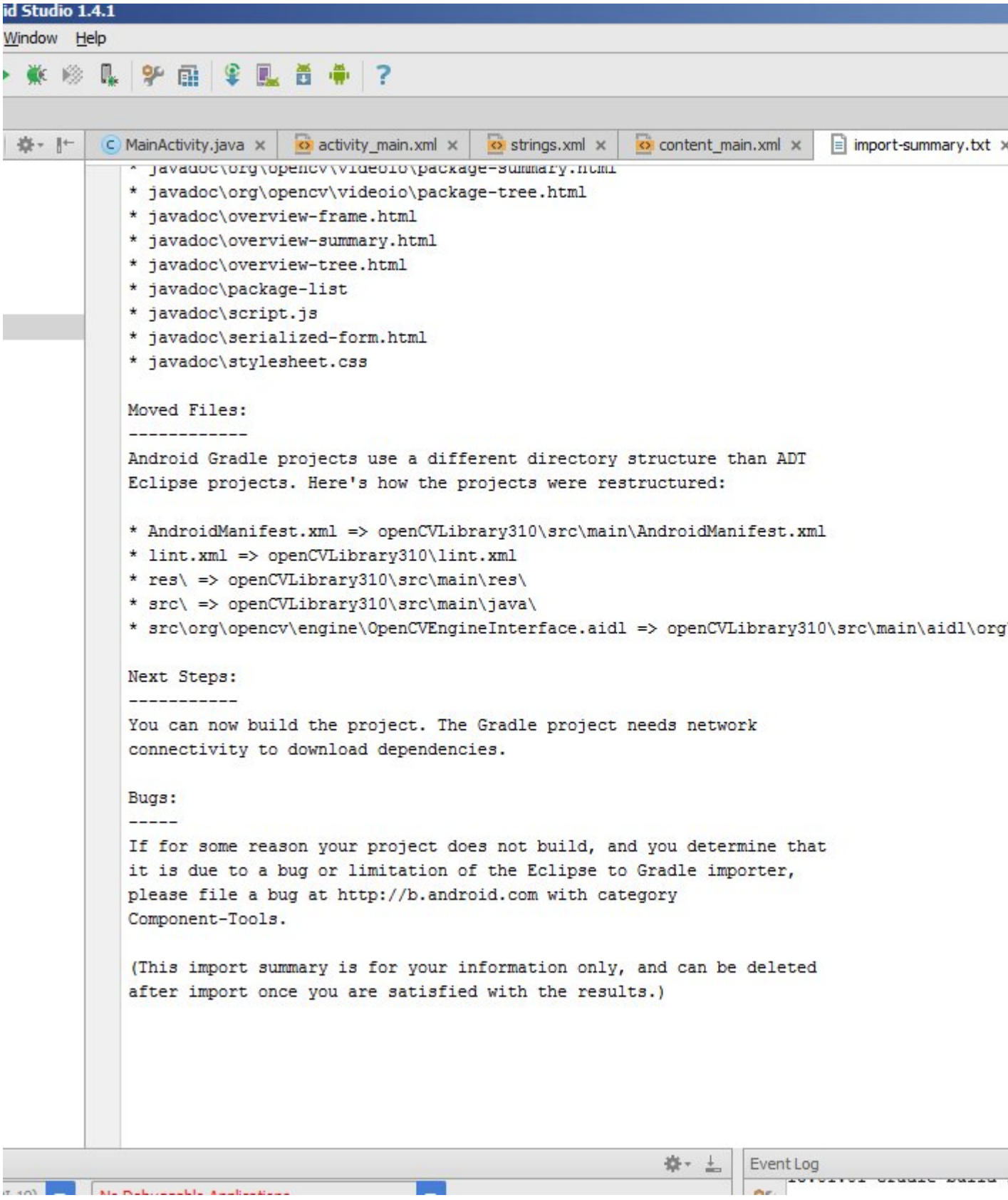
2. Verifique que su aplicación se ejecute correctamente. Intente cambiar algo como el texto "Hola mundo" para confirmar que el ciclo de compilación / prueba está bien para usted. (Estoy probando con un emulador de un dispositivo API 19).

3. Descargue el paquete OpenCV para Android v3.1.0 y descomprímalo en algún directorio temporal en algún lugar. (Asegúrese de que sea el paquete específicamente para Android y no solo el paquete OpenCV para Java). Llamaré a este directorio " **unzip-dir** " Debajo de **unzip-dir** debería tener un directorio **sdk / native / libs** con subdirectorios que comienzan con cosas como el **brazo** ..., **mips** ... y **x86** ... (uno para cada tipo de "arquitectura" con Android)

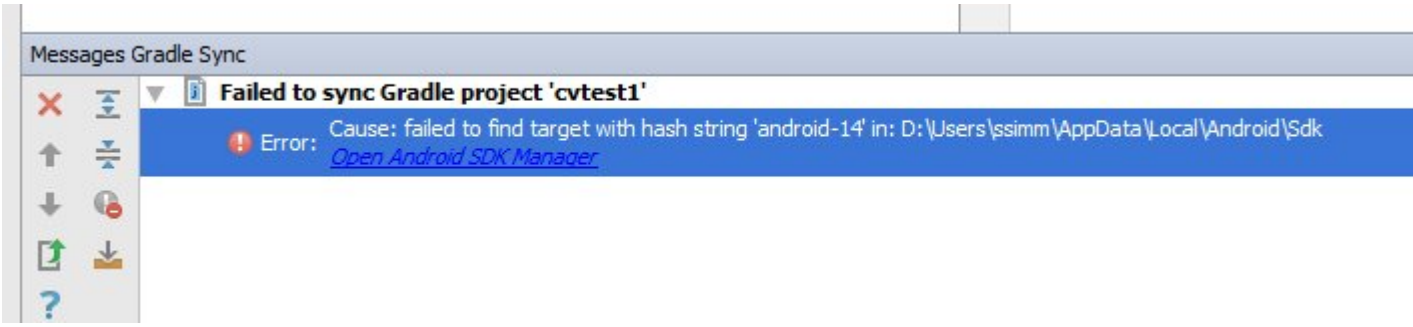
4. Desde Android Studio, importe OpenCV en su proyecto como módulo: **Menú: / Archivo / Nuevo / Import_Module** :

    - Directorio de origen: **{unzip-dir} / sdk / java**
    - Nombre del módulo: Android studio rellena automáticamente este campo con **openCVLibrary310** (el nombre exacto probablemente no importa, pero seguiremos con esto).
    - Haga clic en **siguiente** . Obtendrá una pantalla con tres casillas de verificación y preguntas sobre archivos jar, bibliotecas y opciones de importación. Los tres deben ser revisados. Haga clic en **Finalizar.**

    Android Studio comienza a importar el módulo y se muestra un archivo **import-summary.txt** que contiene una lista de lo que no se importó (en su mayoría archivos javadoc) y otros datos.



Pero también aparece un mensaje de error que indica que **no se pudo encontrar el objetivo con la cadena de hash 'android-14' ....** Esto sucede porque el archivo build.gradle en el archivo zip OpenCV que descargaste dice que compilar utilizando la versión 14 de la API de Android, que de forma predeterminada no tiene con Android Studio v1.4.1.

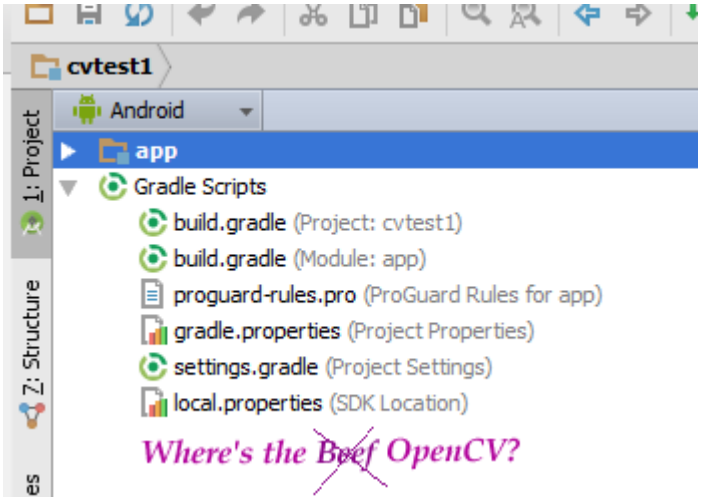


5. Abra el diálogo de la estructura del proyecto (**Menú: / Archivo / Estructura_proyecto**). Seleccione el módulo "aplicación", haga clic en la pestaña **Dependencias** y agregue **: openCVLibrary310** como una dependencia de módulo. Cuando selecciona **Agregar / Módulo_dependencia** , debería aparecer en la lista de módulos que puede agregar. Ahora aparecerá como una dependencia, pero obtendrá algunos errores más de " **no encontrar-android-14"** en el registro de eventos.

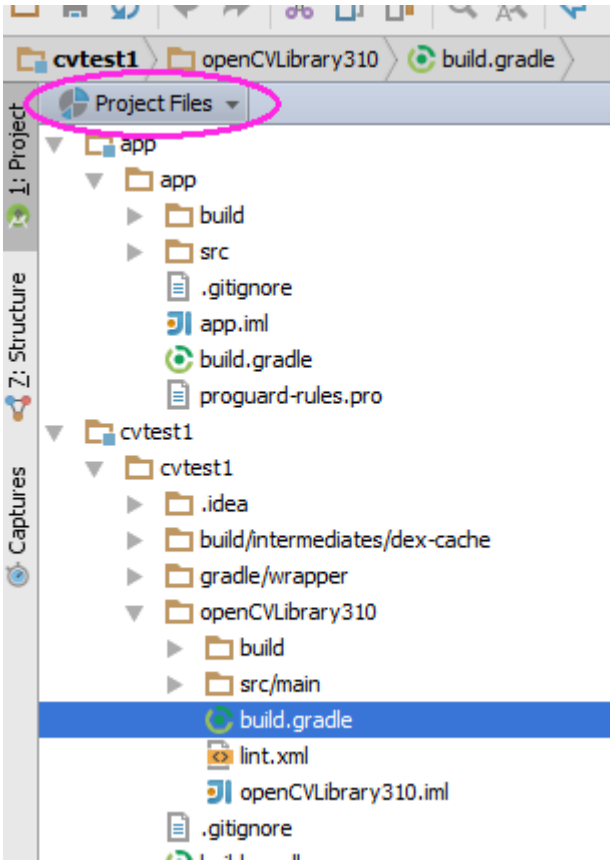
6. Busque en el archivo **build.gradle** para su módulo de aplicación. Hay varios archivos build.gradle en un proyecto de Android. El que desea está en el directorio **cvtest1 / app** y, desde la vista del proyecto, se parece a **build.gradle (Module: app)** . Tenga en cuenta los valores de estos cuatro campos:

    - compileSDKVersion (el mío dice 23)
    - buildToolsVersion (el mío dice 23.0.2)
    - minSdkVersion (el mio dice 19)
    - targetSdkVersion (el mío dice 23)

7. Su proyecto ahora tiene un directorio **cvtest1 / OpenCVLibrary310** pero no es visible desde la vista del proyecto:

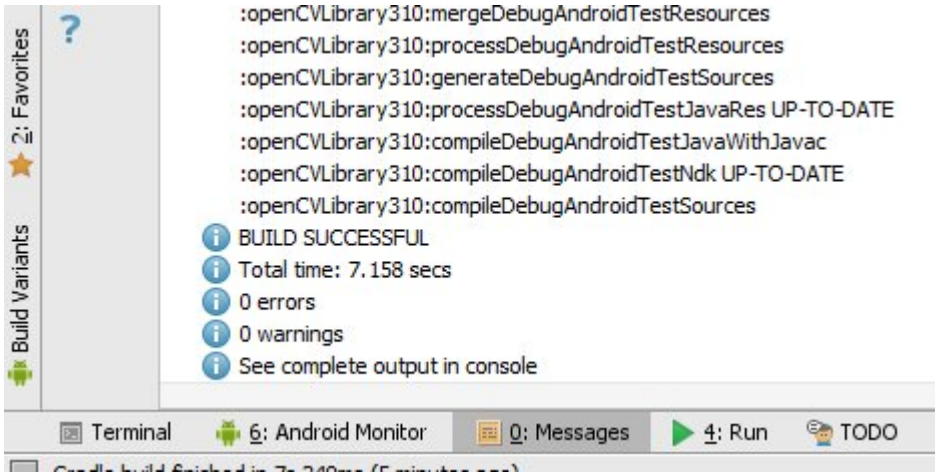


Use alguna otra herramienta, como cualquier administrador de archivos, y vaya a este directorio. También puede cambiar la vista del proyecto de **Android** a **Archivos de proyecto** y puede encontrar este directorio como se muestra en esta captura de pantalla:



Dentro hay otro archivo **build.gradle** (está resaltado en la captura de pantalla anterior). Actualice este archivo con los cuatro valores del paso 6.

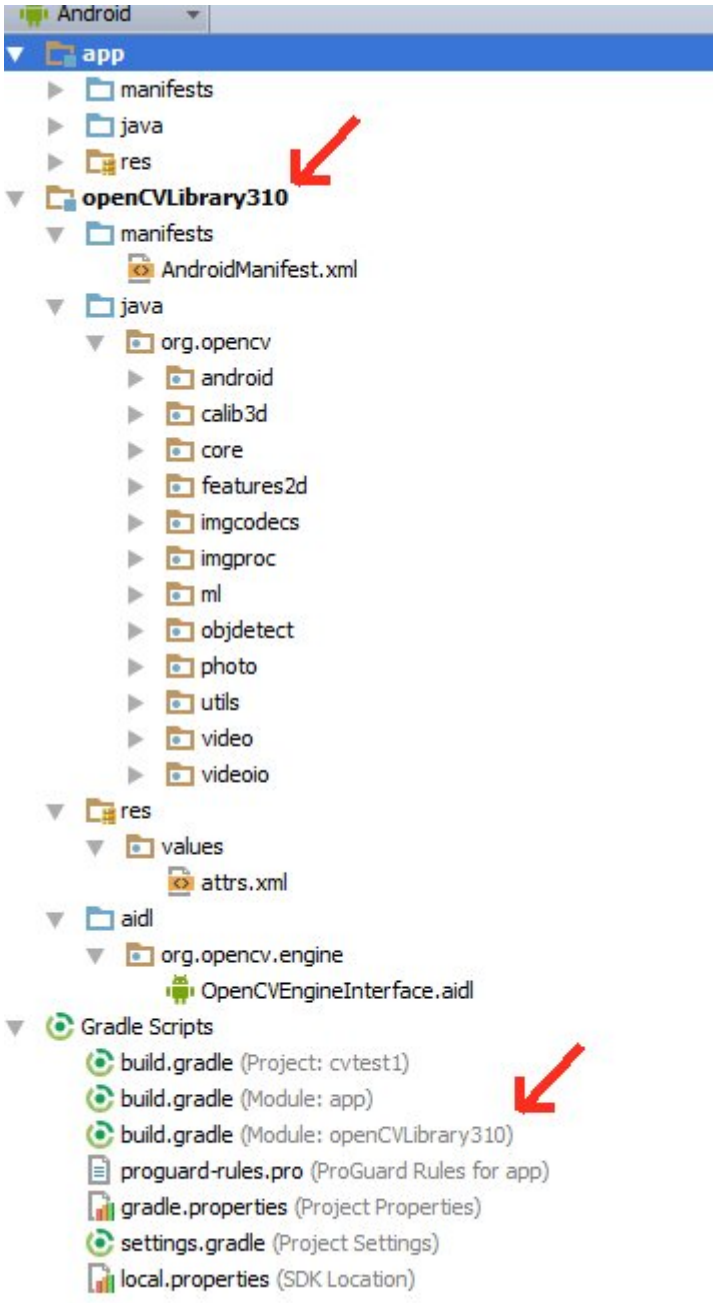
8. Vuelva a sincronizar su proyecto y luego límpielo / reconstrúyalo. (**Menú: / Build / Clean_Project**) Debería limpiarse y construirse sin errores y debería ver muchas referencias a **: openCVLibrary310** en la pantalla **0: Messages** .



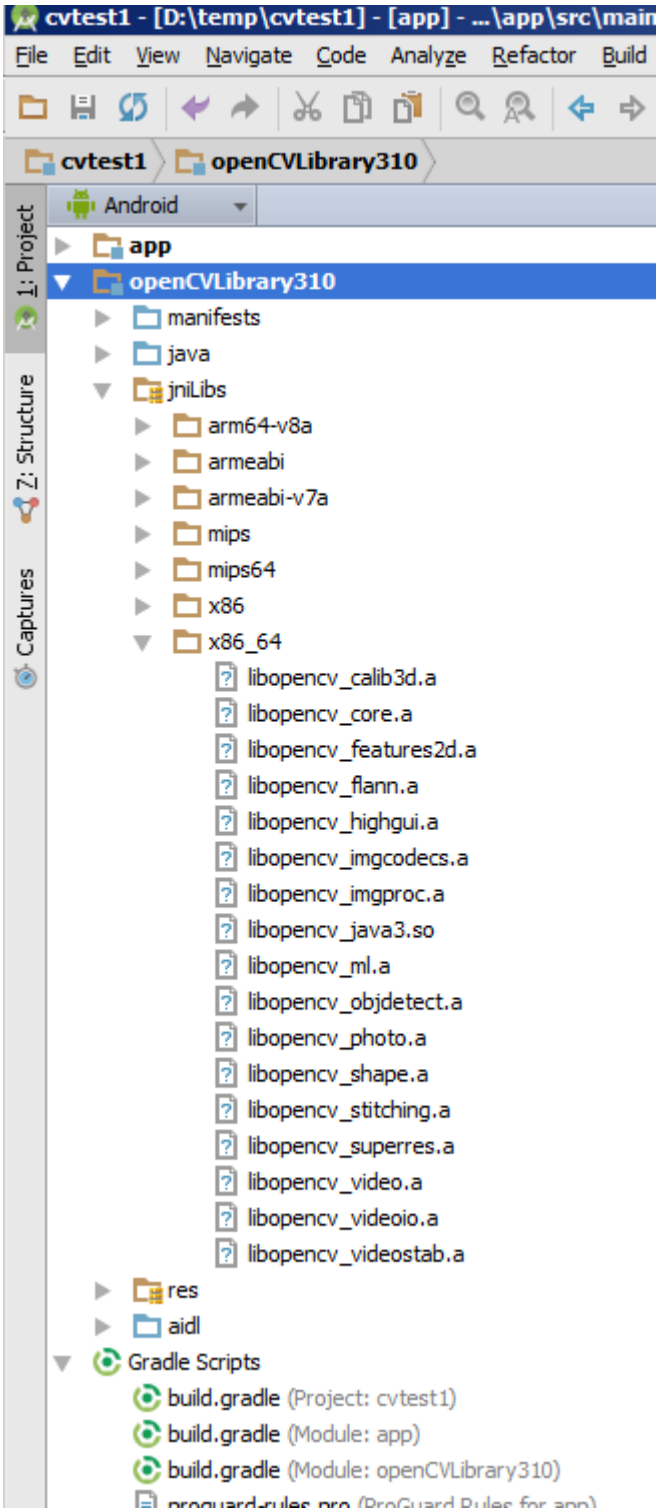
En este punto, el módulo debe aparecer en la jerarquía del proyecto como **openCVLibrary310** , al igual que la **aplicación** . (Tenga en cuenta que en ese pequeño menú desplegable cambié de nuevo desde la **vista de proyecto a la vista** de **Android**). También debería ver un archivo **build.gradle** adicional en "Gradle Scripts" pero encuentro que la interfaz de Android Studio está un poco dañada y que a veces no lo hace de inmediato. Así que intenta volver a sincronizar, limpiar e incluso reiniciar Android Studio.

Debería ver el módulo openCVLibrary310 con todas las funciones de OpenCV bajo java

como en esta captura de pantalla:



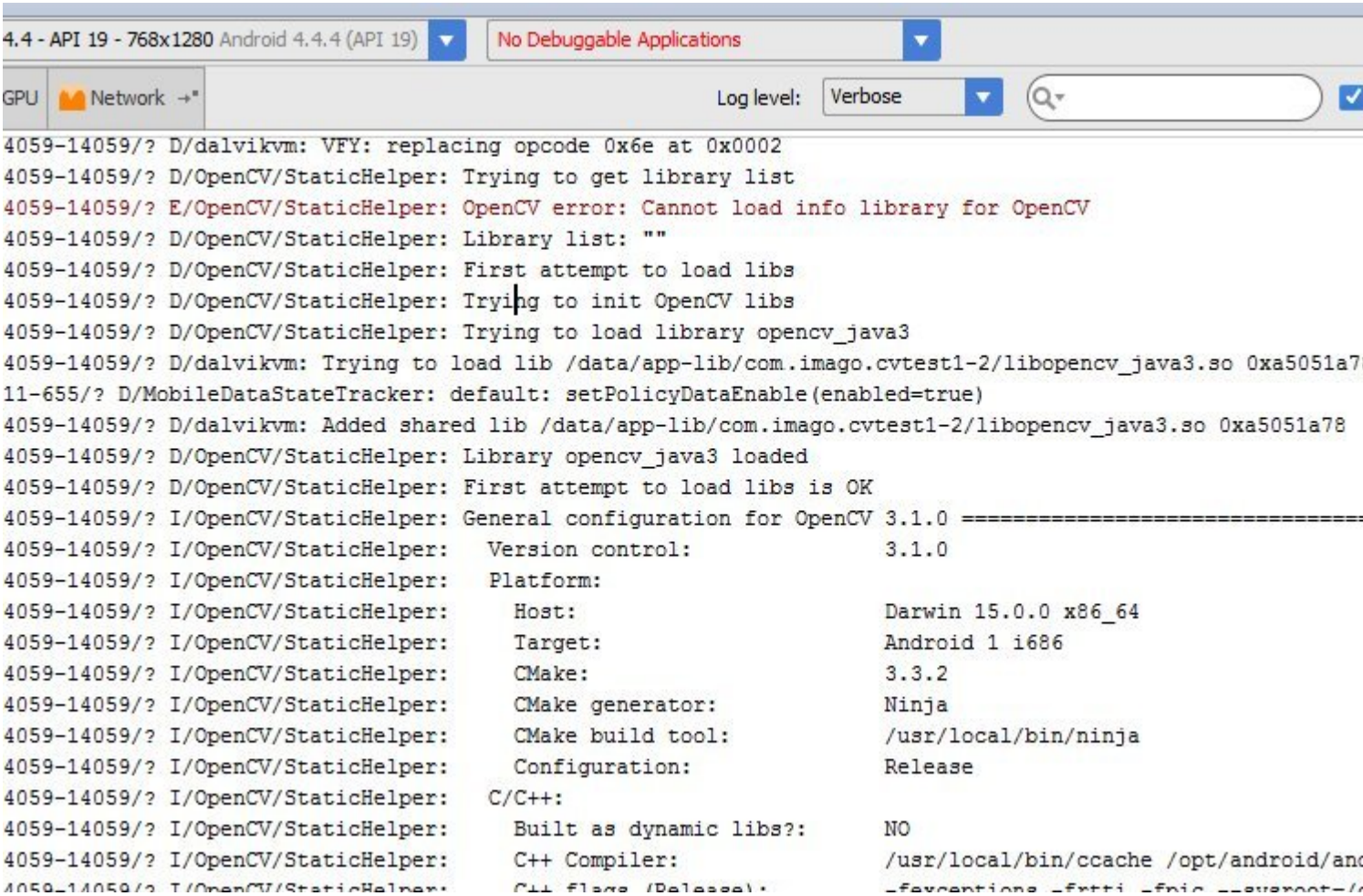
9. Copie el **directorio {unzip-dir} / sdk / native / libs** (y todo lo que **contiene**) a su proyecto de Android, a **cvtest1 / OpenCVLibrary310 / src / main /** , y luego cambie el nombre de su copia de **libs** a **jniLibs** . Ahora debería tener un **directorio cvtest1 / OpenCVLibrary310 / src / main / jniLibs** . **Vuelva** a **sincronizar** su proyecto y este directorio debería aparecer ahora en la vista del proyecto en **openCVLibrary310** .



10. Vaya al método onCreate de *MainActivity.java* y agregue este código:

```
if (!OpenCVLoader.initDebug()) {
        Log.e(this.getClass().getSimpleName(), "  OpenCVLoader.initDebug(), not working.");
    } else {
        Log.d(this.getClass().getSimpleName(), "  OpenCVLoader.initDebug(), working.");
    }
```

Luego ejecuta tu aplicación. Debería ver líneas como esta en el Monitor de Android:



(No sé por qué está esa línea con el mensaje de error)

11. Ahora trata de usar realmente algún código openCV. En el siguiente ejemplo, copié un archivo .jpg al directorio de caché de la aplicación cvtest1 en el emulador de Android. El siguiente código carga esta imagen, ejecuta el algoritmo de detección de bordes y luego vuelve a escribir los resultados en un archivo .png en el mismo directorio.

```
Put this code just below the code from the previous step and alter it to match your own
files/directories.

    String inputFileName="simm_01";
    String inputExtension = "jpg";
    String inputDir = getCacheDir().getAbsolutePath();  // use the cache directory for i/o
    String outputDir = getCacheDir().getAbsolutePath();
    String outputExtension = "png";
    String inputFilePath = inputDir + File.separator + inputFileName + "." + inputExtension;


    Log.d (this.getClass().getSimpleName(), "loading " + inputFilePath + "...");
    Mat image = Imgcodecs.imread(inputFilePath);
    Log.d (this.getClass().getSimpleName(), "width of " + inputFileName + ": " +
image.width());
    // if width is 0 then it did not read your image.


    // for the canny edge detection algorithm, play with these to see different results
    int threshold1 = 70;
    int threshold2 = 100;
```

```
    Mat im_canny = new Mat();  // you have to initialize output image before giving it to the
Canny method
    Imgproc.Canny(image, im_canny, threshold1, threshold2);
    String cannyFilename = outputDir + File.separator + inputFileName + "_canny-" + threshold1
+ "-" + threshold2 + "." + outputExtension;
    Log.d (this.getClass().getSimpleName(), "Writing " + cannyFilename);
    Imgcodecs.imwrite(cannyFilename, im_canny);
```

12. Ejecute su aplicación. Su emulador debe crear una imagen de "borde" en blanco y negro. Puede utilizar el Monitor de dispositivo de Android para recuperar la salida o escribir una actividad para mostrarla.

Lea Integrar OpenCV en Android Studio en línea:
https://riptutorial.com/es/android/topic/7068/integrar-opencv-en-android-studio

Capítulo 139: Intención

Introducción

Un intento es un pequeño mensaje que pasa alrededor del sistema Android. Este mensaje puede contener información sobre nuestra intención de realizar una tarea.

Es básicamente una estructura de datos pasiva que contiene una descripción abstracta de una acción a realizar.

Sintaxis

- Intención Intención ()
- Intención Intención (Intención intención)
- Intención Intención (acción de cuerdas)
- Intención Intención (String action, Uri uri)
- Intención Intención (Context packageContext, Class <?> Cls)
- Intención Intención (Acción de cadena, Uri uri, Context packageContext, Class <?> Cls)
- void startActivity (Intención de intención)
- void startActivity (Intención de intento, opciones de paquete)
- void startActivityForResult (Intención de intención, int requestCode)
- void startActivityForResult (Intención de intento, int requestCode, opciones de paquete)
- Intención putExtra (nombre de cadena, doble [] valor)
- Intención putExtra (nombre de cadena, valor int)
- Intención putExtra (nombre de cadena, valor CharSequence)
- Intención putExtra (nombre de cadena, valor char)
- Intención putExtra (nombre de cadena, valor de paquete)
- Intención putExtra (nombre de cadena, valor parcelable [])
- Intención putExtra (nombre de cadena, valor serializable)
- Intención putExtra (nombre de cadena, valor int [])
- Intención putExtra (nombre de cadena, valor flotante)
- Intención putExtra (nombre de cadena, byte [] valor)
- Intención putExtra (nombre de cadena, valor largo [])
- Intención putExtra (nombre de cadena, valor parcelable)
- Intención putExtra (nombre de la cadena, valor [] flotante)
- Intención putExtra (nombre de cadena, valor largo)
- Intención putExtra (nombre de cadena, cadena [] valor)
- Intención putExtra (nombre de cadena, valor booleano)
- Intención putExtra (nombre de cadena, valor booleano [])
- Intención putExtra (nombre de cadena, valor corto)
- Intención putExtra (nombre de cadena, doble valor)
- Intención putExtra (nombre de cadena, valor corto [])
- Intención putExtra (nombre de cadena, valor de cadena)
- Intención putExtra (nombre de cadena, valor de byte)

- Intención putExtra (nombre de cadena, valor char [])
- Intención putExtra (nombre de cadena, valor CharSequence [])

Parámetros

Parámetro	Detalles
intención	La intención de empezar.
código de solicitud	Número único para identificar la solicitud.
opciones	Opciones adicionales sobre cómo debe iniciarse la actividad.
nombre	El nombre de los datos extra.
valor	El valor de los datos extra.
CHOOSE_CONTACT_REQUEST_CODE	El código de la solicitud, para identificarlo en el método `onActivityResult`
acción	Cualquier acción a realizar a través de esta intención, por ejemplo: Intent.ACTION_VIEW
uri	datos uri para ser utilizados por la intención de realizar una acción específica
packageContext	Contexto a utilizar para inicializar la Intención.
cls	Clase a utilizar por esta intención.

Observaciones

Advertencias de usar la intención implícita

Cuando se llama a una intención implícita, siempre es útil verificar si el sistema lo puede manejar.

Esto se puede hacer verificando utilizando `PackageManager.queryIntentActivities(Intent intent, int flags)`

```
PackageManager pm = getActivity().getPackageManager();
if (intent.resolveActivity(pm) != null) {
    //intent can be handled
    startActivity(intent);
} else {
     //intent can not be handled
}
```

Actividad de inicio que es una `singleTask` o una `singleTop`

Cuando el modo de inicio de la actividad es `singleTask` o `singleTop` , se `onActivityResult` tan pronto como se inicie la actividad con un dato nulo. Para evitar esto, use `Intent.setFlags(0)` para restablecer los indicadores predeterminados.

Examples

Iniciar una actividad

Este ejemplo iniciará `DestinationActivity` desde `OriginActivity` .

Aquí, el constructor `Intent` toma dos parámetros:

1. Un contexto como su primer parámetro (esto se usa porque la clase de actividad es una subclase de contexto)
2. La clase del componente de la aplicación a la que el sistema debe entregar la intención (en este caso, la actividad que debe iniciarse)

```
public class OriginActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_origin);

        Intent intent = new Intent(this, DestinationActivity.class);

        startActivity(intent);
        finish(); // Optionally, you can close OriginActivity. In this way when the user press
back from DestinationActivity he/she won't land on OriginActivity again.
    }
}
```

Otra forma de crear el `Intent` para abrir `DestinationActivity` es usar el constructor predeterminado para el `Intent` , y usar el método `setClass()` para decirle qué Actividad abrir:

```
Intent i=new Intent();
i.setClass(this, DestinationActivity.class);
startActivity(intent);
finish(); // Optionally, you can close OriginActivity. In this way when the user press back
from DestinationActivity he/she won't land on OriginActivity
```

Pasando datos entre actividades.

Este ejemplo ilustra el envío de una `String` con valor como `"Some data!"` de `OriginActivity` a `DestinationActivity` .

NOTA: Esta es la forma más sencilla de enviar datos entre dos actividades. Vea el ejemplo sobre el uso del patrón de inicio para una implementación más robusta.

OrigenActividad

```
public class OriginActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_origin);

        // Create a new Intent object, containing DestinationActivity as target Activity.
        final Intent intent = new Intent(this, DestinationActivity.class);

        // Add data in the form of key/value pairs to the intent object by using putExtra()
        intent.putExtra(DestinationActivity.EXTRA_DATA, "Some data!");

        // Start the target Activity with the intent object
        startActivity(intent);
    }
}
```

DestinoActividad

```
public class DestinationActivity extends AppCompatActivity {

    public static final String EXTRA_DATA = "EXTRA_DATA";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_destination);

        // getIntent() returns the Intent object which was used to start this Activity
        final Intent intent = getIntent();

        // Retrieve the data from the intent object by using the same key that
        // was previously used to add data to the intent object in OriginActivity.
        final String data = intent.getStringExtra(EXTRA_DATA);
    }
}
```

También es posible pasar otros `primitive` tipos de datos, así como `arrays` , `Bundle` y `Parcelable` datos. También es posible pasar a `Serializable` , pero debe evitarse ya que es más de tres veces más lento que `Parcelable` .

Serializable es una `interface` estándar de Java. Simplemente marque una clase como `Serializable` implementando la `interface Serializable` y Java lo serializará automáticamente durante las situaciones requeridas.

Parcelable es una `interface` específica de Android que puede implementarse en tipos de datos

personalizados (es decir, sus propios objetos / objetos POJO), permite que su objeto se aplane y se reconstruya sin que el destino tenga que hacer nada. Hay un ejemplo de documentación de hacer un objeto parcelable .

Una vez que tenga un objeto `parcelable` , puede enviarlo como un tipo primitivo, con un objeto de intención:

```
intent.putExtra(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

O en un paquete / como un argumento para un fragmento:

```
bundle.putParcelable(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

y luego también leerlo desde la intención en el destino usando getParcelableExtra:

```
final MyParcelableType data = intent.getParcelableExtra(EXTRA_DATA);
```

O al leer en un fragmento de un paquete:

```
final MyParcelableType data = bundle.getParcelable(EXTRA_DATA);
```

Una vez que tenga un objeto `Serializable` , puede ponerlo en un objeto de intención:

```
bundle.putSerializable(DestinationActivity.EXTRA_DATA, mySerializableObject);
```

y luego léalo del objeto de intención en el destino como se muestra a continuación:

```
final SerializableType data = (SerializableType)bundle.getSerializable(EXTRA_DATA);
```

Mandando correos electrónicos

```
// Compile a Uri with the 'mailto' schema
Intent emailIntent = new Intent(Intent.ACTION_SENDTO, Uri.fromParts(
        "mailto","johndoe@example.com", null));
// Subject
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Hello World!");
// Body of email
emailIntent.putExtra(Intent.EXTRA_TEXT, "Hi! I am sending you a test email.");
// File attachment
emailIntent.putExtra(Intent.EXTRA_STREAM, attachedFileUri);

// Check if the device has an email client
if (emailIntent.resolveActivity(getPackageManager()) != null) {
     // Prompt the user to select a mail app
     startActivity(Intent.createChooser(emailIntent,"Choose your mail application"));
} else {
    // Inform the user that no email clients are installed or provide an alternative
}
```

Esto completará previamente un correo electrónico en una aplicación de correo de la elección del

usuario.

Si necesita agregar un archivo adjunto, puede usar `Intent.ACTION_SEND` lugar de `Intent.ACTION_SENDTO` . Para varios archivos adjuntos puedes usar `ACTION_SEND_MULTIPLE`

Una palabra de advertencia: no todos los dispositivos tienen un proveedor para `ACTION_SENDTO` , y llamar a `startActivity()` sin verificar con `resolveActivity()` primero puede lanzar una ActivityNotFoundException.

Obtener un resultado de otra actividad

Al usar `startActivityForResult(Intent intent, int requestCode)` puede iniciar otra `Activity` y luego recibir un resultado de esa `Activity` en el `onActivityResult(int requestCode, int resultCode, Intent data)` . El resultado será devuelto como `Intent` . Un intento puede contener datos a través de un paquete

---

En este ejemplo, `MainActivity` iniciará una `DetailActivity` y luego esperará un resultado. Cada tipo de solicitud debe tener su propia `int` código de petición, de modo que en el *reemplazado* `onActivityResult(int requestCode, int resultCode, Intent data)` método en el `MainActivity` , se puede determinar que la solicitud para procesar mediante la comparación de valores de `requestCode` y `REQUEST_CODE_EXAMPLE` (aunque en este ejemplo, solo hay uno).

Actividad principal:

```
public class MainActivity extends Activity {

    // Use a unique request code for each use case
    private static final int REQUEST_CODE_EXAMPLE = 0x9345;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Create a new instance of Intent to start DetailActivity
        final Intent intent = new Intent(this, DetailActivity.class);

        // Start DetailActivity with the request code
        startActivityForResult(intent, REQUEST_CODE_EXAMPLE);
    }

    // onActivityResult only get called
    // when the other Activity previously started using startActivityForResult
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        // First we need to check if the requestCode matches the one we used.
        if(requestCode == REQUEST_CODE_EXAMPLE) {

            // The resultCode is set by the DetailActivity
            // By convention RESULT_OK means that whatever
            // DetailActivity did was executed successfully
```

```
            if(resultCode == Activity.RESULT_OK) {
                // Get the result from the returned Intent
                final String result = data.getStringExtra(DetailActivity.EXTRA_DATA);

                // Use the data - in this case, display it in a Toast.
                Toast.makeText(this, "Result: " + result, Toast.LENGTH_LONG).show();
            } else {
                // setResult wasn't successfully executed by DetailActivity
                // Due to some error or flow of control. No data to retrieve.
            }
        }
    }
}
```

DetailActividad:

```
public class DetailActivity extends Activity {

    // Constant used to identify data sent between Activities.
    public static final String EXTRA_DATA = "EXTRA_DATA";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detail);

        final Button button = (Button) findViewById(R.id.button);
        // When this button is clicked we want to return a result
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Create a new Intent object as container for the result
                final Intent data = new Intent();

                // Add the required data to be returned to the MainActivity
                data.putExtra(EXTRA_DATA, "Some interesting data!");

                // Set the resultCode as Activity.RESULT_OK to
                // indicate a success and attach the Intent
                // which contains our result data
                setResult(Activity.RESULT_OK, data);

                // With finish() we close the DetailActivity to
                // return back to MainActivity
                finish();
            }
        });
    }

    @Override
    public void onBackPressed() {
        // When the user hits the back button set the resultCode
        // as Activity.RESULT_CANCELED to indicate a failure
        setResult(Activity.RESULT_CANCELED);
        super.onBackPressed();
    }
}
```

Algunas cosas que debes tener en cuenta:

- Los datos solo se devuelven una vez que llama a `finish()` . `setResult()` llamar a `setResult()` antes de llamar a `finish()` , de lo contrario, no se devolverá ningún resultado.
- Asegúrese de que su `Activity` no esté usando `android:launchMode="singleTask"` , o la `Activity` ejecutará en una tarea separada y, por lo tanto, no recibirá ningún resultado de ella. Si su `Activity` utiliza `singleTask` como modo de inicio, activará `onActivityResult()` inmediatamente con un código de resultado de `Activity.RESULT_CANCELED` .
- Tenga cuidado al usar `android:launchMode="singleInstance"` . En los dispositivos anteriores a Lollipop (Android 5.0, nivel API 21), Actividades no devolverá un resultado.
- Puede usar intentos explícitos o implícitos cuando llama a `startActivityForResult()` . Al iniciar una de sus propias actividades para recibir un resultado, debe utilizar un intento explícito para asegurarse de que recibe el resultado esperado. Una `intent` explícita siempre se entrega a su objetivo, sin importar lo que contenga; El `filter` no es consultado. Pero un intento implícito se entrega a un componente solo si puede pasar a través de uno de los filtros del componente.

Abre una URL en un navegador

Apertura con el navegador predeterminado

Este ejemplo muestra cómo puede abrir una URL programáticamente en el navegador web incorporado en lugar de dentro de su aplicación. Esto permite que su aplicación abra una página web sin la necesidad de incluir el permiso de `INTERNET` en su archivo de manifiesto.

```
public void onBrowseClick(View v) {
    String url = "http://www.google.com";
    Uri uri = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    // Verify that the intent will resolve to an activity
    if (intent.resolveActivity(getPackageManager()) != null) {
        // Here we use an intent without a Chooser unlike the next example
        startActivity(intent);
    }
}
```

Pedir al usuario que seleccione un navegador

Tenga en cuenta que este ejemplo utiliza el método `Intent.createChooser()` :

```
public void onBrowseClick(View v) {
    String url = "http://www.google.com";
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
```

```
    // Note the Chooser below. If no applications match,
    // Android displays a system message.So here there is no need for try-catch.
    startActivity(Intent.createChooser(intent, "Browse with"));

}
```

En algunos casos, la URL puede comenzar con **"www"** . Si ese es el caso, obtendrás esta excepción:

> `android.content.ActivityNotFoundException` : no se encontró ninguna actividad para manejar la intención

La URL siempre debe comenzar con **"http: //"** o **"https: //"** . Por lo tanto, su código debe verificarlo, como se muestra en el siguiente fragmento de código:

```
if (!url.startsWith("https://") && !url.startsWith("http://")){
    url = "http://" + url;
}
Intent openUrlIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
if (openUrlIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(openUrlIntent);
}
```

Mejores prácticas

Compruebe si no hay aplicaciones en el dispositivo que puedan recibir la intención implícita. De lo contrario, su aplicación se bloqueará cuando llame a `startActivity()` . Para verificar primero que exista una aplicación para recibir la intención, llame a `resolveActivity()` en su objeto Intención. Si el resultado no es nulo, hay al menos una aplicación que puede manejar la intención y es seguro llamar a `startActivity()` . Si el resultado es nulo, no debe usar la intención y, si es posible, debe deshabilitar la función que invoca la intención.

Borrar una pila de actividades

A veces, es posible que desee iniciar una nueva actividad mientras elimina actividades anteriores de la pila trasera, para que el botón Atrás no lo lleve de vuelta a ellas. Un ejemplo de esto podría ser iniciar una aplicación en la actividad de inicio de sesión, que lo lleve a la actividad principal de su aplicación, pero al cerrar la sesión desea volver a iniciar sesión sin tener la posibilidad de volver. En un caso así, puede establecer el indicador `FLAG_ACTIVITY_CLEAR_TOP` para la intención, lo que significa que si la actividad que se está iniciando ya se está ejecutando en la tarea actual (LoginActivity), en lugar de lanzar una nueva instancia de esa actividad, todas las otras actividades en la parte superior se cerrará y esta Intención se entregará a la actividad antigua (ahora arriba) como una Intención nueva.

```
Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
```

También es posible usar los indicadores `FLAG_ACTIVITY_NEW_TASK` junto con `FLAG_ACTIVITY_CLEAR_TASK`

si desea borrar todas las Actividades en la pila de atrás:

```
Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
// Closing all the Activities, clear the back stack.
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(intent);
```

Intención URI

Este ejemplo muestra, cómo iniciar la intención desde el navegador:

```
<a href="intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end">Start
intent</a>
```

Este intento iniciará la aplicación con el paquete `com.sample.test` o abrirá Google Play con este paquete.

También esta intención se puede iniciar con javascript:

```
var intent = "intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end";
window.location.replace(intent)
```

En la actividad, este host y la ruta se pueden obtener a partir de los datos de intención:

```
@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    Uri data = getIntent().getData(); // returns host.com/path
}
```

Sintaxis de URI de intención:

```
HOST/URI-path // Optional host
#Intent;
    package=[string];
    action=[string];
    category=[string];
    component=[string];
    scheme=[string];
end;
```

Transmisión de mensajes a otros componentes

Los intentos se pueden usar para transmitir mensajes a otros componentes de su aplicación (como un servicio en segundo plano en ejecución) o al sistema Android completo.

Para enviar una transmisión **dentro de su aplicación** , use la clase `LocalBroadcastManager` :

```
Intent intent = new Intent("com.example.YOUR_ACTION"); // the intent action
intent.putExtra("key", "value"); // data to be passed with your broadcast
```

```
LocalBroadcastManager manager = LocalBroadcastManager.getInstance(context);
manager.sendBroadcast(intent);
```

Para enviar una difusión a componentes fuera de su aplicación, use el método `sendBroadcast()` en un objeto de `Context` .

```
Intent intent = new Intent("com.example.YOUR_ACTION"); // the intent action
intent.putExtra("key", "value"); // data to be passed with your broadcast

context.sendBroadcast(intent);
```

La información sobre las transmisiones de *recepción* se puede encontrar aquí: Receptor de difusión

CustomTabsIntent para Chrome Custom Tabs

4.0.3

Al usar un `CustomTabsIntent` , ahora es posible configurar las pestañas personalizadas de Chrome para personalizar los componentes clave de la interfaz de usuario en el navegador que se abre desde su aplicación.

Esta es una buena alternativa al uso de una vista web para algunos casos. Permite la carga de una página web con una intención, con la capacidad adicional de inyectar cierto grado de apariencia de su aplicación en el navegador.

Aquí hay un ejemplo de cómo abrir una url usando `CustomTabsIntent`

```
String url = "https://www.google.pl/";
CustomTabsIntent intent = new CustomTabsIntent.Builder()
                    .setStartAnimations(getContext(), R.anim.slide_in_right,
R.anim.slide_out_left)
                    .setExitAnimations(getContext(), android.R.anim.slide_in_left,
android.R.anim.slide_out_right)
                    .setCloseButtonIcon(BitmapFactory.decodeResource(getResources(),
R.drawable.ic_arrow_back_white_24dp))
                    .setToolbarColor(Color.parseColor("#43A047"))
                    .enableUrlBarHiding()
                    .build();
            intent.launchUrl(getActivity(), Uri.parse(url));
```

Nota:

Para usar pestañas personalizadas, debe agregar esta dependencia a su build.gradle

```
compile 'com.android.support:customtabs:24.1.1'
```

Compartiendo múltiples archivos a través de la intención

La Lista de cadenas que se pasa como parámetro al método `share()` contiene las rutas de todos los archivos que desea compartir.

Básicamente recorre las rutas, las agrega a Uri e inicia la Actividad que puede aceptar archivos de este tipo.

```
  public static void share(AppCompatActivity context,List<String> paths) {

        if (paths == null || paths.size() == 0) {
            return;
        }
        ArrayList<Uri> uris = new ArrayList<>();
        Intent intent = new Intent();
        intent.setAction(android.content.Intent.ACTION_SEND_MULTIPLE);
        intent.setType("*/*");
        for (String path : paths) {
                File file = new File(path);
                uris.add(Uri.fromFile(file));
        }
        intent.putParcelableArrayListExtra(Intent.EXTRA_STREAM, uris);
        context.startActivity(intent);
    }
```

Patrón de arranque

Este patrón es un enfoque más estricto para iniciar una `Activity` . Su propósito es mejorar la legibilidad del código, mientras que al mismo tiempo disminuye la complejidad del código, los costos de mantenimiento y el acoplamiento de sus componentes.

El siguiente ejemplo implementa el patrón de inicio, que generalmente se implementa como un método estático en la propia `Activity` . Este método estático acepta todos los parámetros requeridos, construye un `Intent` válido a partir de esos datos y luego inicia la `Activity` .

Un `Intent` es un objeto que proporciona un enlace de tiempo de ejecución entre componentes separados, como dos actividades. La intención representa la "intención de hacer algo" de una aplicación. Puede usar los intentos para una amplia variedad de tareas, pero aquí, su intento inicia otra actividad.

```
public class ExampleActivity extends AppCompatActivity {

    private static final String EXTRA_DATA = "EXTRA_DATA";

    public static void start(Context context, String data) {
        Intent intent = new Intent(context, ExampleActivity.class);
        intent.putExtra(EXTRA_DATA, data);
        context.startActivity(intent);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Intent intent = getIntent();
        if(!intent.getExtras().containsKey(EXTRA_DATA)){
            throw new UnsupportedOperationException("Activity should be started using the
static start method");
        }
        String data = intent.getStringExtra(EXTRA_DATA);
```

```
    }
}
```

Este patrón también le permite forzar la transmisión de datos adicionales con la intención.

La actividad de `ExampleActivity` puede iniciarse así, donde el `context` es un contexto de actividad:

```
ExampleActivity.start(context, "Some data!");
```

Inicia el servicio Unbound usando una intención

Un servicio es un componente que se ejecuta en segundo plano (en el hilo de la interfaz de usuario) sin interacción directa con el usuario. Un servicio no consolidado acaba de iniciarse y no está vinculado al ciclo de vida de ninguna actividad.

Para iniciar un servicio, puede hacer lo que se muestra en el siguiente ejemplo:

```
// This Intent will be used to start the service
Intent i= new Intent(context, ServiceName.class);
// potentially add data to the intent extras
i.putExtra("KEY1", "Value to be used by the service");
context.startService(i);
```

Puede usar cualquier extra de la intención utilizando una `onStartCommand()` :

```
public class MyService extends Service {
    public MyService() {
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId)
    {
        if (intent != null) {
            Bundle extras = intent.getExtras();
            String key1 = extras.getString("KEY1", "");
            if (key1.equals("Value to be used by the service")) {
                //do something
            }
        }
        return START_STICKY;
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

}
```

Compartir intención

Comparte información simple con diferentes aplicaciones.

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, "This is my text to send.");
sendIntent.setType("text/plain");
startActivity(Intent.createChooser(sendIntent, getResources().getText(R.string.send_to)));
```

Comparte una imagen con diferentes aplicaciones.

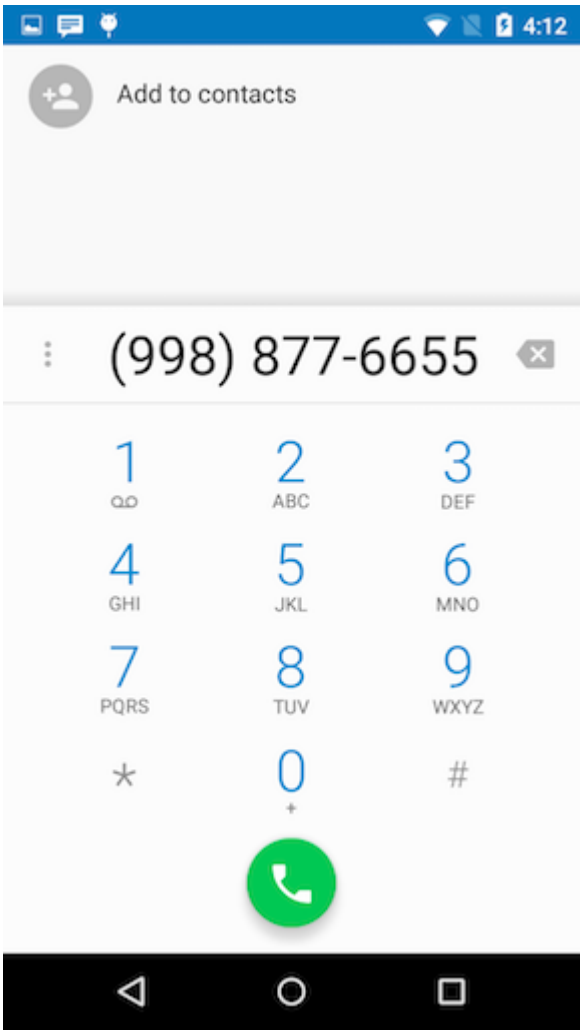
```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uriToImage);
shareIntent.setType("image/jpeg");
startActivity(Intent.createChooser(shareIntent, getResources().getText(R.string.send_to)));
```

Iniciar el marcador

Este ejemplo muestra cómo abrir un marcador predeterminado (una aplicación que realiza llamadas regulares) con un número de teléfono proporcionado que ya está en su lugar:

```
Intent intent = new Intent(Intent.ACTION_DIAL);
intent.setData(Uri.parse("tel:9988776655")); //Replace with valid phone number. Remember to
add the tel: prefix, otherwise it will crash.
startActivity(intent);
```

Resultado de ejecutar el código anterior:



Abrir el mapa de Google con la latitud, longitud especificada

Puede pasar latitud, longitud desde su aplicación a Google map usando Intent

```
String uri = String.format(Locale.ENGLISH, "http://maps.google.com/maps?q=loc:%f,%f",
28.43242324,77.8977673);
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri));
startActivity(intent);
```

Pasando diferentes datos a través de Intención en Actividad.

1. Pasando datos enteros:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("intVariableName", intValue);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
int intValue = mIntent.getIntExtra("intVariableName", 0); // set 0 as the default value if no
```

```
value for intVariableName found
```

2. Pasando datos dobles:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("doubleVariableName", doubleValue);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
double doubleValue = mIntent.getDoubleExtra("doubleVariableName", 0.00); // set 0.00 as the
default value if no value for doubleVariableName found
```

3. Pasando datos de la cadena:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("stringVariableName", stringValue);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
String stringValue = mIntent.getExtras().getString("stringVariableName");
```

o

```
Intent mIntent = getIntent();
String stringValue = mIntent.getStringExtra("stringVariableName");
```

4. Pasando los datos de ArrayList:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putStringArrayListExtra("arrayListVariableName", arrayList);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
arrayList = mIntent.getStringArrayListExtra("arrayListVariableName");
```

5. Pasando datos del objeto:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("ObjectVariableName", yourObject);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
yourObj = mIntent.getSerializableExtra("ObjectVariableName");
```

> **Nota:** tenga en cuenta que su clase personalizada debe implementar la interfaz `Serializable` .

6. Pasando HashMap <String, String> datos:

SenderActivity

HashMap <String, String> hashMap;

```
Intent mIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
mIntent.putExtra("hashMap", hashMap);
startActivity(mIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
HashMap<String, String> hashMap = (HashMap<String, String>)
mIntent.getSerializableExtra("hashMap");
```

7. Pasando datos de mapa de bits:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("image",bitmap);
startActivity(mIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
Bitmap bitmap = mIntent.getParcelableExtra("image");
```

Mostrar un selector de archivos y leer el resultado

Iniciar una actividad de selección de archivos

```
public void showFileChooser() {
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);

    // Update with mime types
    intent.setType("*/*");
```

```
    // Update with additional mime types here using a String[].
    intent.putExtra(Intent.EXTRA_MIME_TYPES, mimeTypes);

    // Only pick openable and local files. Theoretically we could pull files from google drive
    // or other applications that have networked files, but that's unnecessary for this
example.
    intent.addCategory(Intent.CATEGORY_OPENABLE);
    intent.putExtra(Intent.EXTRA_LOCAL_ONLY, true);

    // REQUEST_CODE = <some-integer>
    startActivityForResult(intent, REQUEST_CODE);
}
```

Leyendo el resultado

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // If the user doesn't pick a file just return
    if (requestCode != REQUEST_CODE || resultCode != RESULT_OK) {
        return;
    }

    // Import the file
    importFile(data.getData());
}

public void importFile(Uri uri) {
    String fileName = getFileName(uri);

    // The temp file could be whatever you want
    File fileCopy = copyToTempFile(uri, File tempFile)

    // Done!
}

/**
 * Obtains the file name for a URI using content resolvers. Taken from the following link
 * https://developer.android.com/training/secure-file-sharing/retrieve-
info.html#RetrieveFileInfo
 *
 * @param uri a uri to query
 * @return the file name with no path
 * @throws IllegalArgumentException if the query is null, empty, or the column doesn't exist
 */
private String getFileName(Uri uri) throws IllegalArgumentException {
    // Obtain a cursor with information regarding this uri
    Cursor cursor = getContentResolver().query(uri, null, null, null, null);

    if (cursor.getCount() <= 0) {
        cursor.close();
        throw new IllegalArgumentException("Can't obtain file name, cursor is empty");
    }

    cursor.moveToFirst();

    String fileName =
cursor.getString(cursor.getColumnIndexOrThrow(OpenableColumns.DISPLAY_NAME));
```

```
    cursor.close();

    return fileName;
}

/**
 * Copies a uri reference to a temporary file
 *
 * @param uri      the uri used as the input stream
 * @param tempFile the file used as an output stream
 * @return the input tempFile for convenience
 * @throws IOException if an error occurs
 */
private File copyToTempFile(Uri uri, File tempFile) throws IOException {
    // Obtain an input stream from the uri
    InputStream inputStream = getContentResolver().openInputStream(uri);

    if (inputStream == null) {
        throw new IOException("Unable to obtain input stream from URI");
    }

    // Copy the stream to the temp file
    FileUtils.copyInputStreamToFile(inputStream, tempFile);

    return tempFile;
}
```

Pasando objeto personalizado entre actividades.

También es posible pasar su objeto personalizado a otras actividades usando la clase `Bundle` .

Hay dos maneras:

- Interfaz `Serializable` : para Java y Android
- Interfaz `Parcelable` : memoria eficiente, solo para Android (recomendado)

Parcelable

El procesamiento parcelable es mucho más rápido que el serializable. Una de las razones de esto es que estamos siendo explícitos sobre el proceso de serialización en lugar de utilizar la reflexión para inferirlo. También es lógico pensar que el código ha sido fuertemente optimizado para este propósito.

```
public class MyObjects implements Parcelable {

    private int age;
    private String name;

    private ArrayList<String> address;

    public MyObjects(String name, int age, ArrayList<String> address) {
        this.name = name;
        this.age = age;
        this.address = address;
```

```
    }

    public MyObjects(Parcel source) {
        age = source.readInt();
        name = source.readString();
        address = source.createStringArrayList();
    }

    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeInt(age);
        dest.writeString(name);
        dest.writeStringList(address);
    }

    public int getAge() {
        return age;
    }

    public String getName() {
        return name;
    }

    public ArrayList<String> getAddress() {
        if (!(address == null))
            return address;
        else
            return new ArrayList<String>();
    }

    public static final Creator<MyObjects> CREATOR = new Creator<MyObjects>() {
        @Override
        public MyObjects[] newArray(int size) {
            return new MyObjects[size];
        }

        @Override
        public MyObjects createFromParcel(Parcel source) {
            return new MyObjects(source);
        }
    };
}
```

Código de actividad de envío

```
MyObject mObject = new MyObject("name","age","Address array here");

//Passing MyOject
Intent mIntent = new Intent(FromActivity.this, ToActivity.class);
mIntent.putExtra("UniqueKey", mObject);
startActivity(mIntent);
```

Recibiendo el objeto en actividad de destino.

```
//Getting MyObjects
Intent mIntent = getIntent();
MyObjects workorder = (MyObjects) mIntent.getParcelable("UniqueKey");
```

Puedes pasar Arraylist of Parceble object como abajo

```
//Array of MyObjects
ArrayList<MyObject> mUsers;

//Passing MyObject List
Intent mIntent = new Intent(FromActivity.this, ToActivity.class);
mIntent.putParcelableArrayListExtra("UniqueKey", mUsers);
startActivity(mIntent);

//Getting MyObject List
Intent mIntent = getIntent();
ArrayList<MyObjects> mUsers = mIntent.getParcelableArrayList("UniqueKey");
```

> **Nota:** Hay complementos de Android Studio como este disponibles para generar código Parcelable

Serializable

Código de actividad de envío

```
Product product = new Product();
Bundle bundle = new Bundle();
bundle.putSerializable("product", product);
Intent cartIntent = new Intent(mContext, ShowCartActivity.class);
cartIntent.putExtras(bundle);
mContext.startActivity(cartIntent);
```

Recibiendo el objeto en actividad de destino.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Bundle bundle = this.getIntent().getExtras();
    Product product = null;
    if (bundle != null) {
        product = (Product) bundle.getSerializable("product");
    }
```

`Arraylist` of Serializable object: igual que un solo objeto pasando

El objeto personalizado debe implementar la interfaz `Serializable` .

Obteniendo un resultado de Actividad a Fragmentar

Como obtener un resultado de otra actividad , debe llamar al método `Fragment startActivityForResult(Intent intent, int requestCode)` . tenga en cuenta que no debe llamar a `getActivity().startActivityForResult()` ya que esto devolverá el resultado a la `Activity` principal

del `Fragment` .

La recepción del resultado se puede hacer usando el método de `Fragment` `onActivityResult()` . `onActivityResult()` asegurarse de que la actividad principal del Fragmento también invalida `onActivityResult()` y llame a la `super` implementación.

En el siguiente ejemplo, `ActivityOne` contiene `FragmentOne` , que iniciará `ActivityTwo` y esperará un resultado.

ActivityOne

```
public class ActivityOne extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_one);
    }

    // You must override this method as the second Activity will always send its results to
this Activity and then to the Fragment
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
       super.onActivityResult(requestCode, resultCode, data);
    }
}
```

activity_one.xml

```
<fragment android:name="com.example.FragmentOne"
    android:id="@+id/fragment_one"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

FragmentOne

```
public class FragmentOne extends Fragment {
    public static final int REQUEST_CODE = 11;
    public static final int RESULT_CODE = 12;
    public static final String EXTRA_KEY_TEST = "testKey";

    // Initializing and starting the second Activity
    private void startSecondActivity() {
        Intent intent = new Intent(getActivity(), ActivityTwo.class);
        startActivityForResult(REQUEST_CODE, intent);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_CODE && resultCode == RESULT_CODE) {
            String testResult = data.getStringExtra(EXTRA_KEY_TEST);
            // TODO: Do something with your extra data
        }
    }
}
```

Actividad dos

```
public class ActivityTwo extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_two);
    }

    private void closeActivity() {
        Intent intent = new Intent();
        intent.putExtra(FragmentOne.EXTRA_KEY_TEST, "Testing passing data back to
ActivityOne");
        setResult(FragmentOne.RESULT_CODE, intent); // You can also send result without any
data using setResult(int resultCode)
        finish();
    }
}
```

Lea Intención en línea: https://riptutorial.com/es/android/topic/103/intencion

Capítulo 140: Intenciones implícitas

Sintaxis

- Intención()
- Intención (Intención o)
- Intención (acción de cadena)
- Intención (String action, Uri uri)
- Intención (Context packageContext, Class <?> Cls)
- Intención (Acción de cadena, Uri uri, Context packageContext, Class <?> Cls)

Parámetros

Parámetros	Detalles
o	Intención
acción	`String`: la acción Intención, como `ACTION_VIEW`.
uri	`Uri`: El URI de datos de Intención.
packageContext	`Context`: un contexto del paquete de aplicación que implementa esta clase.
cls	`Class`: la clase de componente que se utilizará para la intención.

Observaciones

- Más sobre Intención
- Más sobre los tipos de intención

Examples

Intenciones implícitas y explícitas

Se utiliza una intención explícita para iniciar una actividad o servicio dentro del mismo paquete de aplicación. En este caso, el nombre de la clase deseada se menciona explícitamente:

```
Intent intent = new Intent(this, MyComponent.class);
startActivity(intent);
```

Sin embargo, se envía una intención implícita a través del sistema para cualquier aplicación instalada en el dispositivo del usuario que pueda manejar esa intención. Esto se utiliza para compartir información entre diferentes aplicaciones.

```
Intent intent = new Intent("com.stackoverflow.example.VIEW");

//We need to check to see if there is an application installed that can handle this intent
if (getPackageManager().resolveActivity(intent, 0) != null){
    startActivity(intent);
}else{
    //Handle error
}
```

Más detalles sobre las diferencias se pueden encontrar en los documentos de Android Developer aquí: Intención de resolución

Intenciones implícitas

Los intentos implícitos no nombran un componente específico, sino que declaran una acción general para realizar, lo que permite que un componente de otra aplicación lo maneje.

Por ejemplo, si desea mostrarle al usuario una ubicación en un mapa, puede usar un intento implícito para solicitar que otra aplicación capaz muestre una ubicación específica en un mapa.

Ejemplo:

```
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
```

Lea Intenciones implícitas en línea: https://riptutorial.com/es/android/topic/5336/intenciones-implicitas

Capítulo 141: Inter-app UI testing con UIAutomator

Sintaxis

- Instrumentación getInstrumentation ()
- UIDevice UiDevice.getInstance (Instrumentación de instrumentos)
- booleano UIDevice.pressHome ()
- UIDevice.pressBack booleano ()
- UIDevice.pressRecentApps booleano ()
- void UIDevice.wakeUp ()
- booleano UIDevice.swipe (int startX, int startY, int endX, int endY, int pasos)
- booleano UIDevice.drag (int startX, int startY, int endX, int endY, int pasos)
- UIObject2 UIDevice.findObject (By.desc (String contentDesc))
- booleano UIObject2.click ()

Observaciones

UIAutomator es especialmente bueno para probar historias de usuarios. Se encuentra con problemas si los elementos de la vista no tienen una *identificación de recurso* única ni *desc* . En la mayoría de los casos, hay una manera de completar la prueba de todos modos, lo que lleva mucho tiempo. Si puede influir en el código de su aplicación, UIAutomator puede ser su herramienta de prueba.

Examples

Prepara tu proyecto y escribe el primer test UIAutomator.

Agregue las bibliotecas necesarias en la sección de dependencias del build.gradle de su módulo Android:

```
android {
...
  defaultConfig {
    ...
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }
}

dependencies {
  ...
  androidTestCompile 'com.android.support.test:runner:0.5'
  androidTestCompile 'com.android.support.test:rules:0.5'
  androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.1.2'
  androidTestCompile 'com.android.support:support-annotations:23.4.0'
}
```

Tenga en cuenta que, por supuesto, las versiones pueden diferir en el tiempo medio.

Después de esta sincronización con los cambios.

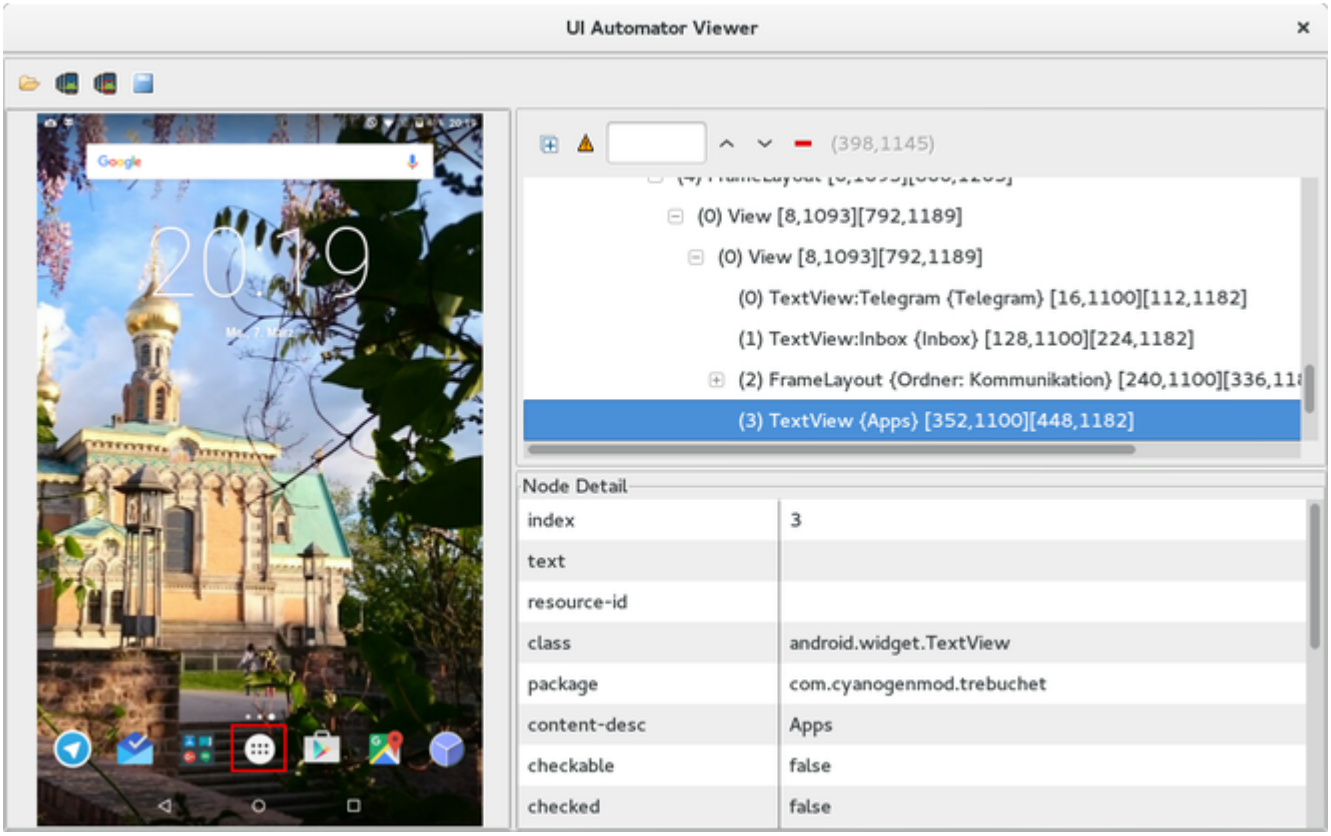
Luego agregue una nueva clase Java dentro de la carpeta androidTest:

```
public class InterAppTest extends InstrumentationTestCase {

  private UiDevice device;

  @Override
  public void setUp() throws Exception {
     device = UiDevice.getInstance(getInstrumentation());
  }

  public void testPressHome() throws Exception {
    device.pressHome();
  }
}
```

Al hacer un clic derecho en la pestaña de clase y en "Ejecutar" InterAppTest "ejecuta esta prueba.

Escribiendo pruebas más complejas usando el UIAutomatorViewer

Para habilitar la escritura de pruebas de IU más complejas, se necesita el *UIAutomatorViewer* . La herramienta ubicada en */ tools /* hace una captura de pantalla completa que incluye los diseños de las vistas que se muestran actualmente. Vea la siguiente imagen para tener una idea de lo que se muestra:



Para las pruebas de interfaz de usuario, estamos buscando *una identificación* de *recursos* , *una*

descripción de contenido o algo más para identificar una vista y usarla dentro de nuestras pruebas.

El *uiautomatorviewer* se ejecuta a través de terminal.

Si ahora, por ejemplo, queremos hacer clic en el botón de aplicaciones y luego abrir alguna aplicación y deslizar alrededor, así es como puede verse el método de prueba:

```
public void testOpenMyApp() throws Exception {
    // wake up your device
    device.wakeUp();

    // switch to launcher (hide the previous application, if some is opened)
    device.pressHome();

    // enter applications menu (timeout=200ms)
    device.wait(Until.hasObject(By.desc(("Apps"))), 200);
    UiObject2 appsButton = device.findObject(By.desc(("Apps")));
    assertNotNull(appsButton);
    appsButton.click();

    // enter some application (timeout=200ms)
    device.wait(Until.hasObject(By.desc(("MyApplication"))), 200);
    UiObject2 someAppIcon = device.findObject(By.desc(("MyApplication")));
    assertNotNull(someAppIcon);
    someAppIcon.click();

    // do a swipe (steps=20 is 0.1 sec.)
    device.swipe(200, 1200, 1300, 1200, 20);
    assertTrue(isSomeConditionTrue)
}
```

Creación de un conjunto de pruebas de pruebas UIAutomator

Poner las pruebas UIAutomator juntas en un conjunto de pruebas es algo rápido:

```
package de.androidtest.myapplication;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({InterAppTest1.class, InterAppTest2.class})
public class AppTestSuite {}
```

Ejecute de forma similar a una sola prueba haciendo clic derecho y ejecutando la suite.

Lea Inter-app UI testing con UIAutomator en línea:
https://riptutorial.com/es/android/topic/6249/inter-app-ui-testing-con-uiautomator

Capítulo 142: Interfaces

Examples

Oyente personalizado

Definir interfaz

```
//In this interface, you can define messages, which will be send to owner.
public interface MyCustomListener {
    //In this case we have two messages,
    //the first that is sent when the process is successful.
    void onSuccess(List<Bitmap> bitmapList);
    //And The second message, when the process will fail.
    void onFailure(String error);
}
```

Crear oyente

En el siguiente paso, debemos definir una variable de instancia en el objeto que enviará la devolución de llamada a través de `MyCustomListener` . Y añadir setter para nuestro oyente.

```
public class SampleClassB {
    private MyCustomListener listener;

    public void setMyCustomListener(MyCustomListener listener) {
        this.listener = listener;
    }
}
```

Implementar oyente

Ahora, en otra clase, podemos crear una instancia de `SampleClassB` .

```
public class SomeActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
    }
}
```

A continuación, podemos configurar nuestro oyente, a `sampleClass` , de dos maneras:

por implementos `MyCustomListener` en nuestra clase:

```
public class SomeActivity extends Activity implements MyCustomListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
        sampleClass.setMyCustomListener(this);
    }

    @Override
    public void onSuccess(List<Bitmap> bitmapList) {

    }

    @Override
    public void onFailure(String error) {

    }
}
```

o simplemente crear una instancia de una clase interna anónima:

```
public class SomeActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
        sampleClass.setMyCustomListener(new MyCustomListener() {

            @Override
            public void onSuccess(List<Bitmap> bitmapList) {

            }

            @Override
            public void onFailure(String error) {

            }
        });
    }
}
```

Oyente del disparador

```
public class SampleClassB {
    private MyCustomListener listener;

    public void setMyCustomListener(MyCustomListener listener) {
        this.listener = listener;
    }

    public void doSomething() {
        fetchImages();
    }

    private void fetchImages() {
        AsyncImagefetch imageFetch = new AsyncImageFetch();
        imageFetch.start(new Response<Bitmap>() {
```

```
            @Override
            public void onDone(List<Bitmap> bitmapList, Exception e) {
                //do some stuff if needed

                //check if listener is set or not.
                if(listener == null)
                    return;
                //Fire proper event. bitmapList or error message will be sent to
                //class which set listener.
                if(e == null)
                    listener.onSuccess(bitmapList);
                else
                    listener.onFailure(e.getMessage());
            }
        });
    }
}
```

Oyente básico

El patrón de "escucha" u "observador" es la estrategia más común para crear devoluciones de llamadas asíncronas en el desarrollo de Android.

```
public class MyCustomObject {

  //1 - Define the interface
  public interface MyCustomObjectListener {
      public void onAction(String action);
  }

  //2 - Declare your listener object
  private MyCustomObjectListener listener;

  // and initialize it in the costructor
  public MyCustomObject() {
    this.listener = null;
 }

 //3 - Create your listener setter
 public void setCustomObjectListener(MyCustomObjectListener listener) {
    this.listener = listener;
 }

 // 4 - Trigger listener event
 public void makeSomething(){
    if (this.listener != null){
       listener.onAction("hello!");
 }
}
```

Ahora en tu actividad:

```
public class MyActivity extends Activity {
   public final String TAG = "MyActivity";

  @Override
  protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);


        MyCustomObject mObj = new MyCustomObject();

        //5 - Implement listener callback
        mObj.setCustomObjectListener(new MyCustomObjectListener() {
          @Override
            public void onAction(String action) {
                Log.d(TAG, "Value: "+action);
            }
        });
    }
}
```

Lea Interfaces en línea: https://riptutorial.com/es/android/topic/1785/interfaces

Capítulo 143: Interfaz nativa de Java para Android (JNI)

Introducción

JNI (Java Native Interface) es una poderosa herramienta que permite a los desarrolladores de Android utilizar el NDK y utilizar el código nativo de C ++ en sus aplicaciones. Este tema describe el uso de la interfaz Java <-> C ++.

Examples

Cómo llamar a funciones en una biblioteca nativa a través de la interfaz JNI

La interfaz nativa de Java (JNI) le permite llamar a funciones nativas desde el código de Java y viceversa. Este ejemplo muestra cómo cargar y llamar a una función nativa a través de JNI, no se utiliza para acceder a los métodos y campos de Java desde el código nativo mediante las funciones de JNI .

Supongamos que tiene una biblioteca nativa llamada `libjniexample.so` en la carpeta `project/libs/<architecture>` y desea llamar a una función de la clase Java de `JNITest` dentro del paquete `com.example.jniexample` .

En la clase JNITest, declare la función así:

```
public native int testJNIfunction(int a, int b);
```

En su código nativo, defina la función así:

```
#include <jni.h>

JNIEXPORT jint JNICALL Java_com_example_jniexample_JNITest_testJNIfunction(JNIEnv *pEnv,
jobject thiz, jint a, jint b)
{
    return a + b;
}
```

El argumento `pEnv` es un puntero al entorno JNI que puede pasar a las funciones JNI para acceder a métodos y campos de objetos y clases de Java. El `thiz` puntero es un `jobject` referencia al objeto de Java que el método nativo fue llamado (o la clase si se trata de un método estático).

En su código Java, en `JNITest` , cargue la biblioteca así:

```
static{
    System.loadLibrary("jniexample");
}
```

Tenga en cuenta la `lib` al principio, y `.so` al final del nombre de archivo se omiten.

Llama a la función nativa desde Java así:

```
JNITest test = new JNITest();
int c = test.testJNIfunction(3, 4);
```

Cómo llamar a un método Java desde código nativo

La interfaz nativa de Java (JNI) le permite llamar a funciones Java desde código nativo. Aquí hay un ejemplo simple de cómo hacerlo:

Código de Java:

```
package com.example.jniexample;
public class JNITest {
    public static int getAnswer(bool) {
        return 42;
    }
}
```

Código nativo:

```
int getTheAnswer()
{
    // Get JNI environment
    JNIEnv *env = JniGetEnv();

    // Find the Java class - provide package ('.' replaced to '/') and class name
    jclass jniTestClass = env->FindClass("com/example/jniexample/JNITest");

    // Find the Java method - provide parameters inside () and return value (see table below
for an explanation of how to encode them)
    jmethodID getAnswerMethod = env->GetStaticMethodID(jniTestClass, "getAnswer", "(Z)I;");

    // Calling the method
    return (int)env->CallStaticObjectMethod(jniTestClass, getAnswerMethod, (jboolean)true);
}
```

Firma del método JNI al tipo Java:

JNI Signature	Tipo de Java
Z	booleano
segundo	byte
do	carbonizarse
S	corto
yo	En t

JNI Signature	Tipo de Java
J	largo
F	flotador
re	doble
L clase completamente calificada;	clase completamente calificada
[tipo	tipo[]

Así que para nuestro ejemplo usamos (Z) I, lo que significa que la función obtiene un valor booleano y devuelve un int.

Método de utilidad en la capa JNI

Este método ayudará a obtener la cadena Java de la cadena C ++.

```
jstring getJavaStringFromCPPString(JNIEnv *global_env, const char* cstring) {

        jstring nullString = global_env->NewStringUTF(NULL);

        if (!cstring) {
            return nullString;
        }

        jclass strClass = global_env->FindClass("java/lang/String");
        jmethodID ctorID = global_env->GetMethodID(strClass, "<init>",
                "([BLjava/lang/String;)V");
        jstring encoding = global_env->NewStringUTF("UTF-8");

        jbyteArray bytes = global_env->NewByteArray(strlen(cstring));
        global_env->SetByteArrayRegion(bytes, 0, strlen(cstring), (jbyte*) cstring);
        jstring str = (jstring) global_env->NewObject(strClass, ctorID, bytes,
                encoding);

        global_env->DeleteLocalRef(strClass);
        global_env->DeleteLocalRef(encoding);
        global_env->DeleteLocalRef(bytes);

        return str;
    }
```

Este método te ayudará a convertir jbyteArray a char

```
char* as_unsigned_char_array(JNIEnv *env, jbyteArray array) {
    jsize length = env->GetArrayLength(array);
    jbyte* buffer = new jbyte[length + 1];

    env->GetByteArrayRegion(array, 0, length, buffer);
    buffer[length] = '\0';

    return (char*) buffer;
}
```

Lea Interfaz nativa de Java para Android (JNI) en línea:
https://riptutorial.com/es/android/topic/8674/interfaz-nativa-de-java-para-android--jni-

Capítulo 144: Internacionalización y localización (I18N y L10N)

Introducción

La internacionalización (i18n) y la localización (L10n) se utilizan para adaptar el software de acuerdo con las diferencias de idiomas, diferencias regionales y público objetivo.

Internacionalización: el proceso de planificación para la futura localización, es decir, hacer que el diseño del software sea flexible en la medida en que pueda adaptarse y adaptarse a los futuros esfuerzos de localización.

Localización: el proceso de adaptación del software a una región / país / mercado en particular (configuración regional).

Observaciones

Para probar la localización de un dispositivo, se puede reiniciar el dispositivo o el emulador en una configuración regional particular utilizando `adb` siguiente manera:

1. Ejecuta adb usando el comando: `adb shell`
2. Ejecute el siguiente comando en el símbolo del sistema adb: `setprop persist.sys.locale [BCP-47 language tag];stop;sleep 5;start` donde [etiqueta de idioma BCP-47] es el código específico del idioma como se describe aquí: códigos BCP47

por ejemplo, para verificar la localización japonesa en la aplicación, use el comando: `setprop persist.sys.locale ja-JP;stop;sleep 5;start`

Examples

Planificación para la localización: habilitar el soporte RTL en Manifiesto

El soporte RTL (de derecha a izquierda) es una parte esencial en la planificación de i18n y L10n. A diferencia del idioma inglés, que está escrito de izquierda a derecha, muchos idiomas como el árabe, el japonés, el hebreo, etc. están escritos de derecha a izquierda. Para atraer a una audiencia más global, es una buena idea planificar sus diseños para que sean compatibles con este idioma desde el principio del proyecto, para que luego sea más fácil agregar localización.

RTL apoyo se puede activar en una aplicación para Android añadiendo la `supportsRtl` etiqueta en el `AndroidManifest` , así:

```
<application
    ...
    android:supportsRtl="true"
    ...>
```

```
...
</application>
```

Planificación para la localización: Añadir soporte RTL en diseños

A partir del SDK 17 (Android 4.2), se agregó soporte RTL en diseños de Android y es una parte esencial de la localización. En el futuro, la notación `left/right` en los diseños debe reemplazarse por la notación `start/end` . Sin embargo, si su proyecto tiene un valor de `minSdk` menor que `17` , entonces se debe usar la notación de `left/right` y de `start/end` en los diseños.

Para diseños relativos, `alignParentStart` y `alignParentEnd` deben usarse, así:

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"/>
</RelativeLayout>
```

Para especificar la gravedad y la gravedad de la disposición, se debe utilizar una notación similar, como así:

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left|start"
        android:gravity="left|start"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|end"
        android:gravity="right|end"/>
```

Los rellenos y márgenes también deben especificarse en consecuencia, así:

```
<include layout="@layout/notification"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="12dp"
    android:layout_marginStart="12dp"
    android:paddingLeft="128dp"
    android:paddingStart="128dp"
    android:layout_toLeftOf="@id/cancel_action"
    android:layout_toStartOf="@id/cancel_action"/>
<include layout="@layout/notification2"
```

```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginRight="12dp"
    android:layout_marginEnd="12dp"
    android:paddingRight="128dp"
    android:paddingEnd="128dp"
    android:layout_toRightOf="@id/cancel_action"
    android:layout_toEndOf="@id/cancel_action"/>
```

Planificación para la localización: diseños de prueba para RTL

Para probar si los diseños que se han creado son compatibles con RTL, haga lo siguiente:

> Vaya a Configuración -> Opciones de desarrollador -> Dibujo -> Forzar dirección de diseño RTL

Habilitar esta opción obligaría al dispositivo a utilizar las configuraciones regionales de RTL y puede verificar fácilmente todas las partes de la aplicación para el soporte de RTL. Tenga en cuenta que no es necesario que agregue ninguna nueva configuración regional / idioma hasta este punto.

Codificación para localización: creación de cadenas y recursos predeterminados

El primer paso para la codificación de la localización es crear recursos predeterminados. Este paso es tan implícito que muchos desarrolladores ni siquiera lo piensan. Sin embargo, crear recursos predeterminados es importante porque si el dispositivo se ejecuta en una configuración regional no compatible, cargaría todos sus recursos desde las carpetas predeterminadas. Si incluso falta uno de los recursos de las carpetas predeterminadas, la aplicación simplemente se bloquearía.

El conjunto predeterminado de cadenas se debe colocar en la siguiente carpeta en la ubicación especificada:

```
res/values/strings.xml
```

Este archivo debe contener las cadenas en el idioma en el que se espera que hablen la mayoría de los usuarios de la aplicación.

Además, los recursos predeterminados para la aplicación se deben colocar en las siguientes carpetas y ubicaciones:

```
res/drawable/
res/layout/
```

Si su aplicación requiere carpetas como `anim` o `xml` , los recursos predeterminados deben agregarse a las siguientes carpetas y ubicaciones:

```
res/anim/
```

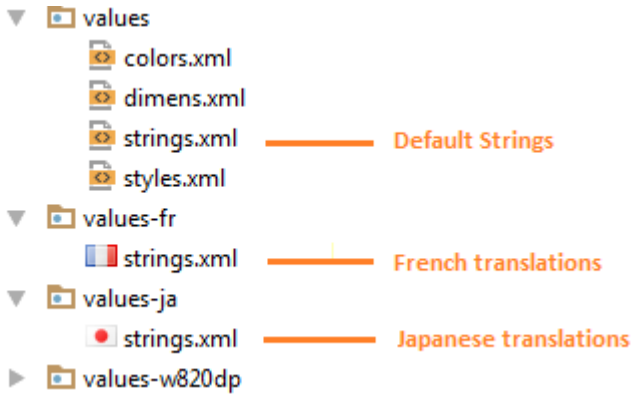
```
res/xml/
res/raw/
```

Codificación para localización: Proporcionar cadenas alternativas.

Para proporcionar traducciones en otros idiomas (locales), debemos crear un `strings.xml` en una carpeta separada según la siguiente convención:

```
res/values-<locale>/strings.xml
```

Un ejemplo para el mismo se da a continuación:



En este ejemplo, tenemos cadenas en inglés predeterminadas en el archivo `res/values/strings.xml` , las traducciones en francés se proporcionan en la carpeta `res/values-fr/strings.xml` y las traducciones en japonés en la carpeta `res/values-ja/strings.xml`

Otras traducciones para otras configuraciones regionales se pueden agregar de manera similar a la aplicación.

> Una lista completa de códigos de configuración regional se puede encontrar aquí: códigos ISO 639

Cuerdas no traducibles:

Su proyecto puede tener ciertas cadenas que no deben ser traducidas. Las cadenas que se utilizan como claves para SharedPreferences o las cadenas que se utilizan como símbolos, entran en esta categoría. Estas cadenas deben almacenarse solo en las `strings.xml` predeterminadas.xml y deben marcarse con un atributo `translatable="false"` . p.ej

```
<string name="pref_widget_display_label_hot">Hot News</string>
<string name="pref_widget_display_key" translatable="false">widget_display</string>
<string name="pref_widget_display_hot" translatable="false">0</string>
```

Este atributo es importante porque las traducciones suelen ser realizadas por profesionales que son bilingües. Esto permitiría a estas personas involucradas en las traducciones identificar cadenas que no se traducirán, ahorrando así tiempo y dinero.

Codificación para localización: Proporcionar diseños alternativos

La creación de diseños específicos del idioma a menudo es innecesaria si ha especificado la notación correcta de `start/end` , como se describe en el ejemplo anterior. Sin embargo, puede haber situaciones en las que los diseños predeterminados no funcionen correctamente para ciertos idiomas. A veces, los diseños de izquierda a derecha pueden no traducirse para los idiomas RTL. Es necesario proporcionar los diseños correctos en tales casos.

Para proporcionar una optimización completa para los diseños de RTL, podemos usar archivos de diseño completamente separados utilizando el `ldrtl` recursos `ldrtl` (`ldrtl` significa layout-direction-direction-right-left}). Por ejemplo, podemos guardar sus archivos de diseño predeterminados en `res/layout/` y nuestros diseños RTL optimizados en `res/layout-ldrtl/` .

El calificador `ldrtl` es ideal para recursos `ldrtl` , por lo que puede proporcionar gráficos orientados en la dirección correspondiente a la dirección de lectura.

Aquí hay una gran publicación que describe la precedencia de los diseños de `ldrtl` : Diseños específicos de idioma

Lea Internacionalización y localización (I18N y L10N) en línea: https://riptutorial.com/es/android/topic/8796/internacionalizacion-y-localizacion--i18n-y-l10n-

Capítulo 145: Jackson

Introducción

Jackson es una biblioteca multipropósito de Java para procesar JSON. Jackson pretende ser la mejor combinación posible de rápida, correcta, ligera y ergonómica para desarrolladores.

Jackson características

Modo multiprocesamiento, y muy buena colaboración.

No solo anotaciones, sino también anotaciones mixtas.

Totalmente compatible con los tipos genéricos.

Soporta tipos polimórficos.

Examples

Ejemplo completo de enlace de datos

Datos JSON

```
{
  "name" : { "first" : "Joe", "last" : "Sixpack" },
  "gender" : "MALE",
  "verified" : false,
  "userImage" : "keliuyue"
}
```

Se requieren dos líneas de Java para convertirlo en una instancia de Usuario:

```
ObjectMapper mapper = new ObjectMapper(); // can reuse, share globally
User user = mapper.readValue(new File("user.json"), User.class);
```

Clase de usuario

```
public class User {

    public enum Gender {MALE, FEMALE};

    public static class Name {
        private String _first, _last;

        public String getFirst() {
            return _first;
        }

        public String getLast() {
            return _last;
```

```
        }

        public void setFirst(String s) {
            _first = s;
        }

        public void setLast(String s) {
            _last = s;
        }
    }

    private Gender _gender;
    private Name _name;
    private boolean _isVerified;
    private byte[] _userImage;

    public Name getName() {
        return _name;
    }

    public boolean isVerified() {
        return _isVerified;
    }

    public Gender getGender() {
        return _gender;
    }

    public byte[] getUserImage() {
        return _userImage;
    }

    public void setName(Name n) {
        _name = n;
    }

    public void setVerified(boolean b) {
        _isVerified = b;
    }

    public void setGender(Gender g) {
        _gender = g;
    }

    public void setUserImage(byte[] b) {
        _userImage = b;
    }
}
```

Marshalling de nuevo a JSON es igualmente sencillo:

```
mapper.writeValue(new File("user-modified.json"), user);
```

Lea Jackson en línea: https://riptutorial.com/es/android/topic/10878/jackson

Capítulo 146: Java en Android

Introducción

Android es compatible con todas las funciones de lenguaje Java 7 y un subconjunto de funciones de lenguaje Java 8 que varían según la versión de la plataforma. Esta página describe las nuevas características de idioma que puede usar, cómo configurar correctamente su proyecto para usarlas y cualquier problema conocido que pueda encontrar.

Examples

Java 8 cuenta con subconjunto con Retrolambda

Retrolambda le permite ejecutar el código Java 8 con expresiones lambda, referencias de métodos y declaraciones try-with-resources en Java 7, 6 o 5. Lo hace transformando su código de bytes compilado de Java 8 para que pueda ejecutarse en un tiempo de ejecución Java más antiguo.

Características del lenguaje de Backported:

- Las expresiones Lambda se respaldan convirtiéndolas en clases internas anónimas. Esto incluye la optimización del uso de una instancia de singleton para expresiones lambda sin estado para evitar la asignación repetida de objetos. Las referencias de los métodos son básicamente solo azúcar de sintaxis para las expresiones lambda y se cargan en la parte posterior de la misma manera.

- Las declaraciones Try-with-resources se devuelven mediante la eliminación de llamadas a `Throwable.addSuppressed` si la versión del bytecode de destino está por debajo de Java 7. Si desea que se registren las excepciones suprimidas en lugar de que se las trague, cree una solicitud de función y la haremos configurable

- `Objects.requireNonNull` llamadas `Objects.requireNonNull` se reemplazan con las llamadas a `Object.getClass` si la versión del bytecode de destino está por debajo de Java 7. Las comprobaciones nulas sintéticas generadas por JDK 9 usan `Objects.requireNonNull` , mientras que las versiones anteriores de JDK usaban `Object.getClass` .

- Opcionalmente también:

    1. Los métodos predeterminados se respaldan al copiar los métodos predeterminados en una clase complementaria (nombre de la interfaz + "$") como métodos estáticos, reemplazando los métodos predeterminados en la interfaz con métodos abstractos y agregando las implementaciones de los métodos necesarios a todas las clases que implementan esa interfaz .

    2. Los métodos estáticos en las interfaces se respaldan moviendo los métodos estáticos a una clase complementaria (nombre de la interfaz + "$"), y cambiando todas las

llamadas a los métodos para llamar a la nueva ubicación del método.

Limitaciones conocidas:

- No respalda las API de Java 8
- Los métodos por defecto de backporting y los métodos estáticos en las interfaces requieren que todas las interfaces con backported y todas las clases que los implementan o que llamen a sus métodos estáticos se carguen con backport, con una ejecución de Retrolambda. En otras palabras, siempre debes hacer una compilación limpia. Además, los métodos predeterminados de backporting no funcionarán a través de módulos o límites de dependencia.
- Puede interrumpirse si una futura compilación JDK 8 deja de generar una nueva clase para cada llamada `invokedynamic` . Retrolambda funciona para capturar el código de byte que `java.lang.invoke.LambdaMetafactory` genera dinámicamente, por lo que las optimizaciones de ese mecanismo pueden interrumpir Retrolambda.

Retrolambda gradle plugin construirá automáticamente tu proyecto de Android con Retrolambda. La última versión se puede encontrar en la página de lanzamientos .

Uso:

1. Descarga e instala jdk8
2. Agrega lo siguiente a tu `build.gradle`

```
buildscript {
  repositories {
     mavenCentral()
  }

  dependencies {
     classpath 'me.tatarka:gradle-retrolambda:<latest version>'
  }
}

// Required because retrolambda is on maven central
repositories {
  mavenCentral()
}

apply plugin: 'com.android.application' //or apply plugin: 'java'
apply plugin: 'me.tatarka.retrolambda'

android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

Problemas conocidos:

- La pelusa falla en archivos java que tienen lambdas. La pelusa de Android no comprende la

sintaxis de java 8 y fallará en silencio o en voz alta. Ahora hay un tenedor experimental que soluciona el problema.

- El uso de los servicios de Google Play hace que Retrolambda falle. La versión 5.0.77 contiene un código de bytes que es incompatible con Retrolambda. Esto debería solucionarse en las versiones más recientes de los servicios de juego, si puede actualizar, esa debería ser la solución preferida. Para solucionar este problema, puede usar una versión anterior como 4.4.52 o agregar `-noverify` a jvm args.

```
retrolambda {
  jvmArgs '-noverify'
}
```

Lea Java en Android en línea: https://riptutorial.com/es/android/topic/9223/java-en-android

Capítulo 147: JCodec

Examples

Empezando

Puedes obtener JCodec automáticamente con maven. Para esto solo agregue el siguiente fragmento de código a su pom.xml.

```
<dependency>
    <groupId>org.jcodec</groupId>
    <artifactId>jcodec-javase</artifactId>
    <version>0.1.9</version>
</dependency>
```

Obtención de fotograma de la película

Obtención de un solo fotograma de una película (solo admite AVC, H.264 en MP4, ISO BMF, contenedor Quicktime):

```
int frameNumber = 150;
BufferedImage frame = FrameGrab.getFrame(new File("filename.mp4"), frameNumber);
ImageIO.write(frame, "png", new File("frame_150.png"));
```

Obtención de una secuencia de fotogramas de una película (solo admite AVC, H.264 en MP4, ISO BMF, contenedor Quicktime):

```
double startSec = 51.632;
FileChannelWrapper ch = null;
try {
    ch = NIOUtils.readableFileChannel(new File("filename.mp4"));
    FrameGrab fg = new FrameGrab(ch);
    grab.seek(startSec);
    for (int i = 0; i < 100; i++) {
        ImageIO.write(grab.getFrame(), "png",
            new File(System.getProperty("user.home"), String.format("Desktop/frame_%08d.png",
i)));
    }
} finally {
    NIOUtils.closeQuietly(ch);
}
```

Lea JCodec en línea: https://riptutorial.com/es/android/topic/9948/jcodec

Capítulo 148: JSON en Android con org.json

Sintaxis

- **Objeto** : un objeto es un conjunto desordenado de pares nombre / valor. Un objeto comienza con {(corchete izquierdo) y termina con} (tirante derecho). Cada nombre va seguido de: (dos puntos) y los pares nombre / valor están separados por, (coma).
- **Array** : una matriz es una colección ordenada de valores. Una matriz comienza con [(corchete izquierdo) y termina con] (corchete derecho). Los valores están separados por, (coma).
- **Valor** : Un valor puede ser una cadena entre comillas dobles, o un número, o verdadero o falso o nulo, o un objeto o una matriz. Estas estructuras pueden ser anidadas.
- **Cadena** : una cadena es una secuencia de cero o más caracteres Unicode, envueltos en comillas dobles, utilizando escapes de barra invertida. Un carácter se representa como una sola cadena de caracteres. Una cadena es muy parecida a una cadena C o Java.
- **Número** : un número es muy parecido a un número C o Java, excepto que no se utilizan los formatos octal y hexadecimal.

Observaciones

Este tema trata sobre el uso del paquete `org.json` que se incluye en el SDK de Android.

Examples

Parse simple objeto JSON

Considera la siguiente cadena JSON:

```
{
  "title": "test",
  "content": "Hello World!!!",
  "year": 2016,
  "names" : [
        "Hannah",
        "David",
        "Steve"
   ]
}
```

Este objeto JSON se puede analizar utilizando el siguiente código:

```
try {
    // create a new instance from a string
```

```
    JSONObject jsonObject = new JSONObject(jsonAsString);
    String title = jsonObject.getString("title");
    String content = jsonObject.getString("content");
    int year = jsonObject.getInt("year");
    JSONArray names = jsonObject.getJSONArray("names"); //for an array of String objects
} catch (JSONException e) {
    Log.w(TAG,"Could not parse JSON. Error: " + e.getMessage());
}
```

Aquí hay otro ejemplo con un JSONArray anidado dentro de JSONObject:

```
{
    "books":[
      {
        "title":"Android JSON Parsing",
        "times_sold":186
      }
    ]
}
```

Esto se puede analizar con el siguiente código:

```
JSONObject root = new JSONObject(booksJson);
JSONArray booksArray = root.getJSONArray("books");
JSONObject firstBook = booksArray.getJSONObject(0);
String title = firstBook.getString("title");
int timesSold = firstBook.getInt("times_sold");
```

Creando un objeto JSON simple

Cree el `JSONObject` usando el constructor vacío y agregue campos usando el método `put()` , que está sobrecargado para que se pueda usar con diferentes tipos:

```
try {
    // Create a new instance of a JSONObject
    final JSONObject object = new JSONObject();

    // With put you can add a name/value pair to the JSONObject
    object.put("name", "test");
    object.put("content", "Hello World!!!1");
    object.put("year", 2016);
    object.put("value", 3.23);
    object.put("member", true);
    object.put("null_value", JSONObject.NULL);

    // Calling toString() on the JSONObject returns the JSON in string format.
    final String json = object.toString();

} catch (JSONException e) {
    Log.e(TAG, "Failed to create JSONObject", e);
}
```

La cadena `JSON` resultante se ve así:

```
{
```

```
   "name":"test",
   "content":"Hello World!!!1",
   "year":2016,
   "value":3.23,
   "member":true,
   "null_value":null
}
```

Añadir JSONArray a JSONObject

```
// Create a new instance of a JSONArray
JSONArray array = new JSONArray();

// With put() you can add a value to the array.
array.put("ASDF");
array.put("QWERTY");

// Create a new instance of a JSONObject
JSONObject obj = new JSONObject();

try {
    // Add the JSONArray to the JSONObject
    obj.put("the_array", array);
} catch (JSONException e) {
    e.printStackTrace();
}

String json = obj.toString();
```

La cadena JSON resultante se ve así:

```
{
   "the_array":[
      "ASDF",
      "QWERTY"
   ]
}
```

Crear una cadena JSON con valor nulo.

Si necesita producir una cadena JSON con un valor `null` como este:

```
{
   "name":null
}
```

Entonces tienes que usar la constante especial JSONObject.NULL .

Ejemplo de funcionamiento:

```
jsonObject.put("name", JSONObject.NULL);
```

Trabajando con una cadena nula al analizar json

```
{
    "some_string": null,
    "ather_string": "something"
}
```

Si vamos a utilizar de esta manera:

```
JSONObject json = new JSONObject(jsonStr);
String someString = json.optString("some_string");
```

Tendremos salida:

```
someString = "null";
```

Así que tenemos que proporcionar esta solución:

```
/**
 * According to http://stackoverflow.com/questions/18226288/json-jsonobject-optstring-returns-
string-null
 * we need to provide a workaround to opt string from json that can be null.
 * <strong></strong>
 */
public static String optNullableString(JSONObject jsonObject, String key) {
    return optNullableString(jsonObject, key, "");
}

/**
 * According to http://stackoverflow.com/questions/18226288/json-jsonobject-optstring-returns-
string-null
 * we need to provide a workaround to opt string from json that can be null.
 * <strong></strong>
 */
public static String optNullableString(JSONObject jsonObject, String key, String fallback) {
    if (jsonObject.isNull(key)) {
        return fallback;
    } else {
        return jsonObject.optString(key, fallback);
    }
}
```

Y luego llamar:

```
JSONObject json = new JSONObject(jsonStr);
String someString = optNullableString(json, "some_string");
String someString2 = optNullableString(json, "some_string", "");
```

Y tendremos salida como esperábamos:

```
someString = null; //not "null"
someString2 = "";
```

Uso de JsonReader para leer JSON desde una secuencia

`JsonReader`

lee un valor codificado JSON como un flujo de tokens.

```
public List<Message> readJsonStream(InputStream in) throws IOException {
  JsonReader reader = new JsonReader(new InputStreamReader(in, "UTF-8"));
  try {
    return readMessagesArray(reader);
  } finally {
    reader.close();
  }
}

public List<Message> readMessagesArray(JsonReader reader) throws IOException {
  List<Message> messages = new ArrayList<Message>();

  reader.beginArray();
  while (reader.hasNext()) {
    messages.add(readMessage(reader));
  }
  reader.endArray();
  return messages;
}

public Message readMessage(JsonReader reader) throws IOException {
  long id = -1;
  String text = null;
  User user = null;
  List<Double> geo = null;

  reader.beginObject();
  while (reader.hasNext()) {
    String name = reader.nextName();
    if (name.equals("id")) {
      id = reader.nextLong();
    } else if (name.equals("text")) {
      text = reader.nextString();
    } else if (name.equals("geo") && reader.peek() != JsonToken.NULL) {
      geo = readDoublesArray(reader);
    } else if (name.equals("user")) {
      user = readUser(reader);
    } else {
      reader.skipValue();
    }
  }
  reader.endObject();
  return new Message(id, text, user, geo);
}

public List<Double> readDoublesArray(JsonReader reader) throws IOException {
  List<Double> doubles = new ArrayList<Double>();

  reader.beginArray();
  while (reader.hasNext()) {
    doubles.add(reader.nextDouble());
  }
  reader.endArray();
  return doubles;
}

public User readUser(JsonReader reader) throws IOException {
  String username = null;
  int followersCount = -1;
```

```
    reader.beginObject();
    while (reader.hasNext()) {
      String name = reader.nextName();
      if (name.equals("name")) {
        username = reader.nextString();
      } else if (name.equals("followers_count")) {
        followersCount = reader.nextInt();
      } else {
        reader.skipValue();
      }
    }
    reader.endObject();
    return new User(username, followersCount);
  }
```

Crear objeto JSON anidado

Para producir un objeto JSON anidado, necesita simplemente agregar un objeto JSON a otro:

```
JSONObject mainObject = new JSONObject();            // Host object
JSONObject requestObject = new JSONObject();         // Included object

try {
    requestObject.put("lastname", lastname);
    requestObject.put("phone", phone);
    requestObject.put("latitude", lat);
    requestObject.put("longitude", lon);
    requestObject.put("theme", theme);
    requestObject.put("text", message);

    mainObject.put("claim", requestObject);
} catch (JSONException e) {
    return "JSON Error";
}
```

Ahora `mainObject` contiene una clave llamada `claim` con todo el `requestObject` como un valor.

Manejo de clave dinámica para respuesta JSON

Este es un ejemplo de cómo manejar la clave dinámica para la respuesta. Aquí `A` y `B` son claves dinámicas puede ser cualquier cosa.

Respuesta

```
{
  "response": [
    {
      "A": [
        {
          "name": "Tango"
        },
        {
          "name": "Ping"
        }
      ],
```

```
      "B": [
        {
          "name": "Jon"
        },
        {
          "name": "Mark"
        }
      ]
    }
  ]
}
```

Código Java

```
// ResponseData is raw string of response
JSONObject responseDataObj = new JSONObject(responseData);
JSONArray responseArray = responseDataObj.getJSONArray("response");
for (int i = 0; i < responseArray.length(); i++) {
    // Nodes ArrayList<ArrayList<String>> declared globally
    nodes = new ArrayList<ArrayList<String>>();
    JSONObject obj = responseArray.getJSONObject(i);
    Iterator keys = obj.keys();
    while(keys.hasNext()) {
       // Loop to get the dynamic key
       String currentDynamicKey = (String)keys.next();
       // Get the value of the dynamic key
       JSONArray currentDynamicValue = obj.getJSONArray(currentDynamicKey);
       int jsonArraySize = currentDynamicValue.length();
       if(jsonArraySize > 0) {
           for (int ii = 0; ii < jsonArraySize; ii++) {
                // NameList ArrayList<String> declared globally
                nameList = new ArrayList<String>();
               if(ii == 0) {
                JSONObject nameObj = currentDynamicValue.getJSONObject(ii);
                String name = nameObj.getString("name");
                System.out.print("Name = " + name);
                // Store name in an array list
                nameList.add(name);
              }
           }
       }
     nodes.add(nameList);
    }
}
```

Compruebe la existencia de campos en JSON

A veces es útil verificar si un campo está presente o ausente en su JSON para evitar alguna `JSONException` en su código.

Para lograrlo, use el `JSONObject#has(String)` o el método, como en el siguiente ejemplo:

Muestra JSON

```
{
   "name":"James"
}
```

Código Java

```
String jsonStr = " { \"name\":\"James\" }";
JSONObject json = new JSONObject(jsonStr);
// Check if the field "name" is present
String name, surname;

// This will be true, since the field "name" is present on our JSON.
if (json.has("name")) {
    name = json.getString("name");
}
else {
    name = "John";
}
// This will be false, since our JSON doesn't have the field "surname".
if (json.has("surname")) {
    surname = json.getString("surname");
}
else {
    surname = "Doe";
}

// Here name == "James" and surname == "Doe".
```

Actualizando los elementos en el JSON.

muestra json para actualizar

```
 {
 "student":{"name":"Rahul", "lastname":"sharma"},
 "marks":{"maths":"88"}
 }
```

Para actualizar el valor de los elementos en el json debemos asignar el valor y actualizar.

```
try {
    // Create a new instance of a JSONObject
    final JSONObject object = new JSONObject(jsonString);

    JSONObject studentJSON = object.getJSONObject("student");
    studentJSON.put("name","Kumar");

     object.remove("student");

     object.put("student",studentJSON);

    // Calling toString() on the JSONObject returns the JSON in string format.
    final String json = object.toString();

} catch (JSONException e) {
    Log.e(TAG, "Failed to create JSONObject", e);
}
```

valor actualizado

```
 {
```

```
"student":{"name":"Kumar", "lastname":"sharma"},
"marks":{"maths":"88"}
}
```

Lea JSON en Android con org.json en línea: https://riptutorial.com/es/android/topic/106/json-en-android-con-org-json

Capítulo 149: Leakcanary

Introducción

Leak Canary es una biblioteca de Android y Java utilizada para detectar fugas en la aplicación.

Observaciones

Puedes ver el ejemplo en el enlace de abajo.

https://github.com/square/leakcanary

Examples

Implementando una aplicación de Leak Canary en Android

En su *build.gradle* debe agregar las siguientes dependencias:

```
debugCompile 'com.squareup.leakcanary:leakcanary-android:1.5.1'
releaseCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
testCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
```

En su clase de `Application` , debe agregar el siguiente código dentro de su `onCreate()` :

```
LeakCanary.install(this);
```

Eso es todo lo que necesita hacer para *LeakCanary* , mostrará notificaciones automáticamente cuando hay una fuga en su compilación.

Lea Leakcanary en línea: https://riptutorial.com/es/android/topic/10041/leakcanary

Capítulo 150: Lectura de códigos de barras y códigos QR

Observaciones

QRCodeReaderView

Zxing

Examples

Usando QRCodeReaderView (basado en Zxing)

QRCodeReaderView implementa una vista de Android que muestra la cámara y notifica cuando hay un código QR dentro de la vista previa.

Utiliza la biblioteca de procesamiento de imágenes de código de barras 1D / 2D multiformato de código abierto zxing .

Agregando la biblioteca a tu proyecto

Agregue la dependencia QRCodeReaderView a su build.gradle

```
dependencies{
    compile 'com.dlazaro66.qrcodereaderview:qrcodereaderview:2.0.0'
}
```

Primer uso

- Añade a tu diseño un `QRCodeReaderView`

```
<com.dlazaro66.qrcodereaderview.QRCodeReaderView
       android:id="@+id/qrdecoderview"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />
```

- Cree una actividad que implemente en `onQRCodeReadListener` y utilícela como un oyente de `QrCodeReaderView` .
- Asegúrate de tener permisos de cámara para usar la biblioteca. (https://developer.android.com/training/permissions/requesting.html)

Luego, en tu Actividad, puedes usarlo de la siguiente manera:

```
    public class DecoderActivity extends Activity implements OnQRCodeReadListener {

    private TextView resultTextView;
    private QRCodeReaderView qrCodeReaderView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_decoder);

        qrCodeReaderView = (QRCodeReaderView) findViewById(R.id.qrdecoderview);
        qrCodeReaderView.setOnQRCodeReadListener(this);

        // Use this function to enable/disable decoding
        qrCodeReaderView.setQRDecodingEnabled(true);

        // Use this function to change the autofocus interval (default is 5 secs)
        qrCodeReaderView.setAutofocusInterval(2000L);

        // Use this function to enable/disable Torch
        qrCodeReaderView.setTorchEnabled(true);

        // Use this function to set front camera preview
        qrCodeReaderView.setFrontCamera();

        // Use this function to set back camera preview
        qrCodeReaderView.setBackCamera();
    }

    // Called when a QR is decoded
    // "text" : the text encoded in QR
    // "points" : points where QR control points are placed in View
    @Override
    public void onQRCodeRead(String text, PointF[] points) {
        resultTextView.setText(text);
    }

    @Override
    protected void onResume() {
        super.onResume();
        qrCodeReaderView.startCamera();
    }

    @Override
    protected void onPause() {
        super.onPause();
        qrCodeReaderView.stopCamera();
    }
}
```

Lea Lectura de códigos de barras y códigos QR en línea:
https://riptutorial.com/es/android/topic/6067/lectura-de-codigos-de-barras-y-codigos-qr

Capítulo 151: Library Dagger 2: Inyección de dependencia en aplicaciones

Introducción

Dagger 2, como se explica en GitHub , es un enfoque de evolución en tiempo de compilación para la inyección de dependencia. Tomando el enfoque iniciado en Dagger 1.x hasta su conclusión final, Dagger 2.x elimina toda reflexión y mejora la claridad del código al eliminar el `ObjectGraph` / `Injector` tradicional en favor de las interfaces `@Component` especificadas `@Component` usuario.

Observaciones

1. Configuración de la biblioteca en la aplicación (para proyectos maven, gradle, java)
2. Ventajas del uso de Dragger
3. Enlaces importantes (para documentación y demos)
4. Cómo integrar y usar los componentes de Dragger

Dagger 2 API:

Daga 2 expone una serie de anotaciones especiales:

@Module para las clases cuyos métodos proporcionan dependencias

@Proporciona los métodos dentro de las clases de @Module

@Inyectar para solicitar una dependencia (un constructor, un campo o un método)

@Component es una interfaz puente entre módulos e inyección.

Links importantes:

GitHub: https://github.com/google/dagger

UserGuide (Google): https://google.github.io/dagger/users-guide.html

Videos: https://google.github.io/dagger/resources.html

Tutorial de Vogella: http://www.vogella.com/tutorials/Dagger/article.html

Tutorial de Codepath: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Examples

Cree la clase @Module y la anotación @Singleton para el objeto

```
import javax.inject.Singleton;
import dagger.Module;
import dagger.Provides;

@Module
public class VehicleModule {

    @Provides @Singleton
    Motor provideMotor(){
        return new Motor();
    }

    @Provides @Singleton
    Vehicle provideVehicle(){
        return new Vehicle(new Motor());
    }
}
```

Cada proveedor (o método) debe tener la anotación `@Provides` y la clase debe tener la anotación `@Module` . La anotación `@Singleton` indica que solo habrá una instancia del objeto.

Solicitud de dependencias en objetos dependientes

Ahora que tiene los proveedores para sus diferentes modelos, debe solicitarlos. Al igual que el `Vehicle` necesita `Motor` , debe agregar la anotación `@Inject` en el constructor del `Vehicle` siguiente manera:

```
@Inject
public Vehicle(Motor motor){
    this.motor = motor;
}
```

Puede usar la anotación `@Inject` para solicitar dependencias en el constructor, los campos o los métodos. En este ejemplo, estoy manteniendo la inyección en el constructor.

Conectando @Modules con @Inject

La conexión entre el proveedor de dependencias, `@Module` y las clases que las solicitan a través de `@Inject` se realiza mediante `@Component` , que es una interfaz:

```
import javax.inject.Singleton;
import dagger.Component;

@Singleton
@Component(modules = {VehicleModule.class})
public interface VehicleComponent {
    Vehicle provideVehicle();
}
```

Para la anotación `@Component` , debe especificar qué módulos se van a utilizar. En este ejemplo, se utiliza `VehicleModule` , que se define en este ejemplo . Si necesita usar más módulos, simplemente agréguelos usando una coma como separador.

Usando la interfaz @Component para obtener objetos

Ahora que tiene todas las conexiones listas, debe obtener una instancia de esta interfaz e invocar sus métodos para obtener el objeto que necesita:

```
VehicleComponent component = Dagger_VehicleComponent.builder().vehicleModule(new
VehicleModule()).build();
vehicle = component.provideVehicle();
Toast.makeText(this, String.valueOf(vehicle.getSpeed()), Toast.LENGTH_SHORT).show();
```

Cuando intenta crear un nuevo objeto de la interfaz con la anotación `@Component` , debe hacerlo utilizando el prefijo `Dagger_<NameOfTheComponentInterface>` , en este caso `Dagger_VehicleComponent` , y luego usar el método del generador para llamar a cada módulo interno.

Lea Library Dagger 2: Inyección de dependencia en aplicaciones en línea: https://riptutorial.com/es/android/topic/9079/library-dagger-2--inyeccion-de-dependencia-en-aplicaciones

Capítulo 152: Lienzo de dibujo utilizando SurfaceView

Observaciones

Es importante comprender el concepto básico de la vista de superficie antes de usar:

- Es básicamente un agujero en la ventana actual.
- La interfaz de usuario nativa se puede colocar encima de ella
- El dibujo se realiza mediante un subproceso dedicado, sin interfaz de usuario
- El dibujo no es acelerado por hardware
- Utiliza dos buffers: uno se muestra actualmente, uno se usa para dibujar.
- `unlockCanvasAndPost()` intercambia los buffers.

Los puntos muertos pueden ocurrir fácilmente si los `lockCanvas()` y `unlockCanvasAndPost()` no se llaman en el orden correcto.

Examples

SurfaceView con hilo de dibujo

Este ejemplo describe cómo crear un SurfaceView con un hilo de dibujo dedicado. Esta implementación también maneja casos extremos, como problemas específicos de fabricación, así como el inicio / detención del hilo para ahorrar tiempo de CPU.

```
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
/**
* Defines a custom SurfaceView class which handles the drawing thread
**/
public class BaseSurface extends SurfaceView implements SurfaceHolder.Callback,
View.OnTouchListener, Runnable
{

    /**
     * Holds the surface frame
     */
    private SurfaceHolder holder;

    /**
     * Draw thread
     */
```

```
    private Thread drawThread;

    /**
     * True when the surface is ready to draw
     */
    private boolean surfaceReady = false;


    /**
     * Drawing thread flag
     */

    private boolean drawingActive = false;

    /**
     * Paint for drawing the sample rectangle
     */
    private Paint samplePaint = new Paint();

    /**
     * Time per frame for 60 FPS
     */
    private static final int MAX_FRAME_TIME = (int) (1000.0 / 60.0);

    private static final String LOGTAG = "surface";

    public BaseSurface(Context context, AttributeSet attrs)
    {
        super(context, attrs);
        SurfaceHolder holder = getHolder();
        holder.addCallback(this);
        setOnTouchListener(this);

        // red
        samplePaint.setColor(0xffff0000);
        // smooth edges
        samplePaint.setAntiAlias(true);
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
    {
        if (width == 0 || height == 0)
        {
            return;
        }

        // resize your UI
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder)
    {
        this.holder = holder;

        if (drawThread != null)
        {
            Log.d(LOGTAG, "draw thread still active..");
            drawingActive = false;
            try
            {
```

```
            drawThread.join();
        } catch (InterruptedException e)
        { // do nothing
        }
    }

    surfaceReady = true;
    startDrawThread();
    Log.d(LOGTAG, "Created");
}

@Override
public void surfaceDestroyed(SurfaceHolder holder)
{
    // Surface is not used anymore - stop the drawing thread
    stopDrawThread();
    // and release the surface
    holder.getSurface().release();

    this.holder = null;
    surfaceReady = false;
    Log.d(LOGTAG, "Destroyed");
}

@Override
public boolean onTouch(View v, MotionEvent event)
{
    // Handle touch events
    return true;
}

/**
 * Stops the drawing thread
 */
public void stopDrawThread()
{
    if (drawThread == null)
    {
        Log.d(LOGTAG, "DrawThread is null");
        return;
    }
    drawingActive = false;
    while (true)
    {
        try
        {
            Log.d(LOGTAG, "Request last frame");
            drawThread.join(5000);
            break;
        } catch (Exception e)
        {
            Log.e(LOGTAG, "Could not join with draw thread");
        }
    }
    drawThread = null;
}

/**
 * Creates a new draw thread and starts it.
 */
public void startDrawThread()
```

```
    {
        if (surfaceReady && drawThread == null)
        {
            drawThread = new Thread(this, "Draw thread");
            drawingActive = true;
            drawThread.start();
        }
    }

    @Override
    public void run()
    {
        Log.d(LOGTAG, "Draw thread started");
        long frameStartTime;
        long frameTime;

        /*
         * In order to work reliable on Nexus 7, we place ~500ms delay at the start of drawing
thread
         * (AOSP - Issue 58385)
         */
        if (android.os.Build.BRAND.equalsIgnoreCase("google") &&
android.os.Build.MANUFACTURER.equalsIgnoreCase("asus") &&
android.os.Build.MODEL.equalsIgnoreCase("Nexus 7"))
        {
            Log.w(LOGTAG, "Sleep 500ms (Device: Asus Nexus 7)");
            try
            {
                Thread.sleep(500);
            } catch (InterruptedException ignored)
            {
            }
        }
        try
        {
            while (drawingActive)
            {
                if (holder == null)
                {
                    return;
                }

                frameStartTime = System.nanoTime();
                Canvas canvas = holder.lockCanvas();
                if (canvas != null)
                {
                    // clear the screen using black
                    canvas.drawARGB(255, 0, 0, 0);

                    try
                    {
                        // Your drawing here
                        canvas.drawRect(0, 0, getWidth() / 2, getHeight() / 2, samplePaint);
                    } finally
                    {

                        holder.unlockCanvasAndPost(canvas);
                    }
                }

                // calculate the time required to draw the frame in ms
```

```
                frameTime = (System.nanoTime() - frameStartTime) / 1000000;

                if (frameTime < MAX_FRAME_TIME) // faster than the max fps - limit the FPS
                {
                    try
                    {
                        Thread.sleep(MAX_FRAME_TIME - frameTime);
                    } catch (InterruptedException e)
                    {
                        // ignore
                    }
                }
            }
        } catch (Exception e)
        {
            Log.w(LOGTAG, "Exception while locking/unlocking");
        }
        Log.d(LOGTAG, "Draw thread finished");
    }
}
```

Este diseño solo contiene el SurfaceView personalizado y lo maximiza al tamaño de la pantalla.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="sample.devcore.org.surfaceviewsample.MainActivity">

    <sample.devcore.org.surfaceviewsample.BaseSurface
        android:id="@+id/baseSurface"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</LinearLayout>
```

La actividad que utiliza SurfaceView es responsable de iniciar y detener el hilo de dibujo. Este enfoque ahorra batería cuando el dibujo se detiene tan pronto como la actividad se pone en segundo plano.

```
import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity
{

    /**
     * Surface object
     */
    private BaseSurface surface;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        surface = (BaseSurface) findViewById(R.id.baseSurface);
```

```
    }

    @Override
    protected void onResume()
    {
        super.onResume();
        // start the drawing
        surface.startDrawThread();
    }

    @Override
    protected void onPause()
    {
        // stop the drawing to save cpu time
        surface.stopDrawThread();
        super.onPause();
    }
}
```

Lea Lienzo de dibujo utilizando SurfaceView en línea:
https://riptutorial.com/es/android/topic/3754/lienzo-de-dibujo-utilizando-surfaceview

Capítulo 153: Localización con recursos en Android.

Examples

Moneda

```
Currency currency = Currency.getInstance("USD");
NumberFormat format = NumberFormat.getCurrencyInstance();
format.setCurrency(currency);
format.format(10.00);
```

Añadiendo traducción a tu aplicación de Android

Tienes que crear un archivo `strings.xml` diferente para cada nuevo idioma.

1. Haga clic derecho en la carpeta *res*
2. Elegir *Nuevo* → *Archivo de recursos de valores*
3. Seleccione una configuración regional de los calificadores disponibles
4. Haga clic en el botón *Siguiente* (>>)
5. Selecciona un idioma
6. Nombra el archivo *strings.xml*

strings.xml

```
<resources>
    <string name="app_name">Testing Application</string>
    <string name="hello">Hello World</string>
</resources>
```

strings.xml (hi)

```
<resources>
    <string name="app_name">परीक्षण आवेदन</string>
    <string name="hello">नमस्ते दुनिया</string>
</resources>
```

Configurando el idioma programáticamente:

```
public void setLocale(String locale) // Pass "en","hi", etc.
{
    myLocale = new Locale(locale);
    // Saving selected locale to session - SharedPreferences.
    saveLocale(locale);
    // Changing locale.
    Locale.setDefault(myLocale);
    android.content.res.Configuration config = new android.content.res.Configuration();
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
```

```
        config.setLocale(myLocale);
    } else {
        config.locale = myLocale;
    }
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1) {
        getBaseContext().createConfigurationContext(config);
    } else {
        getBaseContext().getResources().updateConfiguration(config,
getBaseContext().getResources().getDisplayMetrics());
    }
}
```

La función anterior cambiará los campos de texto a los que se hace referencia desde *strings.xml* . Por ejemplo, suponga que tiene las siguientes dos vistas de texto:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/app_name"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello"/>
```

Luego, después de cambiar la configuración regional, las cadenas de idioma que tienen los `app_name` y `hello` se cambiarán en consecuencia.

Tipo de directorios de recursos en la carpeta "res"

Al localizar diferentes tipos de recursos se requieren, cada uno de los cuales tiene su propio hogar en la estructura del proyecto de Android. A continuación se muestran los diferentes directorios que podemos colocar en el directorio `\res` . Los tipos de recursos colocados en cada uno de estos directorios se explican en la siguiente tabla:

Directorio	Tipo de recurso
animador/	Archivos XML que definen animaciones de propiedades.
anim /	Archivos XML que definen animaciones de interpolación. (Las animaciones de propiedades también se pueden guardar en este directorio, pero se prefiere el animador / directorio para que las animaciones de propiedades distingan entre los dos tipos).
color/	Archivos XML que definen una lista de colores de estado. Ver recurso de lista de estados de color
dibujable /	"Archivos de mapa de bits (.png, .9.png, .jpg, .gif) o archivos XML que se compilan en los siguientes subtipos de recursos dibujables:: `Bitmap files - Nine-Patches (re-sizable bitmaps) - State lists - Shapes - Animation drawables - Other drawables - "`
mipmap /	Archivos dibujables para diferentes densidades de iconos de lanzadores. Para

Directorio	Tipo de recurso
	obtener más información sobre la administración de los iconos del iniciador con mipmap / carpetas, consulte Administración de proyectos.
diseño/	Archivos XML que definen un diseño de interfaz de usuario. Ver recurso de diseño.
menú/	Archivos XML que definen menús de aplicaciones, como un menú de opciones, un menú contextual o un submenú. Ver recurso de menú.
crudo/	Archivos arbitrarios para guardar en su forma cruda. Para abrir estos recursos con un InputStream sin formato, llame a Resources.openRawResource () con el ID de recurso, que es R.raw.filename.
	Sin embargo, si necesita acceder a los nombres de archivos originales y a la jerarquía de archivos, puede considerar guardar algunos recursos en el directorio de activos (en lugar deres / raw /). Los archivos en los activos / no tienen un ID de recurso, por lo que puede leerlos solo con AssetManager.
valores/	Archivos XML que contienen valores simples, como cadenas, enteros y colores, así como estilos y temas
xml /	Archivos XML arbitrarios que se pueden leer en tiempo de ejecución llamando a Resources.getXML (). Aquí se deben guardar varios archivos de configuración XML, como una configuración de búsqueda.

Tipos de configuración y nombres de calificadores para cada carpeta en el directorio "res"

Cada directorio de recursos en la carpeta `res` (que se enumera en el ejemplo anterior) puede tener diferentes variaciones de los recursos contenidos en un directorio de nombre similar con un sufijo con diferentes `qualifier-values` de `qualifier-values` para cada `configuration-type` .

Ejemplo de variaciones del directorio `` con diferentes valores de calificador con sufijo que se ven a menudo en nuestros proyectos de Android:

- dibujable /
- drawable-en /
- dibujable-fr-rCA /
- drawable-en-port /
- drawable-en-notouch-12key /
- drawable-port-ldpi /
- drawable-port-notouch-12key /

Lista exhaustiva de todos los diferentes tipos de configuración y sus valores calificadores para los recursos

de Android:

Configuración	Valores calificadores
MCC y MNC	Ejemplos:
	mcc310
	mcc310-mnc004
	mcc208-mnc00
	etc.
Idioma y región	Ejemplos:
	en
	fr
	en-rUS
	fr-rFR
	fr-rCA
Dirección de diseño	ldrtl
	ldltr
Ancho más pequeño	swdp
	Ejemplos:
	sw320dp
	sw600dp
	sw720dp
Ancho disponible	wdp
	w720dp
	w1024dp
Altura disponible	hdp
	h720dp

Configuración	Valores calificadores
	h1024dp
Tamaño de pantalla	pequeña
	normal
	grande
	xlarge
Aspecto de la pantalla	largo
	No largo
Pantalla redonda	redondo
	cerca
Orientación de la pantalla	Puerto
	tierra
Modo de interfaz de usuario	coche
	escritorio
	televisión
	aparato
Modo nocturno	noche
	no de noche
Densidad de píxeles de la pantalla (dpi)	ldpi
	mdpi
	hdpi
	xhdpi
	xxhdpi
	xxxhdpi
	nodpi
	tvdpi

Configuración	Valores calificadores
	anydpi
Tipo de pantalla táctil	no tocar
	dedo
Disponibilidad de teclado	llave sujeta
	llave oculta
	keyssoft
Método de entrada de texto primario	nokeys
	QWERTY
	12 teclas
Disponibilidad de la tecla de navegación	navexposed
	navhidden
Método de navegación no táctil principal	nonav
	dpad
	bola de seguimiento
	rueda
Versión de plataforma (nivel API)	Ejemplos:
	v3
	v4
	v7

Cambiar la configuración regional de la aplicación de Android programáticamente

En los ejemplos anteriores se entiende cómo localizar los recursos de la aplicación. El siguiente ejemplo explica cómo cambiar la configuración regional de la aplicación dentro de la aplicación, no desde el dispositivo. Para cambiar solo la configuración regional de la aplicación, puede utilizar la siguiente configuración de configuración regional.

```
import android.app.Application;
import android.content.Context;
```

```
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.content.res.Resources;
import android.os.Build;
import android.preference.PreferenceManager;
import android.view.ContextThemeWrapper;

import java.util.Locale;

/**
 * Created by Umesh on 10/10/16.
 */
public class LocaleUtils {

    private static Locale mLocale;

    public static void setLocale(Locale locale){
        mLocale = locale;
        if(mLocale != null){
            Locale.setDefault(mLocale);
        }
    }

    public static void updateConfiguration(ContextThemeWrapper wrapper){
        if(mLocale != null && Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1){
            Configuration configuration = new Configuration();
            configuration.setLocale(mLocale);
            wrapper.applyOverrideConfiguration(configuration);
        }
    }

    public static void updateConfiguration(Application application, Configuration
configuration){
        if(mLocale != null && Build.VERSION.SDK_INT < Build.VERSION_CODES.JELLY_BEAN_MR1){
            Configuration config = new Configuration(configuration);
            config.locale = mLocale;
            Resources res = application.getBaseContext().getResources();
            res.updateConfiguration(configuration, res.getDisplayMetrics());
        }
    }

    public static void updateConfiguration(Context context, String language, String country){
        Locale locale = new Locale(language,country);
        setLocale(locale);
        if(mLocale != null){
            Resources res = context.getResources();
            Configuration configuration = res.getConfiguration();
            configuration.locale = mLocale;
            res.updateConfiguration(configuration,res.getDisplayMetrics());
        }
    }




    public static String getPrefLangCode(Context context) {
        return
PreferenceManager.getDefaultSharedPreferences(context).getString("lang_code","en");
    }

    public static void setPrefLangCode(Context context, String mPrefLangCode) {
```

```
        SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(context).edit();
        editor.putString("lang_code",mPrefLangCode);
        editor.commit();
    }

    public static  String getPrefCountryCode(Context context) {
        return
PreferenceManager.getDefaultSharedPreferences(context).getString("country_code","US");
    }

    public static void setPrefCountryCode(Context context,String mPrefCountryCode) {

        SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(context).edit();
        editor.putString("country_code",mPrefCountryCode);
        editor.commit();
    }
}
```

Inicialice la configuración regional que el usuario prefiera, desde la clase de aplicación.

```
public class LocaleApp extends Application{

    @Override
    public void onCreate() {
        super.onCreate();

        LocaleUtils.setLocale(new Locale(LocaleUtils.getPrefLangCode(this),
LocaleUtils.getPrefCountryCode(this)));
        LocaleUtils.updateConfiguration(this, getResources().getConfiguration());
    }
}
```

También necesita crear una actividad base y extender esta actividad a todas las demás actividades para que pueda cambiar la configuración regional de la aplicación solo en un lugar de la siguiente manera:

```
public abstract class LocalizationActivity extends AppCompatActivity {

    public LocalizationActivity() {
        LocaleUtils.updateConfiguration(this);
    }

    // We only override onCreate
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

}
```

Nota: Siempre inicialice la configuración regional en el constructor.

Ahora puedes usar LocalizationActivity como sigue.

```
public class MainActivity extends LocalizationActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

    }
}
```

> Nota: cuando cambia la configuración regional de la aplicación mediante programación, debe reiniciar su actividad para que tenga efecto el cambio de configuración regional Para que funcione correctamente para esta solución, use la configuración regional de las preferencias compartidas en el inicio de la aplicación `android:name=".LocaleApp"` en Usted Manifest.xml.

A veces, el mensaje de Lint checker para crear la versión de lanzamiento. Para resolver tal problema, siga las siguientes opciones.

Primero:

Si desea deshabilitar la traducción solo para algunas cadenas, agregue el siguiente atributo a la cadena predeterminada.xml

```
<string name="developer" translatable="false">Developer Name</string>
```

Segundo:

Ignore todas las traducciones faltantes del archivo de recursos. Añada el siguiente atributo. Es el atributo de ignorar del espacio de nombres de las herramientas en su archivo de cadenas, de la siguiente manera:

```
<?xml version="1.0" encoding="utf-8"?>
<resources
  xmlns:tools="http://schemas.android.com/tools"
  tools:ignore="MissingTranslation" >
http://stackoverflow.com/documentation/android/3345/localization-with-resources-in-android#
  <!-- your strings here; no need now for the translatable attribute -->

</resources>
```

Tercero:

Otra forma de deshabilitar la cadena no traducible

http://tools.android.com/recent/non-translatablestrings

Si tiene muchos recursos que no se deben traducir, puede colocarlos en un archivo llamado donottranslate.xml y lint los considerará todos recursos no traducibles.

Cuarto:

También puede agregar la configuración regional en el archivo de recursos

```
<resources
xmlns:tools="http://schemas.android.com/tools"
    tools:locale="en" tools:ignore="MissingTranslation">
```

También puede deshabilitar la comprobación de traducción faltante para la pelusa de app / build.gradle

```
lintOptions {

        disable 'MissingTranslation'
    }
```

Lea Localización con recursos en Android. en línea: https://riptutorial.com/es/android/topic/3345/localizacion-con-recursos-en-android-

Capítulo 154: Looper

Introducción

Un `Looper` es una clase de Android que se utiliza para ejecutar un bucle de mensajes para un hilo, que generalmente no tiene uno asociado con ellos.

El `Looper` más común en Android es el bucle principal, también conocido comúnmente como el hilo principal. Esta instancia es única para una aplicación y se puede acceder de forma estática con `Looper.getMainLooper()` .

Si un `Looper` está asociado con el subproceso actual, se puede recuperar con `Looper.myLooper()` .

Examples

Crear un LooperThread simple

Un ejemplo típico de la implementación de un subproceso `Looper` proporcionado por la documentación oficial utiliza `Looper.prepare()` y `Looper.loop()` y asocia un `Handler` con el bucle entre estas llamadas.

```
class LooperThread extends Thread {
    public Handler mHandler;

    public void run() {
        Looper.prepare();

        mHandler = new Handler() {
            public void handleMessage(Message msg) {
                // process incoming messages here
            }
        };

        Looper.loop();
    }
}
```

Ejecutar un bucle con un HandlerThread

Se puede utilizar un `HandlerThread` para iniciar un hilo con un `Looper` . Este looper se puede usar para crear un `Handler` para las comunicaciones con él.

```
HandlerThread thread = new HandlerThread("thread-name");
thread.start();
Handler handler = new Handler(thread.getLooper());
```

Lea Looper en línea: https://riptutorial.com/es/android/topic/10593/looper

Capítulo 155: LruCache

Observaciones

Debe usar Lru Cache en aplicaciones donde las cargas repetitivas de recursos afecten a un comportamiento suave de la aplicación. Por ejemplo, una galería de fotos con miniaturas grandes (128x128).

Siempre tenga cuidado con el tamaño de la memoria caché Lru, ya que su configuración demasiado alta podría afectar a la aplicación.

Después de que el Lru Cache ya no sea útil, evite mantener referencias al mismo para permitir que el recolector de basura lo limpie de la memoria.

Para obtener el mejor rendimiento, recuerde cargar recursos como mapas de bits usando las mejores prácticas, como seleccionar un inSampleSize adecuado antes de agregarlo a la memoria caché Lru.

Examples

Inicializando el caché

El Lru Cache almacenará todos los recursos agregados (valores) para un acceso rápido hasta que alcance un límite de memoria, en cuyo caso eliminará el recurso menos usado (valor) para almacenar el nuevo.

Para inicializar el caché Lru debe proporcionar un valor de memoria máximo. Este valor depende de los requisitos de su aplicación y de la importancia del recurso para mantener un uso sin problemas de la aplicación. Un valor recomendado para una galería de imágenes, por ejemplo, sería 1/8 de su memoria máxima disponible.

También tenga en cuenta que el Lru Cache funciona sobre una base de valor-clave. En el siguiente ejemplo, la clave es una `String` y el valor es un `Bitmap` :

```
int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);
int cacheSize = maxMemory / 8;

LruCache<String, Bitmap> = memoryCache = new LruCache<String, Bitmap>(cacheSize) {
    protected int sizeOf(String key, Bitmap bitmap) {
        return bitmap.getByteCount();
    }
};
```

Agregar un mapa de bits (recurso) a la caché

Para agregar un recurso al caché, debe proporcionar una clave y el recurso. Primero asegúrese de que el valor no esté en el caché ya

```
public void addResourceToMemoryCache(String key, Bitmap resource) {
    if (memoryCache.get(key) == null)
        memoryCache.put(key, resource);
}
```

Obtención de un mapa de bits (respuesta) de la caché

Para obtener un recurso del caché, simplemente pase la clave de su recurso (Cadena en este ejemplo)

```
public Bitmap getResourceFromMemoryCache(String key) {
    memoryCache.get(key);
}
```

Lea LruCache en línea: https://riptutorial.com/es/android/topic/7709/lrucache

Capítulo 156: Manejo de enlaces profundos

Introducción

Los enlaces profundos son URL que llevan a los usuarios directamente a contenido específico en su aplicación. Puede configurar enlaces profundos agregando filtros de intención y extrayendo datos de intentos entrantes para llevar a los usuarios a la pantalla correcta en su aplicación.

Parámetros

Atributo `<data>`	Detalles
esquema	El *esquema* parte de una URI (distingue entre mayúsculas y minúsculas). Ejemplos: `http` , `https` , `ftp`
anfitrión	La parte del *host* de un URI (distingue entre mayúsculas y minúsculas). Ejemplos: `google.com` , `example.org`
Puerto	El *puerto* parte de un URI. Ejemplos: `80` , `443`
camino	La parte del *camino* de un URI. Debe comenzar con `/` . Ejemplos: `/` , `/about`
PathPrefix	Un prefijo para la parte de la *ruta* de un URI. Ejemplos: `/item` , `/article`
pathPattern	Un patrón para que coincida con la parte de la *ruta* de una URI. Ejemplos: `/item/.*` , `/article/[0-9]*`
tipo MIME	Un tipo mime para que coincida. Ejemplos: `image/jpeg` , `audio/*`

Observaciones

El `<intent-filter>`

Esta combinación de elementos `<action>` y `<category>` es lo que le dice al sistema Android que una Actividad específica debe iniciarse cuando el usuario hace clic en un enlace en otra aplicación.

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />

    <data ... />

</intent-filter>
```

Múltiples etiquetas `<data>`

El conjunto de enlaces profundos que admite su `<intent-filter>` es el producto cruzado de todos los elementos `<data>` que usted define en ese filtro de intención. El dominio múltiple, la ruta múltiple y los ejemplos de múltiples esquemas lo demuestran.

Recursos

- Habilitación de enlaces profundos para contenido de la aplicación (developer.android.com)
- `<intent-filter>` (developer.android.com

Examples

Enlace profundo simple

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />

    </intent-filter>

</activity>
```

Esto aceptará cualquier enlace que comience con `http://www.example.com` como un enlace profundo para iniciar su `MainActivity` .

Múltiples rutas en un solo dominio

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />

        <data android:path="/" />
        <data android:path="/about" />
        <data android:path="/map" />
```

```
    </intent-filter>

</activity>
```

Esto lanzará su `MainActivity` cuando el usuario haga clic en cualquiera de estos enlaces:

- `http://www.example.com/`
- `http://www.example.com/about`
- `http://www.example.com/map`

Múltiples dominios y múltiples caminos.

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />

        <data android:scheme="http"
              android:host="www.example2.com" />

        <data android:path="/" />
        <data android:path="/map" />

    </intent-filter>

</activity>
```

Esto lanzará su MainActivity cuando el usuario haga clic en cualquiera de estos enlaces:

- `http://www.example.com/`
- `http://www.example2.com/`
- `http://www.example.com/map`
- `http://www.example2.com/map`

Tanto http como https para el mismo dominio.

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http" />
        <data android:scheme="https" />
```

```
        <data android:host="www.example.com" />

        <data android:path="/" />
        <data android:path="/map" />

    </intent-filter>

</activity>
```

Esto lanzará su MainActivity cuando el usuario haga clic en cualquiera de estos enlaces:

- `http://www.example.com/`
- `https://www.example.com/`
- `http://www.example.com/map`
- `https://www.example.com/map`

Recuperando parámetros de consulta

```
public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Intent intent = getIntent();
        Uri data = intent.getData();

        if (data != null) {
            String param1 = data.getQueryParameter("param1");
            String param2 = data.getQueryParameter("param2");
        }
    }

}
```

Si el usuario hace clic en un enlace para `http://www.example.com/map?param1=FOO&param2=BAR` , entonces `param1` tendrá un valor de `"FOO"` y `param2` tendrá un valor de `"BAR"` .

Usando pathPrefix

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com"
              android:path="/item" />

    </intent-filter>
```

```
</activity>
```

Esto lanzará su MainActivity cuando el usuario haga clic en cualquier enlace que comience con `http://www.example.com/item` , como:

- `https://www.example.com/item`
- `http://www.example.com/item/1234`
- `https://www.example.com/item/xyz/details`

Lea Manejo de enlaces profundos en línea: https://riptutorial.com/es/android/topic/3716/manejo-de-enlaces-profundos

Capítulo 157: Manejo de eventos táctiles y de movimiento.

Introducción

Un resumen de algunos de los sistemas básicos de manejo táctil / movimiento en la API de Android.

Parámetros

Oyente	Detalles
onTouchListener	Maneja un solo toque para botones, superficies y más
onTouchEvent	Un oyente que se puede encontrar en superficies (por ejemplo, SurfaceView). No es necesario configurarlo como otros oyentes (e, g. OnTouchListener)
onLongTouch	Similar a onTouch, pero escucha pulsaciones largas en botones, superficies y más.

Examples

Botones

Los eventos táctiles relacionados con un `Button` se pueden verificar de la siguiente manera:

```
public class ExampleClass extends Activity implements View.OnClickListener,
View.OnLongClickListener{
    public Button onLong, onClick;

    @Override
    public void onCreate(Bundle sis){
        super.onCreate(sis);
        setContentView(R.layout.layout);
        onLong = (Button) findViewById(R.id.onLong);
        onClick = (Button) findViewById(R.id.onClick);
        // The buttons are created. Now we need to tell the system that
        // these buttons have a listener to check for touch events.
        // "this" refers to this class, as it contains the appropriate event listeners.
        onLong.setOnLongClickListener(this);
        onClick.setOnClickListener(this);

        [OR]

        onClick.setOnClickListener(new View.OnClickListener(){
            @Override
```

```
            public void onClick(View v){
                // Take action. This listener is only designed for one button.
                // This means, no other input will come here.
                // This makes a switch statement unnecessary here.
            }
        });

        onLong.setOnLongClickListener(new View.OnLongClickListener(){
            @Override
            public boolean onLongClick(View v){
                // See comment in onClick.setOnClickListener().
            }
        });
    }

    @Override
    public void onClick(View v) {
        // If you have several buttons to handle, use a switch to handle them.
        switch(v.getId()){
            case R.id.onClick:
                // Take action.
                break;
        }
    }

    @Override
    public boolean onLongClick(View v) {
        // If you have several buttons to handle, use a switch to handle them.
        switch(v.getId()){
            case R.id.onLong:
                // Take action.
                break;
        }
        return false;
    }
}
```

Superficie

Toque el controlador de eventos para superficies (por ejemplo, SurfaceView, GLSurfaceView y otros):

```
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.SurfaceView;
import android.view.View;

public class ExampleClass extends Activity implements View.OnTouchListener{
    @Override
    public void onCreate(Bundle sis){
        super.onCreate(sis);
        CustomSurfaceView csv = new CustomSurfaceView(this);
        csv.setOnTouchListener(this);
        setContentView(csv);
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
```

```
        // Add a switch (see buttons example) if you handle multiple views
        // here you can see (using MotionEvent event) to see what touch event
        // is being taken. Is the pointer touching or lifted? Is it moving?
        return false;
    }
}
```

O alternativamente (en la superficie):

```
public class CustomSurfaceView extends SurfaceView {
    @Override
    public boolean onTouchEvent(MotionEvent ev) {
        super.onTouchEvent(ev);
        // Handle touch events here. When doing this, you do not need to call a listener.
        // Please note that this listener only applies to the surface it is placed in
        // (in this case, CustomSurfaceView), which means that anything else which is
        // pressed outside the SurfaceView is handled by the parts of your app that
        // have a listener in that area.
        return true;
    }
}
```

Manipulación multitáctil en una superficie.

```
public class CustomSurfaceView extends SurfaceView {
    @Override
    public boolean onTouchEvent(MotionEvent e) {
        super.onTouchEvent(e);
        if(e.getPointerCount() > 2){
            return false; // If we want to limit the amount of pointers, we return false
                // which disallows the pointer. It will not be reacted on either, for
                // any future touch events until it has been lifted and repressed.
        }

        // What can you do here? Check if the amount of pointers are [x] and take action,
        // if a pointer leaves, a new enters, or the [x] pointers are moved.
        // Some examples as to handling etc. touch/motion events.

        switch (MotionEventCompat.getActionMasked(e)) {
            case MotionEvent.ACTION_DOWN:
            case MotionEvent.ACTION_POINTER_DOWN:
                // One or more pointers touch the screen.
                break;
            case MotionEvent.ACTION_UP:
            case MotionEvent.ACTION_POINTER_UP:
                // One or more pointers stop touching the screen.
                break;
            case MotionEvent.ACTION_MOVE:
                // One or more pointers move.
                if(e.getPointerCount() == 2){
                    move();
                }else if(e.getPointerCount() == 1){
                    paint();
                }else{
                    zoom();
                }
                break;
        }
```

```
        return true; // Allow repeated action.
    }
}
```

Lea Manejo de eventos táctiles y de movimiento. en línea:
https://riptutorial.com/es/android/topic/9315/manejo-de-eventos-tactiles-y-de-movimiento-

Capítulo 158: Mapeo de puertos usando la biblioteca Cling en Android

Examples

Añadiendo soporte de Cling a tu proyecto de Android

construir.gradle

```
repositories {
  maven { url 'http://4thline.org/m2' }
}

dependencies {

// Cling
compile 'org.fourthline.cling:cling-support:2.1.0'

//Other dependencies required by Cling
compile 'org.eclipse.jetty:jetty-server:8.1.18.v20150929'
compile 'org.eclipse.jetty:jetty-servlet:8.1.18.v20150929'
compile 'org.eclipse.jetty:jetty-client:8.1.18.v20150929'
compile 'org.slf4j:slf4j-jdk14:1.7.14'

}
```

Mapeo de un puerto NAT

```
String myIp = getIpAddress();
int port = 55555;

//creates a port mapping configuration with the external/internal port, an internal host IP,
the protocol and an optional description
PortMapping[] desiredMapping = new PortMapping[2];
desiredMapping[0] = new PortMapping(port,myIp, PortMapping.Protocol.TCP);
desiredMapping[1] = new PortMapping(port,myIp, PortMapping.Protocol.UDP);

//starting the UPnP service
UpnpService upnpService = new UpnpServiceImpl(new AndroidUpnpServiceConfiguration());
RegistryListener registryListener = new PortMappingListener(desiredMapping);
upnpService.getRegistry().addListener(registryListener);
upnpService.getControlPoint().search();



//method for getting local ip
private String getIpAddress() {
    String ip = "";
    try {
        Enumeration<NetworkInterface> enumNetworkInterfaces = NetworkInterface
                .getNetworkInterfaces();
        while (enumNetworkInterfaces.hasMoreElements()) {
            NetworkInterface networkInterface = enumNetworkInterfaces
```

```
                .nextElement();
        Enumeration<InetAddress> enumInetAddress = networkInterface
                .getInetAddresses();
        while (enumInetAddress.hasMoreElements()) {
            InetAddress inetAddress = enumInetAddress.nextElement();

            if (inetAddress.isSiteLocalAddress()) {
                ip +=inetAddress.getHostAddress();
            }
        }
    }
 } catch (SocketException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    ip += "Something Wrong! " + e.toString() + "\n";
 }
 return ip;
}
```

Lea Mapeo de puertos usando la biblioteca Cling en Android en línea: https://riptutorial.com/es/android/topic/6208/mapeo-de-puertos-usando-la-biblioteca-cling-en-android

Capítulo 159: MediaSession

Sintaxis

- void mediaSessionCompat.setFlags (int flags)
- void mediaSessionCompat.setMediaButtonReceiver (PendingIntent mbr)
- void mediaSessionCompat.setCallback (MediaSessionCompat.Callback callback)
- void mediaSessionCompat.setActive (booleano activo)
- MediaSessionCompat.Token mediaSessionCompat.getSessionToken ()
- void mediaSessionCompat.release ()
- void mediaSessionCompat.setPlaybackState (estado de PlaybackStateCompat)
- void mediaSessionCompat.setMetadata (metadatos de MediaMetadataCompat)

Observaciones

Para las mejores prácticas, use la biblioteca de **compatibilidad de medios** . La biblioteca se encarga de la compatibilidad con versiones anteriores al traducir los métodos de sesión de medios a los métodos equivalentes en versiones anteriores de la plataforma cuando estén disponibles.

Examples

Recepción y manejo de eventos de botones.

Este ejemplo crea un objeto `MediaSession` cuando se inicia un `Service` . El objeto `MediaSession` se libera cuando el `Service` se destruye:

```
public final class MyService extends Service {
    private static MediaSession s_mediaSession;

    @Override
    public void onCreate() {
        // Instantiate new MediaSession object.
        configureMediaSession();
    }

    @Override
    public void onDestroy() {
        if (s_mediaSession != null)
            s_mediaSession.release();
    }
}
```

El siguiente método `MediaSession` instancia y configura las devoluciones de llamada del botón `MediaSession` :

```
private void configureMediaSession {
```

```
    s_mediaSession = new MediaSession(this, "MyMediaSession");

    // Overridden methods in the MediaSession.Callback class.
    s_mediaSession.setCallback(new MediaSession.Callback() {
        @Override
        public boolean onMediaButtonEvent(Intent mediaButtonIntent) {
            Log.d(TAG, "onMediaButtonEvent called: " + mediaButtonIntent);
            KeyEvent ke = mediaButtonIntent.getParcelableExtra(Intent.EXTRA_KEY_EVENT);
            if (ke != null && ke.getAction() == KeyEvent.ACTION_DOWN) {
                int keyCode = ke.getKeyCode();
                Log.d(TAG, "onMediaButtonEvent Received command: " + ke);
            }
            return super.onMediaButtonEvent(mediaButtonIntent);
        }

        @Override
        public void onSkipToNext() {
            Log.d(TAG, "onSkipToNext called (media button pressed)");
            Toast.makeText(getApplicationContext(), "onSkipToNext called",
Toast.LENGTH_SHORT).show();
            skipToNextPlaylistItem(); // Handle this button press.
            super.onSkipToNext();
        }

        @Override
        public void onSkipToPrevious() {
            Log.d(TAG, "onSkipToPrevious called (media button pressed)");
            Toast.makeText(getApplicationContext(), "onSkipToPrevious called",
Toast.LENGTH_SHORT).show();
            skipToPreviousPlaylistItem(); // Handle this button press.
            super.onSkipToPrevious();
        }

        @Override
        public void onPause() {
            Log.d(TAG, "onPause called (media button pressed)");
            Toast.makeText(getApplicationContext(), "onPause called",
Toast.LENGTH_SHORT).show();
            mpPause(); // Pause the player.
            super.onPause();
        }

        @Override
        public void onPlay() {
            Log.d(TAG, "onPlay called (media button pressed)");
            mpStart(); // Start player/playback.
            super.onPlay();
        }

        @Override
        public void onStop() {
            Log.d(TAG, "onStop called (media button pressed)");
            mpReset(); // Stop and/or reset the player.
            super.onStop();
        }
    });

    s_mediaSession.setFlags(MediaSession.FLAG_HANDLES_MEDIA_BUTTONS |
MediaSession.FLAG_HANDLES_TRANSPORT_CONTROLS);
    s_mediaSession.setActive(true);
}
```

El siguiente método envía metadatos (almacenados en un `HashMap`) al dispositivo utilizando A2DP:

```
void sendMetaData(@NonNull final HashMap<String, String> hm) {
    // Return if Bluetooth A2DP is not in use.
    if (!((AudioManager) getSystemService(Context.AUDIO_SERVICE)).isBluetoothA2dpOn()) return;

    MediaMetadata metadata = new MediaMetadata.Builder()
            .putString(MediaMetadata.METADATA_KEY_TITLE, hm.get("Title"))
            .putString(MediaMetadata.METADATA_KEY_ALBUM, hm.get("Album"))
            .putString(MediaMetadata.METADATA_KEY_ARTIST, hm.get("Artist"))
            .putString(MediaMetadata.METADATA_KEY_AUTHOR, hm.get("Author"))
            .putString(MediaMetadata.METADATA_KEY_COMPOSER, hm.get("Composer"))
            .putString(MediaMetadata.METADATA_KEY_WRITER, hm.get("Writer"))
            .putString(MediaMetadata.METADATA_KEY_DATE, hm.get("Date"))
            .putString(MediaMetadata.METADATA_KEY_GENRE, hm.get("Genre"))
            .putLong(MediaMetadata.METADATA_KEY_YEAR, tryParse(hm.get("Year")))
            .putLong(MediaMetadata.METADATA_KEY_DURATION, tryParse(hm.get("Raw Duration")))
            .putLong(MediaMetadata.METADATA_KEY_TRACK_NUMBER, tryParse(hm.get("Track
Number")))
            .build();

    s_mediaSession.setMetadata(metadata);
}
```

El siguiente método establece el `PlaybackState` . También establece a qué acciones de botón responderá `MediaSession` :

```
private void setPlaybackState(@NonNull final int stateValue) {
    PlaybackState state = new PlaybackState.Builder()
            .setActions(PlaybackState.ACTION_PLAY | PlaybackState.ACTION_SKIP_TO_NEXT
                    | PlaybackState.ACTION_PAUSE | PlaybackState.ACTION_SKIP_TO_PREVIOUS
                    | PlaybackState.ACTION_STOP | PlaybackState.ACTION_PLAY_PAUSE)
            .setState(stateValue, PlaybackState.PLAYBACK_POSITION_UNKNOWN, 0)
            .build();

    s_mediaSession.setPlaybackState(state);
}
```

Lea MediaSession en línea: https://riptutorial.com/es/android/topic/6250/mediasession

Capítulo 160: Mediastore

Examples

Obtenga archivos de audio / MP3 de una carpeta específica del dispositivo o busque todos los archivos

Primero, agregue los siguientes permisos al manifiesto de su proyecto para habilitar el acceso al almacenamiento del dispositivo:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Luego, cree el archivo *AudioModel.class* y coloque la siguiente clase de modelo para permitir obtener y configurar los elementos de la lista:

```
public class AudioModel {
    String aPath;
    String aName;
    String aAlbum;
    String aArtist;

    public String getaPath() {
        return aPath;
    }
    public void setaPath(String aPath) {
        this.aPath = aPath;
    }
    public String getaName() {
        return aName;
    }
    public void setaName(String aName) {
        this.aName = aName;
    }
    public String getaAlbum() {
        return aAlbum;
    }
    public void setaAlbum(String aAlbum) {
        this.aAlbum = aAlbum;
    }
    public String getaArtist() {
        return aArtist;
    }
    public void setaArtist(String aArtist) {
        this.aArtist = aArtist;
    }
}
```

Luego, use el siguiente método para leer todos los archivos MP3 de una carpeta de su dispositivo o para leer todos los archivos de su dispositivo:

```
public List<AudioModel> getAllAudioFromDevice(final Context context) {
```

```
    final List<AudioModel> tempAudioList = new ArrayList<>();

    Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
    String[] projection = {MediaStore.Audio.AudioColumns.DATA,
MediaStore.Audio.AudioColumns.TITLE, MediaStore.Audio.AudioColumns.ALBUM,
MediaStore.Audio.ArtistColumns.ARTIST,};
    Cursor c = context.getContentResolver().query(uri, projection, MediaStore.Audio.Media.DATA
+ " like ? ", new String[]{"%utm%"}, null);

    if (c != null) {
        while (c.moveToNext()) {
            AudioModel audioModel = new AudioModel();
            String path = c.getString(0);
            String name = c.getString(1);
            String album = c.getString(2);
            String artist = c.getString(3);

            audioModel.setaName(name);
            audioModel.setaAlbum(album);
            audioModel.setaArtist(artist);
            audioModel.setaPath(path);

            Log.e("Name :" + name, " Album :" + album);
            Log.e("Path :" + path, " Artist :" + artist);

            tempAudioList.add(audioModel);
        }
        c.close();
    }

    return tempAudioList;
}
```

El código anterior devolverá una lista de todos los archivos MP3 con el nombre, la ruta, el artista y el álbum de la música. Para más detalles, consulte la documentación de Media.Store.Audio .

Para leer archivos de una carpeta específica, use la siguiente consulta (debe reemplazar el nombre de la carpeta):

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    MediaStore.Audio.Media.DATA + " like ? ",
    new String[]{"%yourFolderName%"}, // Put your device folder / file location here.
    null);
```

Si desea recuperar todos los archivos de su dispositivo, utilice la siguiente consulta:

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    null,
    null,
    null);
```

Nota: No olvide habilitar los permisos de acceso al almacenamiento.

Ahora, todo lo que tiene que hacer es llamar al método anterior para obtener los archivos MP3:

```
getAllAudioFromDevice(this);
```

Ejemplo con Actividad

```
public class ReadAudioFilesActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_audio_list);

        /**
         * This will return a list of all MP3 files. Use the list to display data.
         */
        getAllAudioFromDevice(this);
    }

    // Method to read all the audio/MP3 files.
    public List<AudioModel> getAllAudioFromDevice(final Context context) {
        final List<AudioModel> tempAudioList = new ArrayList<>();

        Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
        String[] projection =
{MediaStore.Audio.AudioColumns.DATA,MediaStore.Audio.AudioColumns.TITLE
,MediaStore.Audio.AudioColumns.ALBUM, MediaStore.Audio.ArtistColumns.ARTIST,};
        Cursor c = context.getContentResolver().query(uri, projection,
MediaStore.Audio.Media.DATA + " like ? ", new String[]{"%utm%"}, null);

        if (c != null) {
            while (c.moveToNext()) {
                // Create a model object.
                AudioModel audioModel = new AudioModel();

                String path = c.getString(0);   // Retrieve path.
                String name = c.getString(1);   // Retrieve name.
                String album = c.getString(2);  // Retrieve album name.
                String artist = c.getString(3); // Retrieve artist name.

                // Set data to the model object.
                audioModel.setaName(name);
                audioModel.setaAlbum(album);
                audioModel.setaArtist(artist);
                audioModel.setaPath(path);

                Log.e("Name :" + name, " Album :" + album);
                Log.e("Path :" + path, " Artist :" + artist);

                // Add the model object to the list .
                tempAudioList.add(audioModel);
            }
            c.close();
        }

        // Return the list.
        return tempAudioList;
    }
}
```

Lea Mediastore en línea: https://riptutorial.com/es/android/topic/7136/mediastore

Capítulo 161: Mejora de los diálogos de alerta

Introducción

Este tema trata sobre la mejora de un `AlertDialog` con características adicionales.

Examples

Cuadro de diálogo de alerta que contiene un enlace cliqueable

Para mostrar un cuadro de diálogo de alerta que contiene un enlace que se puede abrir haciendo clic en él, puede usar el siguiente código:

```
AlertDialog.Builder builder1 = new AlertDialog.Builder(youractivity.this);

builder1.setMessage(Html.fromHtml("your message,<a href=\"http://www.google.com\">link</a>"));

builder1.setCancelable(false);
builder1.setPositiveButton("ok", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
    }
});

AlertDialog Alert1 = builder1.create();
Alert1 .show();
((TextView)Alert1.findViewById(android.R.id.message)).setMovementMethod(LinkMovementMethod.getInstance
```

Lea Mejora de los diálogos de alerta en línea:
https://riptutorial.com/es/android/topic/10163/mejora-de-los-dialogos-de-alerta

Capítulo 162: Mejora del rendimiento de Android utilizando fuentes de iconos

Observaciones

Las fuentes de iconos son como los tipos de fuentes normales que tienen símbolos en lugar de letras. Se puede utilizar en su aplicación con la mayor facilidad.

Son:

- Flexible
- Escalable
- Vectores
- Procesable rapido
- Peso ligero
- Accesible

Efecto sobre el tamaño

Exportar una imagen en varios tamaños para dispositivos Android costaría su aplicación, un tamaño de activo adicional de alrededor de 30kB por imagen. Al agregar un archivo de fuente (.ttf) con alrededor de 36 iconos, solo costaría 9kB. Solo imagine el caso si está agregando 36 archivos individuales de varias configuraciones que serían alrededor de 1000kB. Es una cantidad razonable de espacio que ahorrará al usar fuentes de iconos.

Limitaciones de las fuentes de iconos.

- Las fuentes de iconos se pueden utilizar en el cajón de navegación. No es posible utilizarlos en las vistas de navegación como icono de elementos de menú, ya que el archivo de menú no se puede crear sin especificar el título. Por lo tanto, es recomendable utilizar archivos svg como recursos para estos íconos.
- Las fuentes de iconos no se pueden utilizar en el botón de acción flotante. ya que no tienen un atributo `setText()` .
- Las fuentes externas no se pueden aplicar desde xml. Deben especificarse utilizando el archivo java. O bien, debe ampliar la vista básica y crear una vista como se especifica en esta publicación.

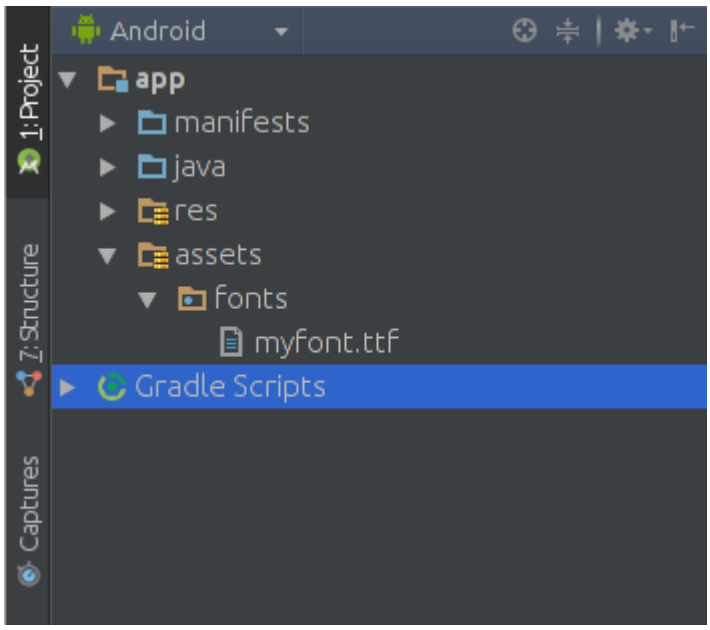
Examples

Cómo integrar fuentes de iconos

Para utilizar las fuentes de iconos, simplemente siga los pasos a continuación:

- **Agrega el archivo de fuente a tu proyecto**

  Puede crear su archivo de icono de fuente a partir de sitios web en línea como icomoon , donde puede cargar archivos SVG de los iconos requeridos y luego descargar la fuente de icono creada. Luego, coloque el *archivo de* fuentes *.ttf* en una carpeta llamada *fuentes* (nombre como desee) en la carpeta de activos:



- **Crear una clase de ayuda**

  Ahora, cree la siguiente clase auxiliar, de modo que pueda evitar repetir el código de inicialización para la fuente:

```
public class FontManager {
  public static final String ROOT = "fonts/";
  FONT_AWESOME = ROOT + "myfont.ttf";
  public static Typeface getTypeface(Context context) {
    return Typeface.createFromAsset(context.getAssets(), FONT_AWESOME);
  }
}
```

  Puede utilizar la clase `Typeface` para elegir la fuente de los activos. De esta manera puede establecer el tipo de letra para varias vistas, por ejemplo, para un botón:

```
Button button=(Button) findViewById(R.id.button);
Typeface iconFont=FontManager.getTypeface(getApplicationContext());
button.setTypeface(iconFont);
```

  Ahora, el tipo de letra del botón se ha cambiado a la fuente de icono recién creada.

- **Recoge los iconos que quieras.**

  Abra el archivo *styles.css* adjunto a la fuente del icono. Allí encontrarás los estilos con caracteres Unicode de tus iconos:

```
.icon-arrow-circle-down:before {
  content: "\e001";
}
.icon-arrow-circle-left:before {
  content: "\e002";
}
.icon-arrow-circle-o-down:before {
  content: "\e003";
}
.icon-arrow-circle-o-left:before {
  content: "\e004";
}
```

Este archivo de recursos servirá como un diccionario, que asigna el carácter Unicode asociado con un icono específico a un nombre legible para el ser humano. Ahora, crea los recursos de cadena de la siguiente manera:

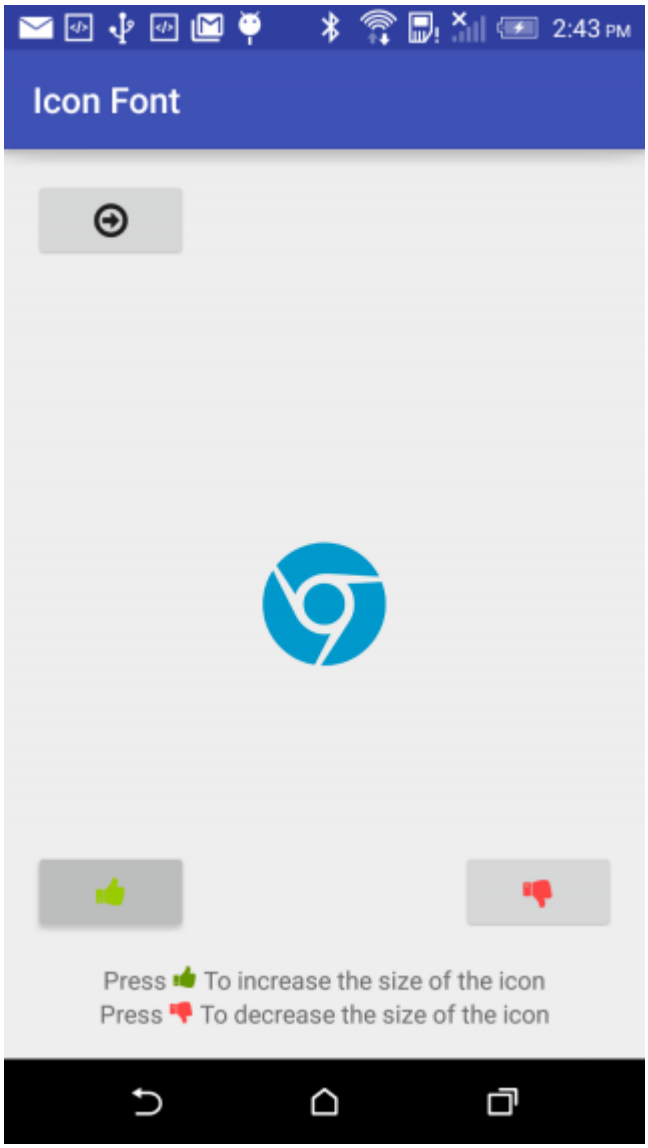
```
<resources>
  <! — Icon Fonts -->
  <string name="icon_arrow_circle_down">&#xe001; </string>
  <string name="icon_arrow_circle_left">&#xe002; </string>
  <string name="icon_arrow_circle-o_down">&#xe003; </string>
  <string name="icon_arrow_circle_o_left">&#xe004; </string>
</resources>
```

- **Usa los iconos en tu código**

  Ahora, puede usar su fuente en varias vistas, por ejemplo, de la siguiente manera:

```
button.setText(getString(R.string.icon_arrow_circle_left))
```

  También puede crear vistas de texto de botón usando las fuentes de iconos:



TabLayout con fuentes de iconos

```
public class TabAdapter extends FragmentPagerAdapter {

CustomTypefaceSpan fonte;
List<Fragment> fragments = new ArrayList<>(4);
private String[] icons = {"\ue001","\uE002","\uE003","\uE004"};

public TabAdapter(FragmentManager fm, CustomTypefaceSpan fonte) {
    super(fm);
    this.fonte = fonte
    for (int i = 0; i < 4; i++){
        fragments.add(MyFragment.newInstance());
    }
}

public List<Fragment> getFragments() {
    return fragments;
}

@Override
public Fragment getItem(int position) {
    return fragments.get(position);
}
```

```
@Override
public CharSequence getPageTitle(int position) {
    SpannableStringBuilder ss = new SpannableStringBuilder(icons[position]);
    ss.setSpan(fonte,0,ss.length(), Spanned.SPAN_INCLUSIVE_INCLUSIVE);
    ss.setSpan(new RelativeSizeSpan(1.5f),0,ss.length(), Spanned.SPAN_INCLUSIVE_INCLUSIVE );
    return ss;
}



@Override
public int getCount() {
    return 4;
}

}
```

- En este ejemplo, myfont.ttf está en la carpeta Activos. Creando carpeta de activos
- En tu clase de actividad

```
//..
TabLayout tabs;
ViewPager tabs_pager;
public CustomTypefaceSpan fonte;
//..

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //...
    fm = getSupportFragmentManager();
    fonte = new
CustomTypefaceSpan("icomoon",Typeface.createFromAsset(getAssets(),"myfont.ttf"));
    this.tabs = ((TabLayout) hasViews.findViewById(R.id.tabs));
    this.tabs_pager = ((ViewPager) hasViews.findViewById(R.id.tabs_pager));
    //...
}

@Override
protected void onStart() {
    super.onStart();
    //..
    tabs_pager.setAdapter(new TabAdapter(fm,fonte));
    tabs.setupWithViewPager(tabs_pager);
    //..
```

Lea Mejora del rendimiento de Android utilizando fuentes de iconos en línea: https://riptutorial.com/es/android/topic/3642/mejora-del-rendimiento-de-android-utilizando-fuentes-de-iconos

Capítulo 163: Menú

Sintaxis

- inflater.inflate (R.menu.your_xml_file, menu);

Parámetros

Parámetro	Descripción
`inflate(int menuRes, Menu menu)`	Inflar una jerarquía de menús del recurso XML especificado.
`getMenuInflater ()`	Devuelve un `MenuInflater` con este contexto.
`onCreateOptionsMenu (Menu menu)`	Inicialice los contenidos del menú de opciones estándar de la Actividad. Debe colocar los elementos de su menú en el menú.
`onOptionsItemSelected (MenuItem item)`	Este método se llama siempre que se selecciona un elemento en su menú de opciones

Observaciones

Para saber más sobre los **menús** , lea esto . ¡Espero eso ayude!

Examples

Menú de opciones con separadores.

En Android hay un menú de opciones por defecto, que puede tomar varias opciones. Si es necesario mostrar un número mayor de opciones, entonces tiene sentido agrupar esas opciones para mantener la claridad. Las opciones pueden agruparse colocando separadores (es decir, líneas horizontales) entre ellas. Para permitir divisiones, se puede utilizar el siguiente tema:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
    <item name="android:dropDownListViewStyle">@style/PopupMenuListView</item>
</style>
<style name="PopupMenuListView" parent="@style/Widget.AppCompat.ListView.DropDown">
    <item name="android:divider">@color/black</item>
    <item name="android:dividerHeight">1dp</item>
</style>
```

Al cambiar el tema, se pueden agregar divisiones a un menú.

Aplicar fuente personalizada al menú

```
public static void applyFontToMenu(Menu m, Context mContext){
    for(int i=0;i<m.size();i++) {
        applyFontToMenuItem(m.getItem(i),mContext);
    }
}
public static void applyFontToMenuItem(MenuItem mi, Context mContext) {
    if(mi.hasSubMenu())
        for(int i=0;i<mi.getSubMenu().size();i++) {
            applyFontToMenuItem(mi.getSubMenu().getItem(i),mContext);
        }
    Typeface font = Typeface.createFromAsset(mContext.getAssets(),
"fonts/yourCustomFont.ttf");
    SpannableString mNewTitle = new SpannableString(mi.getTitle());
    mNewTitle.setSpan(new CustomTypefaceSpan("", font, mContext), 0, mNewTitle.length(),
Spannable.SPAN_INCLUSIVE_INCLUSIVE);
    mi.setTitle(mNewTitle);
}
```

y luego en la Actividad:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    applyFontToMenu(menu,this);
    return true;
}
```

Creando un Menú en una Actividad

Para definir su propio menú, cree un archivo XML dentro del directorio `res/menu/` del proyecto y cree el menú con los siguientes elementos:

- `<menu>` : define un menú, que contiene todos los elementos del menú.
- `<item>` : crea un MenuItem, que representa un solo elemento en un menú. También podemos crear un elemento anidado para crear un submenú.

Paso 1:

Crea tu propio archivo xml de la siguiente manera:

En `res/menu/main_menu.xml` :

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/aboutMenu"
```

```
        android:title="About" />
    <item
        android:id="@+id/helpMenu"
        android:title="Help" />
    <item
        android:id="@+id/signOutMenu"
        android:title="Sign Out" />
</menu>
```

Paso 2:

Para especificar el menú de opciones, anule `onCreateOptionsMenu()` en su *actividad* .

En este método, puede inflar su recurso de menú (definido en su archivo XML, es decir, `res/menu/main_menu.xml`)

```
 @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }
```

Cuando el usuario selecciona un elemento del menú de opciones, el sistema llama al método `onOptionsItemSelected()` anulado de su *actividad* .

- Este método pasa el MenuItem seleccionado.
- Puede identificar el elemento llamando a `getItemId()` , que devuelve la ID única para el elemento de menú (definida por el `android:id attribute` en el recurso de menú - `res/menu/main_menu.xml`) * /

```
 @Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.aboutMenu:
            Log.d(TAG, "Clicked on About!");
            // Code for About goes here
            return true;
        case R.id.helpMenu:
            Log.d(TAG, "Clicked on Help!");
            // Code for Help goes here
            return true;
        case R.id.signOutMenu:
            Log.d(TAG, "Clicked on Sign Out!");
            // SignOut method call goes here
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

¡Terminando!

Su código de `Activity` debe verse como a continuación:

```
public class MainActivity extends AppCompatActivity {

    private static final String TAG = "mytag";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.aboutMenu:
                Log.d(TAG, "Clicked on About!");
                // Code for About goes here
                return true;
            case R.id.helpMenu:
                Log.d(TAG, "Clicked on Help!");
                // Code for Help goes here
                return true;
            case R.id.signOutMenu:
                Log.d(TAG, "User signed out");
                // SignOut method call goes here
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

Captura de pantalla de cómo se ve tu propio menú:



Lea Menú en línea: https://riptutorial.com/es/android/topic/2028/menu

Capítulo 164: Métricas de la pantalla del dispositivo

Examples

Consigue las dimensiones de las pantallas en píxeles.

Para recuperar el ancho y la altura de las pantallas en píxeles, podemos hacer uso de las métricas de visualización de WindowManagers .

```
// Get display metrics
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
```

Estos DisplayMetrics contienen una serie de información sobre la pantalla del dispositivo, como su densidad o tamaño:

```
// Get width and height in pixel
Integer heightPixels = metrics.heightPixels;
Integer widthPixels = metrics.widthPixels;
```

Obtener densidad de pantalla

Para obtener la densidad de las pantallas, también podemos utilizar DisplayMetrics de Windowmanagers . Este es un ejemplo rápido:

```
// Get density in dpi
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
int densityInDpi =  metrics.densityDpi;
```

Fórmula px a dp, dp a px conversación

DP a Pixel:

```
private int dpToPx(int dp)
{
    return (int) (dp * Resources.getSystem().getDisplayMetrics().density);
}
```

Pixel a DP:

```
private int pxToDp(int px)
{
    return (int) (px / Resources.getSystem().getDisplayMetrics().density);
}
```

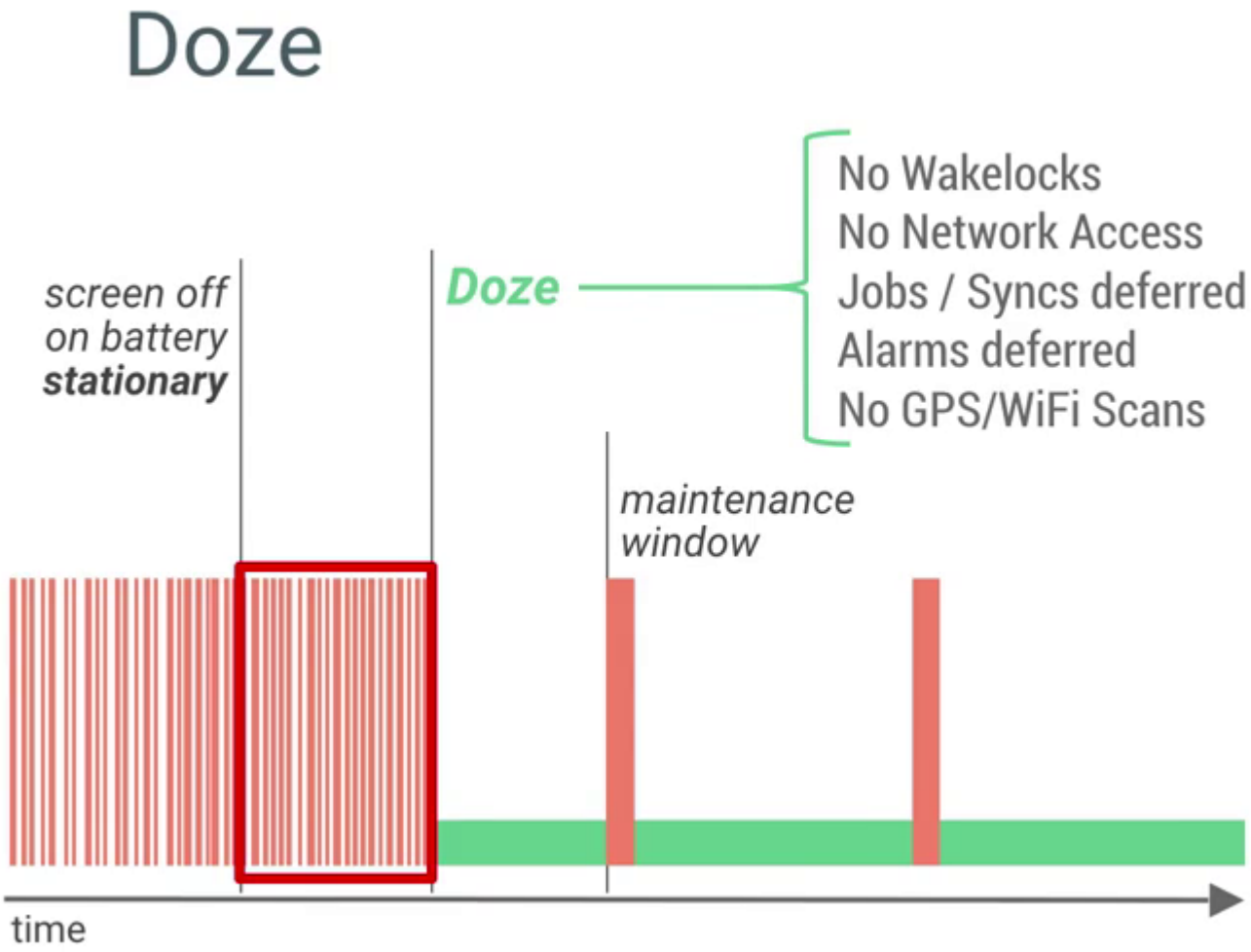
Lea Métricas de la pantalla del dispositivo en línea:
https://riptutorial.com/es/android/topic/4207/metricas-de-la-pantalla-del-dispositivo

Capítulo 165: Modo Doze

Observaciones

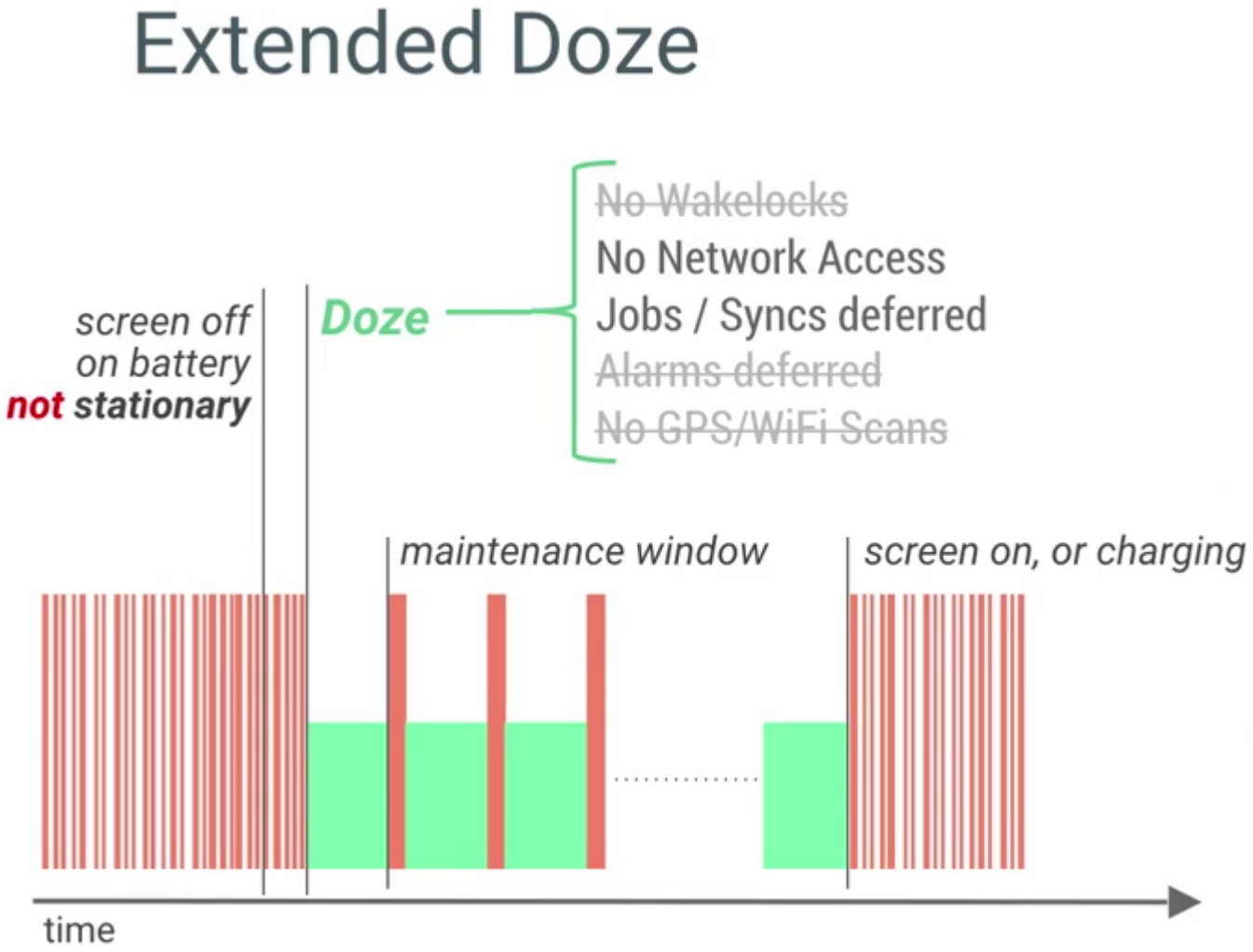
El modo Doze es un conjunto de cambios y reglas que ponen su teléfono en suspensión cuando está inactivo.

En Android 6.0 Marshmallow: el modo Doze se activa después de un tiempo en que la pantalla está apagada, el dispositivo está parado y funciona con batería.



Como puede ver en el diagrama anterior, cuando se activa el Modo Doze, el dispositivo no recibe ningún wakelocks, acceso a la red, trabajos / sincronizaciones, alarmas, GPS / Wi-Fi.

En Android 7.0 Nougat: imagínese si su teléfono está en su bolsillo (la pantalla está apagada, se está quedando sin batería, pero no está estacionada) es posible que también desee obtener las funciones del modo Doze, ¿verdad? Por eso es que Google anunció el Modo Doze extendido: se ejecuta cuando la pantalla está apagada, pero no estacionaria.



Como puede ver en este diagrama, solo el Acceso a la red y los trabajos / sincronizaciones están deshabilitados. Tenga en cuenta que el Doze extendido no reemplaza el primer Modo Doze. Trabajan juntos, dependiendo del estado del teléfono (estacionario o no). Aquí están las distinciones:

	Doze	extended
Trigger	Screen off, on battery, stationary	Screen off, on battery
Timing	Successively increasing periods with maintenance windows	Repeated N-minute periods with maintenance windows
Restrictions	No Network Access Jobs / Syncs deferred No Wakelocks Alarms deferred No GPS/WiFi Scans	No Network Access Jobs / Syncs deferred
Exit	Motion, screen on, alarm clock alarm, or device charging	Screen on or device charging

Los desarrolladores deben ser conscientes de que:

- Doze puede mantener el acceso temporal a wakelock y a la red para mensajes GCM (Google Cloud Messaging) de alta prioridad (para casos donde el usuario necesita una notificación inmediata);
- Los servicios de primer plano (como la reproducción de música) continuarán funcionando.

Puede encontrar más información aquí: https://developer.android.com/training/monitoring-device-state/doze-standby.html

Examples

Excluir la aplicación del uso del modo dormido

1. Abrir la configuración del teléfono
2. batería abierta
3. Abre el menú y selecciona "optimización de batería".
4. En el menú desplegable, seleccione "Todas las aplicaciones".
5. Selecciona la aplicación que deseas incluir en la lista blanca.
6. seleccione "no optimizar"

Ahora esta aplicación se mostrará bajo aplicaciones no optimizadas.

Una aplicación puede verificar si está en la lista blanca llamando a `isIgnoringBatteryOptimizations()`

Lista blanca de una aplicación de Android mediante programación

La lista blanca no deshabilitará el modo de inactividad de su aplicación, pero puede hacerlo mediante el uso de la red y los bloqueos de retención / activación.

La lista blanca de una aplicación de Android se puede hacer mediante programación de la siguiente manera:

```
boolean isIgnoringBatteryOptimizations = pm.isIgnoringBatteryOptimizations(getPackageName());
if(!isIgnoringBatteryOptimizations){
    Intent intent = new Intent();
    intent.setAction(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);
    intent.setData(Uri.parse("package:" + getPackageName()));
    startActivityForResult(intent, MY_IGNORE_OPTIMIZATION_REQUEST);
}
```

El resultado del inicio de la actividad anterior se puede verificar mediante el siguiente código:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == MY_IGNORE_OPTIMIZATION_REQUEST) {
        PowerManager pm = (PowerManager)getSystemService(Context.POWER_SERVICE);
        boolean isIgnoringBatteryOptimizations =
pm.isIgnoringBatteryOptimizations(getPackageName());
        if(isIgnoringBatteryOptimizations){
            // Ignoring battery optimization
        }else{
           // Not ignoring battery optimization
        }
    }
}
```

Lea Modo Doze en línea: https://riptutorial.com/es/android/topic/4719/modo-doze

Capítulo 166: Modo PorterDuff

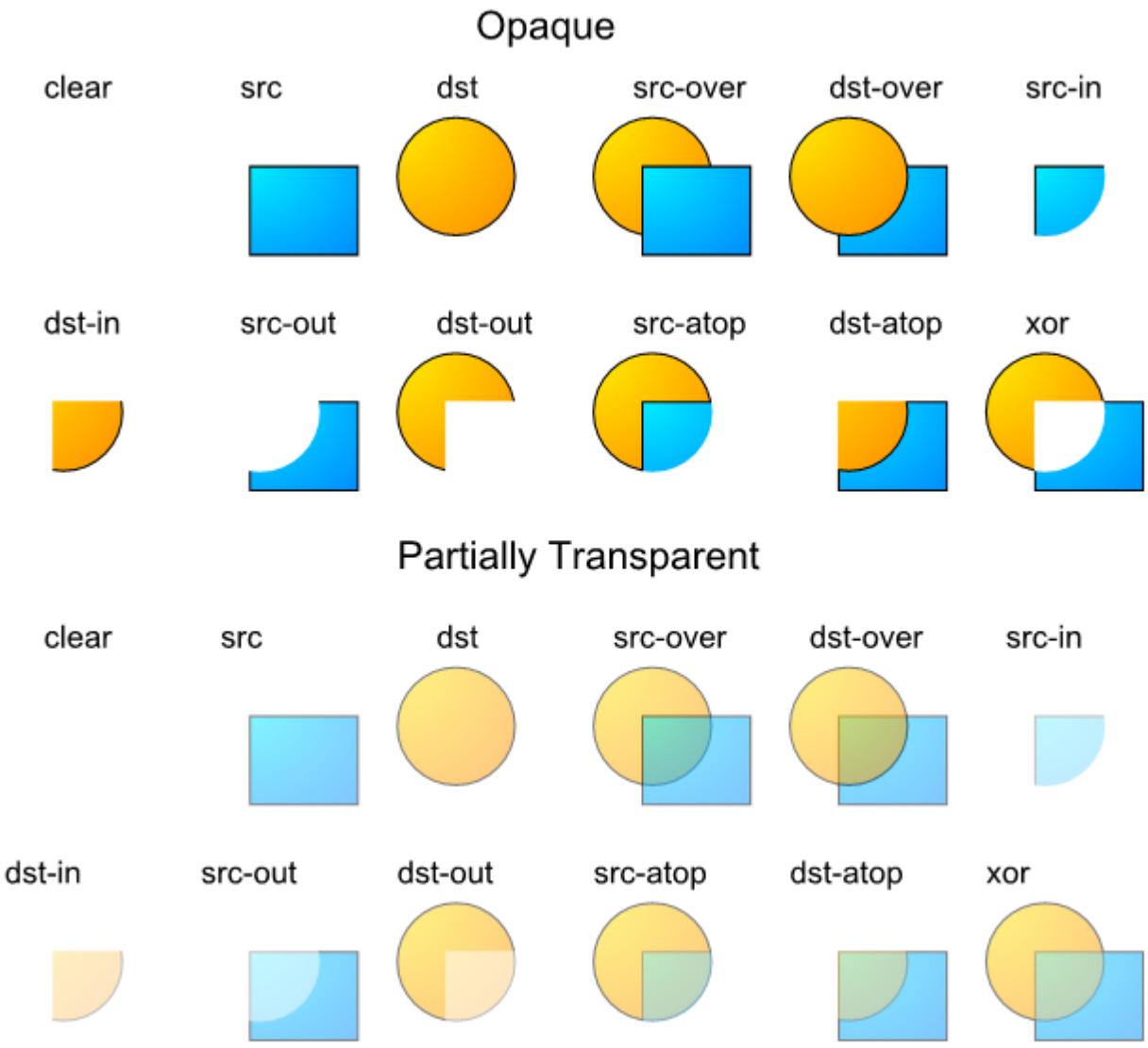
Introducción

PorterDuff se describe como una forma de combinar imágenes como si fueran "piezas de cartón de forma irregular" superpuestas unas sobre otras, así como un esquema para mezclar las partes superpuestas

Observaciones

"Porter Duff" en sí mismo es una técnica de composición alfa que lleva el nombre de un artículo de Thomas Porter y Tom Duff.

Para resumir, la técnica toma dos imágenes con canal alfa y genera la imagen de salida combinando los valores de píxeles de dos imágenes. Los diferentes modos de combinación dan como resultado una imagen de salida diferente. Por ejemplo, en la siguiente imagen, la forma azul (origen, píxeles existentes) se combina con la forma amarilla (destino, nuevos píxeles) en diferentes modos:



Examples

Creando un PorterDuff ColorFilter

`PorterDuff.Mode` se utiliza para crear un `PorterDuffColorFilter` . Un filtro de color modifica el color de cada píxel de un recurso visual.

```
ColorFilter filter = new PorterDuffColorFilter(Color.BLUE, PorterDuff.Mode.SRC_IN);
```

El filtro de arriba teñirá los píxeles no transparentes al color azul.

El filtro de color se puede aplicar a un `Drawable` :

```
drawable.setColorFilter(filter);
```

Se puede aplicar a un `ImageView` :

```
imageView.setColorFilter(filter);
```

Además, se puede aplicar a una `Paint` , de modo que el color que se dibuja con esa pintura sea modificado por el filtro:

```
paint.setColorFilter(filter);
```

Creando un PorterDuff XferMode

Un `Xfermode` (piense en el modo "transferencia") funciona como un paso de transferencia en el trazado de la tubería. Cuando se aplica un `Xfermode` a una `Paint` , los píxeles dibujados con la pintura se combinan con los píxeles subyacentes (ya dibujados) según el modo:

```
paint.setColor(Color.BLUE);
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
```

Ahora tenemos una pintura de tinte azul. Cualquier forma dibujada teñirá de azul los píxeles no transparentes ya existentes en el área de la forma.

Aplique una máscara radial (viñeta) a un mapa de bits usando PorterDuffXfermode

```
/**
 * Apply a radial mask (vignette, i.e. fading to black at the borders) to a bitmap
 * @param imageToApplyMaskTo Bitmap to modify
 */
public static void radialMask(final Bitmap imageToApplyMaskTo) {
    Canvas canvas = new Canvas(imageToApplyMaskTo);

    final float centerX = imageToApplyMaskTo.getWidth() * 0.5f;
    final float centerY = imageToApplyMaskTo.getHeight() * 0.5f;
    final float radius = imageToApplyMaskTo.getHeight() * 0.7f;

    RadialGradient gradient = new RadialGradient(centerX, centerY, radius,
            0x00000000, 0xFF000000, android.graphics.Shader.TileMode.CLAMP);

    Paint p = new Paint();
    p.setShader(gradient);
    p.setColor(0xFF000000);
    p.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.DST_OUT));
    canvas.drawRect(0, 0, imageToApplyMaskTo.getWidth(), imageToApplyMaskTo.getHeight(), p);
}
```

Lea Modo PorterDuff en línea: https://riptutorial.com/es/android/topic/377/modo-porterduff

Capítulo 167: Moshi

Introducción

Moshi es una moderna biblioteca JSON para Android y Java. Facilita el análisis de JSON en objetos Java y Java en JSON.

Observaciones

No lo olvides, siempre lee el archivo README !

Examples

JSON en Java

```
String json = ...;

Moshi moshi = new Moshi.Builder().build();
JsonAdapter<BlackjackHand> jsonAdapter = moshi.adapter(BlackjackHand.class);

BlackjackHand blackjackHand = jsonAdapter.fromJson(json);
System.out.println(blackjackHand);
```

serializar objetos Java como JSON

```
BlackjackHand blackjackHand = new BlackjackHand(
    new Card('6', SPADES),
    Arrays.asList(new Card('4', CLUBS), new Card('A', HEARTS)));

Moshi moshi = new Moshi.Builder().build();
JsonAdapter<BlackjackHand> jsonAdapter = moshi.adapter(BlackjackHand.class);

String json = jsonAdapter.toJson(blackjackHand);
System.out.println(json);
```

Construido en adaptadores de tipo

Moshi tiene soporte incorporado para leer y escribir los tipos de datos centrales de Java:

- Primitivas (int, float, char ...) y sus equivalentes en caja (Integer, Float, Character ...).
- Arrays
- Colecciones
- Liza
- Conjuntos
- Mapas Cadenas Enums

Es compatible con sus clases modelo escribiéndolas campo por campo. En el ejemplo anterior,

Moshi usa estas clases:

```
class BlackjackHand {
  public final Card hidden_card;
  public final List<Card> visible_cards;
  ...
}

class Card {
  public final char rank;
  public final Suit suit;
  ...
}

enum Suit {
  CLUBS, DIAMONDS, HEARTS, SPADES;
}
to read and write this JSON:

{
  "hidden_card": {
    "rank": "6",
    "suit": "SPADES"
  },
  "visible_cards": [
    {
      "rank": "4",
      "suit": "CLUBS"
    },
    {
      "rank": "A",
      "suit": "HEARTS"
    }
  ]
}
```

Lea Moshi en línea: https://riptutorial.com/es/android/topic/8744/moshi

Capítulo 168: Multidex y el método Dex Limit

Introducción

DEX significa archivos de código de bytes ejecutables de la aplicación de Android (APK) en forma de archivos ejecutables de Dalvik (DEX), que contienen el código compilado utilizado para ejecutar la aplicación.

La especificación de Dalvik Executable limita el número total de métodos a los que se puede hacer referencia dentro de un solo archivo DEX a 65,536 (64K), incluidos los métodos de marco de Android, los métodos de biblioteca y los métodos en su propio código.

Para superar este límite es necesario configurar el proceso de construcción de su aplicación para generar más de un archivo DEX, conocido como Multidex.

Observaciones

¿Qué es dex?

Dex es el nombre del formato de archivo y la codificación en la que se compila el código Java de Android. Las primeras versiones de Android cargarían y ejecutarían binarios de `dex` directamente en una máquina virtual llamada Dalvik. Las versiones más recientes de Android utilizan el Android Runtime (ART), que trata los archivos `dex` como una representación intermedia y realiza compilaciones adicionales antes de ejecutar la aplicación.

Dex es un formato de archivo muy antiguo, en términos de la vida útil de los teléfonos inteligentes, y fue diseñado para dispositivos cuya memoria principal se midió en decenas de megabytes. Las limitaciones de diseño de esos días han permanecido con nosotros hasta el día de hoy.

El problema:

El formato de archivo `dex` codifica un límite para la cantidad de métodos a los que se puede hacer referencia en un solo binario. Debido a que la parte del formato de archivo que almacena el número de referencias tiene una longitud de dos bytes, el número máximo de referencias de método es `0xFFFF` , o 65535. Si una aplicación contiene más de esa cantidad de referencias de método, no podrá compilarse.

Qué hacer al respecto:

Google ha proporcionado una solución a este problema, llamado Multidex. Tiene componentes en tiempo de compilación y en tiempo de ejecución. Como su nombre lo indica, en tiempo de compilación dividirá el código entre uno o más archivos `dex` . En tiempo de ejecución, le enseñará al `ClassLoader` predeterminado cómo buscar clases de estos archivos.

Este enfoque funciona bien en dispositivos más nuevos, pero tiene algunos inconvenientes importantes. Puede aumentar dramáticamente el tiempo de inicio de la aplicación, y en dispositivos más antiguos puede causar fallas en la `Application Not Responding`.

Multidex, si bien es efectivo, debe evitarse si es posible.

Cómo evitar el límite:

Antes de configurar su aplicación para permitir el uso de 64 K o más referencias de métodos, debe tomar medidas para reducir el número total de referencias a las que llama el código de su aplicación, incluidos los métodos definidos por su código de aplicación o las bibliotecas incluidas. Las siguientes estrategias pueden ayudarlo a evitar alcanzar el límite de referencia de dex:

- **Revise las dependencias directas y transitivas de su aplicación** : asegúrese de que cualquier dependencia de biblioteca grande que incluya en su aplicación se use de una manera que supere la cantidad de código que se agrega a la aplicación. Un anti-patrón común es incluir una biblioteca muy grande porque algunos métodos de utilidad fueron útiles. Reducir las dependencias del código de la aplicación a menudo puede ayudarlo a evitar el límite de referencia de dex.
- **Elimine el código no utilizado con ProGuard** : configure los ajustes de ProGuard para que su aplicación ejecute ProGuard y asegúrese de que haya activado el encogimiento para las versiones de lanzamiento. La habilitación de la reducción asegura que no esté enviando código no utilizado con sus APK.

El primer punto requiere diligencia y disciplina por parte del desarrollador. Al incorporar bibliotecas de terceros, se debe considerar el tamaño de la biblioteca. Por ejemplo, dos bibliotecas JSON populares son Jackson y Gson. Funcionalmente son bastante similares, pero Gson tiende a ver un mayor uso en Android. Una razón es que Jackson pesa alrededor de 9,000 métodos, mientras que Gson contribuye con 1,900.

Hay varias herramientas disponibles para ayudar a los desarrolladores a realizar un seguimiento del tamaño de su aplicación:

- dexcount-gradle-plugin informa el número de referencias de métodos en su APK o AAR en cada compilación
- dex-method-count es una herramienta de línea de comandos que cuenta el número de referencias de métodos en un APK
- www.methodscount.com es un servicio web que contará las referencias de los métodos en cualquier APK que cargue.

Examples

Multidex utilizando MultiDexApplication directamente

Utilice esta opción si no necesita una subclase de `Application`.

Esta es la opción más simple, pero de esta manera no puede proporcionar su propia subclase de

`Application` . Si se necesita una subclase de `Application` , tendrá que cambiar a una de las otras opciones para hacerlo.

Para esta opción, simplemente especifique el nombre de clase completamente calificado `android.support.multidex.MultiDexApplication` para la propiedad `android:name` de la etiqueta de la `application` en el archivo AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.multidex.myapplication">
    <application
        ...
        android:name="android.support.multidex.MultiDexApplication">
        ...
    </application>
</manifest>
```

Multidex extendiendo la aplicación

Utilice esta opción si su proyecto requiere una subclase de `Application` .

Especifique esta subclase de `Application` utilizando la propiedad `android:name` en el archivo de manifiesto dentro de la etiqueta de la `application` .

En la subclase de la `Application` , agregue la `attachBaseContext()` método `attachBaseContext()` , y en ese método llame a `MultiDex.install()` :

```
package com.example;

import android.app.Application;
import android.content.Context;

/**
 * Extended application that support multidex
 */
public class MyApplication extends Application {

    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        MultiDex.install(this);
    }
}
```

Asegúrese de que la subclase de la `Application` esté especificada en la etiqueta de la `application` de su AndroidManifest.xml:

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

Habilitando Multidex

Para habilitar una configuración multidex necesitas:

- para cambiar su configuración de construcción Gradle
- para usar una aplicación `MultiDexApplication` o habilitar MultiDex en su clase de `Application`

Configuracion gradle

En `app/build.gradle` agregue estas partes:

```
android {
    compileSdkVersion 24
    buildToolsVersion "24.0.1"

    defaultConfig {
        ...
        minSdkVersion 14
        targetSdkVersion 24
        ...

        // Enabling multidex support.
        multiDexEnabled true
    }
    ...
}

dependencies {
  compile 'com.android.support:multidex:1.0.1'
}
```

Habilite MultiDex en su aplicación

Luego proceda con una de las tres opciones:

- Multidex extendiendo la aplicación
- Multidex extendiendo MultiDexApplication
- Multidex utilizando MultiDexApplication directamente

Cuando estos ajustes de configuración se agregan a una aplicación, las herramientas de compilación de Android construyen un dex primario (classes.dex) y el soporte (classes2.dex, classes3.dex) según sea necesario.
El sistema de compilación luego los empaquetará en un archivo APK para su distribución.

Referencias del método de conteo en cada compilación (Dexcount Gradle Plugin)

El complemento de dexcount cuenta los métodos y el recuento de recursos de clase después de una compilación exitosa.

Agregue el complemento en la `app/build.gradle` :

```
apply plugin: 'com.android.application'

buildscript {
    repositories {
        mavenCentral() // or jcenter()
    }

    dependencies {
        classpath 'com.getkeepsafe.dexcount:dexcount-gradle-plugin:0.5.5'
    }
}
```

Aplique el complemento en el archivo `app/build.gradle` :

```
apply plugin: 'com.getkeepsafe.dexcount'
```

Busque los datos de salida generados por el complemento en:

../app/build/outputs/dexcount

Especialmente útil es el gráfico .html en:

../app/build/outputs/dexcount/debugChart/index.html

Multidex extendiendo MultiDexApplication

Esto es muy similar a usar una subclase de `Application` y anular el método `attachBaseContext()` .

Sin embargo, al usar este método, no es necesario que sustituya a `attachBaseContext()` ya que esto ya se hizo en la superclase `MultiDexApplication` .

Extienda la aplicación `MultiDexApplication` lugar de la `Application` :

```
package com.example;

import android.support.multidex.MultiDexApplication;
import android.content.Context;

/**
 * Extended MultiDexApplication
 */
public class MyApplication extends MultiDexApplication {

     // No need to override attachBaseContext()

     //.........
}
```

Agrega esta clase a tu AndroidManifest.xml exactamente como si estuvieras extendiendo la aplicación:

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

Lea Multidex y el método Dex Limit en línea: https://riptutorial.com/es/android/topic/1887/multidex-y-el-metodo-dex-limit

Capítulo 169: MVVM (Arquitectura)

Observaciones

Sintaxis con el enlace de datos

Cuando se vincula una función viewModel a una propiedad en xml, se eliminan ciertos prefijos de función como `get` o `is` . P.ej. `ViewModel::getFormattedText` en el ViewModel se convertirá en `@{viewModel.formattedText}` al enlazarlo a una propiedad en xml. De forma similar con `ViewModel::isContentVisible` -> `@{viewModel.contentVisible}` (notación Java Bean)

Las clases de enlace generadas como `ActivityMainBinding` se nombran después del xml para el que están creando enlaces, no la clase java.

Encuadernaciones personalizadas

En el activity_main.xml establecí el atributo textColor en la `app` y no el espacio de nombres de `android` . ¿Porqué es eso? Debido a que hay un `textColor` personalizado definido para el atributo `textColor` que resuelve un ID de recurso ColorRes enviado por ViewModel a un color real.

```
public class CustomBindings {

    @TargetApi(23)
    @BindingAdapter({"bind:textColor"})
    public static void setTextColor(TextView textView, int colorResId) {
        final Context context = textView.getContext();
        final Resources resources = context.getResources();
        final int apiVersion = Build.VERSION.SDK_INT;
        int color;

        if (apiVersion >= Build.VERSION_CODES.M) {
           color = resources.getColor(colorResId, context.getTheme());
        } else {
            color = resources.getColor(colorResId);
        }

        textView.setTextColor(color);
    }

}
```

Para obtener detalles sobre cómo funciona esto, consulte la Biblioteca de DataBinding: Configuradores personalizados

Espera ... ¿hay lógica en tu xml?

Podría argumentar que las cosas que hago en xml para `android:visibility` y `app:textColor` son incorrectas / anti-patrones en el contexto MVVM porque hay una lógica de vista en mi opinión. Sin embargo, diría que es más importante para mí mantener las dependencias de Android fuera de mi ViewModel por razones de prueba.

Además, ¿qué hace realmente la `app:textColor` ? Solo resuelve un puntero de recurso al color real asociado con él. Así que el ViewModel aún decide qué color se muestra en función de alguna condición.

En cuanto a `android:visibility` que siento por el nombre del método, en realidad está bien usar el operador ternario aquí. Debido al nombre `isLoadingVisible` y `isContentVisible` realmente no hay duda acerca de a qué se debe resolver cada resultado en la vista. Así que siento que es más bien ejecutar un comando dado por ViewModel en lugar de hacer realmente la lógica de visualización.

Por otro lado, estoy de acuerdo en que usar `viewModel.isLoading ? View.VISIBLE : View.GONE` sería algo malo porque está haciendo suposiciones en la vista qué significa ese estado para la vista.

Material util

Los siguientes recursos me han ayudado mucho para tratar de entender este concepto:

- Jeremy Likness - Explicación (C #) (08.2010) de Model-View-ViewModel (MVVM)
- Shamlia Shukkur - Comprensión de los conceptos básicos del patrón de diseño de MVVM (C #) (03.2013)
- Frode Nilsen - Encuadernación de datos de Android: ¡Adiós presentador, hola ViewModel! (07.2015)
- Joe Birch - Acercándose a Android con MVVM (09.2015)
- Florina Muntenescu - Patrones de arquitectura de Android, parte 3: Modelo-Vista-Modelo de vista (10.2016)

Examples

Ejemplo de MVVM usando la biblioteca de DataBinding

El objetivo principal de MVVM es separar las capas que contienen lógica de la capa de vista.

En Android, podemos usar la biblioteca de DataBinding para ayudarnos con esto y hacer que la mayoría de nuestras unidades lógicas puedan probarse sin preocuparse por las dependencias de Android.

En este ejemplo, mostraré los componentes centrales para una aplicación simple y estúpida que hace lo siguiente:

- En el inicio simula una llamada de red y muestra un spinner de carga
- Muestre una vista con un contador de clic TextView, un mensaje TextView y un botón para incrementar el contador
- Al hacer clic en el botón actualizar contador y actualizar el color y el mensaje de texto si el contador alcanza algún número

Vamos a empezar con la capa de vista:

`activity_main.xml` :

Si no está familiarizado con el funcionamiento de DataBinding, probablemente debería tomar 10

minutos para familiarizarse con él. Como puede ver, todos los campos que normalmente actualizaría con los configuradores están vinculados a funciones en la variable viewModel.

Si tiene una pregunta sobre las propiedades de `android:visibility` o `app:textColor` , consulte la sección 'Comentarios'.

```
<layout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools">

   <data>

       <import type="android.view.View" />

       <variable
           name="viewModel"
           type="de.walled.mvvmtest.viewmodel.ClickerViewModel"/>
   </data>

   <RelativeLayout
       android:id="@+id/activity_main"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:padding="@dimen/activity_horizontal_margin"

       tools:context="de.walled.mvvmtest.view.MainActivity">

       <LinearLayout
           android:id="@+id/click_counter"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_centerHorizontal="true"
           android:layout_marginTop="60dp"
           android:visibility="@{viewModel.contentVisible ? View.VISIBLE : View.GONE}"

           android:padding="8dp"

           android:orientation="horizontal">

           <TextView
               android:id="@+id/number_of_clicks"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               style="@style/ClickCounter"

               android:text="@{viewModel.numberOfClicks}"
               android:textAlignment="center"
               app:textColor="@{viewModel.counterColor}"

               tools:text="8"
               tools:textColor="@color/red"
           />

           <TextView
               android:id="@+id/static_label"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:layout_marginLeft="4dp"
               android:layout_marginStart="4dp"
```

```
                style="@style/ClickCounter"

                android:text="@string/label.clicks"
                app:textColor="@{viewModel.counterColor}"
                android:textAlignment="center"

                tools:textColor="@color/red"
            />
        </LinearLayout>


        <TextView
            android:id="@+id/message"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/click_counter"
            android:layout_centerHorizontal="true"
            android:visibility="@{viewModel.contentVisible ? View.VISIBLE : View.GONE}"

            android:text="@{viewModel.labelText}"
            android:textAlignment="center"
            android:textSize="18sp"

            tools:text="You're bad and you should feel bad!"
        />

        <Button
            android:id="@+id/clicker"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/message"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="8dp"
            android:visibility="@{viewModel.contentVisible ? View.VISIBLE : View.GONE}"

            android:padding="8dp"

            android:text="@string/label.button"

            android:onClick="@{() -> viewModel.onClickIncrement()}"
        />

        <android.support.v4.widget.ContentLoadingProgressBar
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="90dp"
            android:layout_centerHorizontal="true"
            style="@android:style/Widget.ProgressBar.Inverse"
            android:visibility="@{viewModel.loadingVisible ? View.VISIBLE : View.GONE}"

            android:indeterminate="true"
        />

    </RelativeLayout>

</layout>
```

A continuación la capa modelo. Aquí tengo:

- Dos campos que representan el estado de la aplicación.

- Captadores de leer el número de clics y el estado de emoción
- un método para incrementar mi cuenta de clics
- un método para restaurar un estado anterior (importante para cambios de orientación)

También defino aquí un 'estado de emoción' que depende del número de clics. Esto se usará más adelante para actualizar el color y el mensaje en la Vista.

Es importante tener en cuenta que no hay suposiciones en el modelo sobre cómo se puede mostrar el estado al usuario.

`ClickerModel.java`

```
import com.google.common.base.Optional;

import de.walled.mvvmtest.viewmodel.ViewState;

public class ClickerModel implements IClickerModel {

    private int numberOfClicks;
    private Excitement stateOfExcitement;

    public void incrementClicks() {
        numberOfClicks += 1;
        updateStateOfExcitement();
    }

    public int getNumberOfClicks() {
        return Optional.fromNullable(numberOfClicks).or(0);
    }

    public Excitement getStateOfExcitement() {
        return Optional.fromNullable(stateOfExcitement).or(Excitement.BOO);
    }

    public void restoreState(ViewState state) {
        numberOfClicks = state.getNumberOfClicks();
        updateStateOfExcitement();
    }

    private void updateStateOfExcitement() {
        if (numberOfClicks < 10) {
            stateOfExcitement = Excitement.BOO;
        } else if (numberOfClicks <= 20) {
            stateOfExcitement = Excitement.MEH;
        } else {
            stateOfExcitement = Excitement.WOOHOO;
        }
    }
}
```

Siguiente el ViewModel.

Esto activará cambios en el modelo y formateará los datos del modelo para mostrarlos en la vista. Tenga en cuenta que es aquí donde evaluamos qué representación de GUI es apropiada para el estado dado por el modelo (`resolveCounterColor` y `resolveLabelText`). Entonces, por ejemplo, podríamos implementar fácilmente un `UnderachieverClickerModel` que tiene umbrales más bajos para el estado de emoción sin tocar ningún código en viewModel o view.

También tenga en cuenta que ViewModel no contiene ninguna referencia para ver objetos. Todas las propiedades están vinculadas a través de las anotaciones de `@Bindable` y se actualizan cuando `notifyChange()` (señala que todas las propiedades deben actualizarse) o `notifyPropertyChanged(BR.propertyName)` (indica que es necesario actualizar estas propiedades).

`ClickerViewModel.java`

```
import android.databinding.BaseObservable;

import android.databinding.Bindable;
import android.support.annotation.ColorRes;
import android.support.annotation.StringRes;

import com.android.databinding.library.baseAdapters.BR;

import de.walled.mvvmtest.R;
import de.walled.mvvmtest.api.IClickerApi;
import de.walled.mvvmtest.model.Excitement;
import de.walled.mvvmtest.model.IClickerModel;
import rx.Observable;

public class ClickerViewModel extends BaseObservable {

    private final IClickerApi api;
    boolean isLoading = false;
    private IClickerModel model;

    public ClickerViewModel(IClickerModel model, IClickerApi api) {
        this.model = model;
        this.api = api;
    }

    public void onClickIncrement() {
        model.incrementClicks();
        notifyChange();
    }

    public ViewState getViewState() {
        ViewState viewState = new ViewState();
        viewState.setNumberOfClicks(model.getNumberOfClicks());
        return viewState;
    }

    public Observable<ViewState> loadData() {
        isLoading = true;
        return api.fetchInitialState()
                .doOnNext(this::initModel)
                .doOnTerminate(() -> {
                    isLoading = false;
                    notifyPropertyChanged(BR.loadingVisible);
                    notifyPropertyChanged(BR.contentVisible);
                });
    }

    public void initFromSavedState(ViewState savedState) {
        initModel(savedState);
    }

    @Bindable
    public String getNumberOfClicks() {
```

```
        final int clicks = model.getNumberOfClicks();
        return String.valueOf(clicks);
    }

    @Bindable
    @StringRes
    public int getLabelText() {
        final Excitement stateOfExcitement = model.getStateOfExcitement();
        return resolveLabelText(stateOfExcitement);
    }

    @Bindable
    @ColorRes
    public int getCounterColor() {
        final Excitement stateOfExcitement = model.getStateOfExcitement();
        return resolveCounterColor(stateOfExcitement);
    }

    @Bindable
    public boolean isLoadingVisible() {
        return isLoading;
    }

    @Bindable
    public boolean isContentVisible() {
        return !isLoading;
    }

    private void initModel(final ViewState viewState) {
        model.restoreState(viewState);
        notifyChange();
    }

    @ColorRes
    private int resolveCounterColor(Excitement stateOfExcitement) {
        switch (stateOfExcitement) {
            case MEH:
                return R.color.yellow;
            case WOOHOO:
                return R.color.green;
            default:
                return R.color.red;
        }
    }

    @StringRes
    private int resolveLabelText(Excitement stateOfExcitement) {
        switch (stateOfExcitement) {
            case MEH:
                return R.string.label_indifferent;
            case WOOHOO:
                return R.string.label_excited;
            default:
                return R.string.label_negative;
        }
    }

}
```

¡Atando todo junto en la Actividad!

Aquí vemos la vista que inicializa viewModel con todas las dependencias que pueda necesitar, que deben ser instanciadas desde un contexto de Android.

Una vez que se inicializa viewModel, se vincula al diseño xml a través de DataBindingUtil (consulte la sección 'Sintaxis' para nombrar las clases generadas).

Las suscripciones a las notas están suscritas en esta capa porque tenemos que gestionar la cancelación de la suscripción cuando la actividad se detiene o se destruye para evitar pérdidas de memoria y NPE. También se activa aquí la persistencia y la recarga de viewState en OrientationChanges

`MainActivity.java`

```
import android.databinding.DataBindingUtil;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

import de.walled.mvvmtest.R;
import de.walled.mvvmtest.api.ClickerApi;
import de.walled.mvvmtest.api.IClickerApi;
import de.walled.mvvmtest.databinding.ActivityMainBinding;
import de.walled.mvvmtest.model.ClickerModel;
import de.walled.mvvmtest.viewmodel.ClickerViewModel;
import de.walled.mvvmtest.viewmodel.ViewState;
import rx.Subscription;
import rx.subscriptions.Subscriptions;

public class MainActivity extends AppCompatActivity {

    private static final String KEY_VIEW_STATE = "state.view";

    private ClickerViewModel viewModel;
    private Subscription fakeLoader = Subscriptions.unsubscribed();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // would usually be injected but I feel Dagger would be out of scope
        final IClickerApi api = new ClickerApi();
        setupViewModel(savedInstanceState, api);

        ActivityMainBinding binding = DataBindingUtil.setContentView(this,
R.layout.activity_main);
        binding.setViewModel(viewModel);
    }

    @Override
    protected void onPause() {
        fakeLoader.unsubscribe();
        super.onPause();
    }

    @Override
    protected void onDestroy() {
        fakeLoader.unsubscribe();
        super.onDestroy();
    }
```

```
    @Override
    protected void onSaveInstanceState(Bundle outState) {
        outState.putSerializable(KEY_VIEW_STATE, viewModel.getViewState());
    }

    private void setupViewModel(Bundle savedInstance, IClickerApi api) {
        viewModel = new ClickerViewModel(new ClickerModel(), api);
        final ViewState savedState = getViewStateFromBundle(savedInstance);

        if (savedState == null) {
            fakeLoader = viewModel.loadData().subscribe();
        } else {
            viewModel.initFromSavedState(savedState);
        }
    }

    private ViewState getViewStateFromBundle(Bundle savedInstance) {
        if (savedInstance != null) {
            return (ViewState) savedInstance.getSerializable(KEY_VIEW_STATE);
        }
        return null;
    }
}
```

Para ver todo en acción mira este proyecto de ejemplo .

Lea MVVM (Arquitectura) en línea: https://riptutorial.com/es/android/topic/7549/mvvm--arquitectura-

Capítulo 170: NavigationView

Observaciones

El **NavigationView** representa un menú de navegación estándar para la aplicación. Los contenidos del menú se pueden rellenar con un archivo de recursos de menú.

Antes de usar `NavigationView` , debe agregar la dependencia de la biblioteca de soporte de diseño en el archivo build.gradle:

```
 dependencies {
    compile 'com.android.support:design:24.2.0'
}
```

Documentación oficial:

https://developer.android.com/reference/android/support/design/widget/NavigationView.html

Especificaciones de materiales de diseño:

https://material.google.com/patterns/navigation-drawer.html#navigation-drawer-content

Examples

Cómo agregar el NavigationView

Para usar un NavigationView, simplemente agregue la dependencia en el archivo `build.gradle` como se describe en la sección de comentarios

A continuación, agregue el `NavigationView` en el diseño

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
```

```
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>
```

res/layout/nav_header_main.xml : la vista que se mostrará en la parte superior del cajón

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="@dimen/nav_header_height"
    android:background="@drawable/side_nav_bar"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:theme="@style/ThemeOverlay.AppCompat.Dark"
    android:orientation="vertical"
    android:gravity="bottom">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="@dimen/nav_header_vertical_spacing"
        android:src="@android:drawable/sym_def_app_icon"
        android:id="@+id/imageView" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="@dimen/nav_header_vertical_spacing"
        android:text="Android Studio"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="android.studio@android.com"
        android:id="@+id/textView" />

</LinearLayout>
```

res/layout/app_bar_main.xml Una capa de abstracción para que la barra de herramientas la separe del contenido:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="eu.rekisoft.playground.MainActivity">
```

```
    <android.support.design.widget.AppBarLayout
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main"/>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

res/layout/content_main.xml El contenido real de la actividad solo para demostración, aquí debe colocar su xml de diseño normal:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/app_bar_main"
    tools:context="eu.rekisoft.playground.MainActivity">

    <TextView
        android:text="Hello World!"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>
```

Defina su archivo de menú como *res/menu/activity_main_drawer.xml* :

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <group android:checkableBehavior="single">
        <item
```

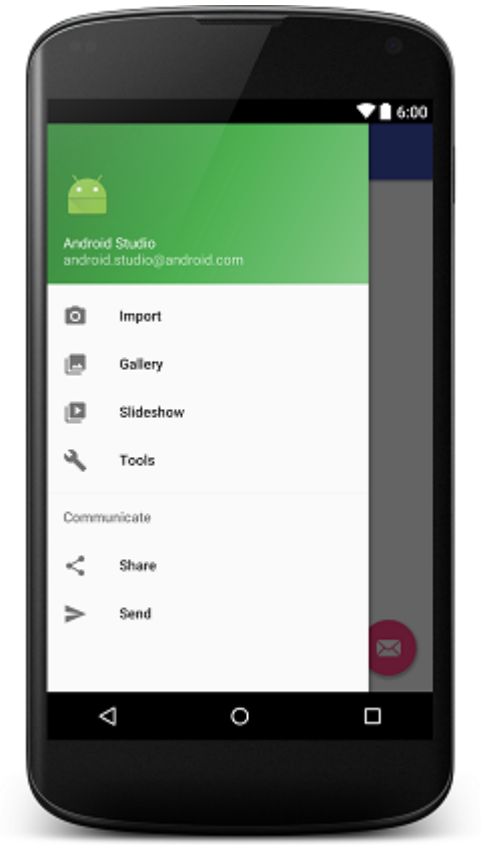
```
            android:id="@+id/nav_camera"
            android:icon="@drawable/ic_menu_camera"
            android:title="Import" />
        <item
            android:id="@+id/nav_gallery"
            android:icon="@drawable/ic_menu_gallery"
            android:title="Gallery" />
        <item
            android:id="@+id/nav_slideshow"
            android:icon="@drawable/ic_menu_slideshow"
            android:title="Slideshow" />
        <item
            android:id="@+id/nav_manage"
            android:icon="@drawable/ic_menu_manage"
            android:title="Tools" />
    </group>

    <item android:title="Communicate">
        <menu>
            <item
                android:id="@+id/nav_share"
                android:icon="@drawable/ic_menu_share"
                android:title="Share" />
            <item
                android:id="@+id/nav_send"
                android:icon="@drawable/ic_menu_send"
                android:title="Send" />
        </menu>
    </item>

</menu>
```

Y finalmente el `java/main/eu/rekisoft/playground/MainActivity.java` :

```
public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.setDrawerListener(toggle);
        toggle.syncState();
```

```
        NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);
    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
        switch(item.getItemId()) {/*...*/}

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
}
```

Se verá así:

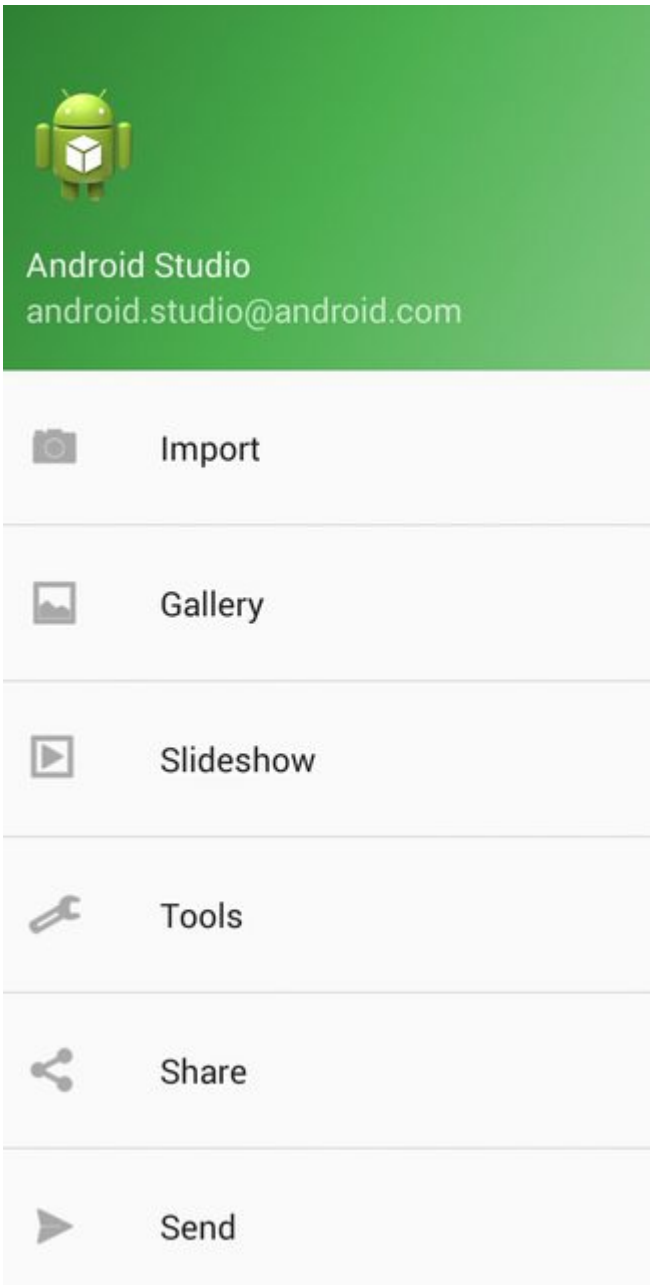


Añadir subrayado en los elementos del menú

Cada grupo termina con un separador de línea. Si cada elemento de su menú tiene su propio grupo, logrará la salida gráfica deseada. Solo funcionará si sus diferentes grupos tienen diferentes `android:id` . Además, en `menu.xml` recuerde mencionar `android:checkable="true"` para un solo elemento y `android:checkableBehavior="single"` para un grupo de elementos.

```
<?xml version="1.0" encoding="utf-8"?>
    <menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
          android:id="@+id/pos_item_help"
          android:checkable="true"
          android:title="Help" />
     <item
        android:id="@+id/pos_item_pos"
        android:checkable="true"
        android:title="POS" />

    <item
        android:id="@+id/pos_item_orders"
        android:checkable="true"
        android:title="Orders" />

       <group
            android:id="@+id/group"
            android:checkableBehavior="single">

                <item
                android:id="@+id/menu_nav_home"
                android:icon="@drawable/ic_home_black_24dp"
                android:title="@string/menu_nav_home" />
```

```
        </group>

    ......
    </menu>
```



Añadir separadores al menú

Acceda a RecyclerView en NavigationView y agregue ItemDecoration a él.

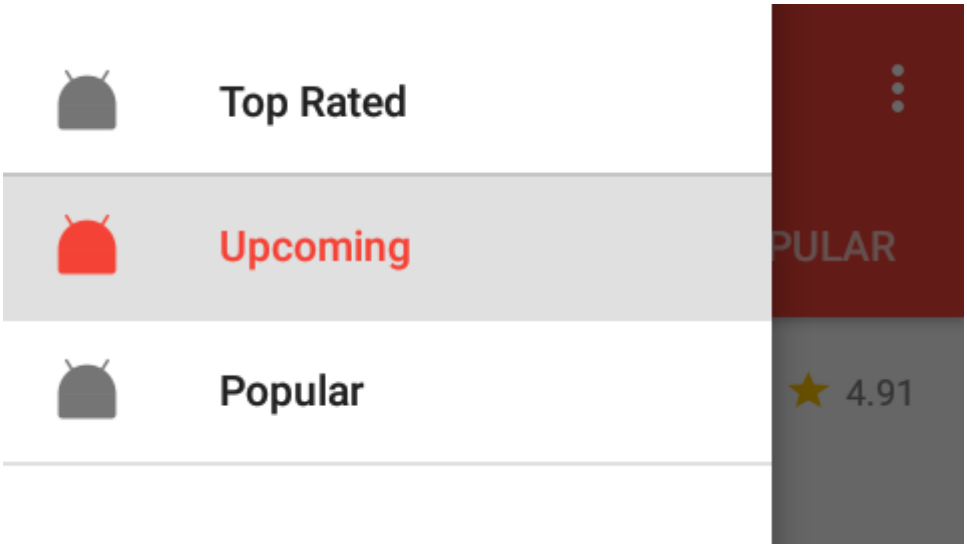
```
NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new DividerItemDecoration(this));
```

Código para DividerItemDecoration

```
public class DividerItemDecoration extends RecyclerView.ItemDecoration {
```

```
    private static final int[] ATTRS = new int[]{android.R.attr.listDivider};

    private Drawable mDivider;

    public DividerItemDecoration(Context context) {
        final TypedArray styledAttributes = context.obtainStyledAttributes(ATTRS);
        mDivider = styledAttributes.getDrawable(0);
        styledAttributes.recycle();
    }


    @Override
    public void onDraw(Canvas c, RecyclerView parent, RecyclerView.State state) {
        int left = parent.getPaddingLeft();
        int right = parent.getWidth() - parent.getPaddingRight();

        int childCount = parent.getChildCount();
        for (int i = 1; i < childCount; i++) {
            View child = parent.getChildAt(i);

            RecyclerView.LayoutParams params = (RecyclerView.LayoutParams)
child.getLayoutParams();

            int top = child.getBottom() + params.bottomMargin;
            int bottom = top + mDivider.getIntrinsicHeight();

            mDivider.setBounds(left, top, right, bottom);
            mDivider.draw(c);
        }
    }
}
```

Avance:



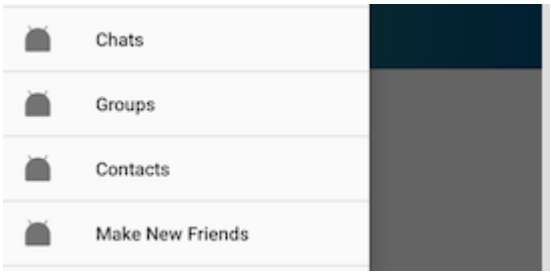
Agregue el menú Divider usando la opción predeterminada DividerItemDecoration.

Solo usa la clase de DividerItemDecoration por defecto:

```
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation);
```

```
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new
DividerItemDecoration(context,DividerItemDecoration.VERTICAL));
```

Vista previa:



Lea NavigationView en línea: https://riptutorial.com/es/android/topic/97/navigationview

Capítulo 171: Notificacion canal android o

Introducción

Los canales de notificación nos permiten a los desarrolladores de aplicaciones agrupar nuestras notificaciones en grupos (canales), y el usuario tiene la capacidad de modificar la configuración de notificación para todo el canal a la vez. En Android O se presenta esta función. Ahora mismo está disponible la vista previa para desarrolladores.

Sintaxis

1. clase NotificationUtils {} // para crear un canal de notificación
2. createChannel () // Método genérico para crear notificaciones

Parámetros

Método	Descripción
IMPORTANCE_MAX	no usado
IMPORTANCIA: ALTA	Se muestra por todas partes, hace ruido y asoma.
IMPORTANCE_DEFAULT	Se muestra en todas partes, hace ruido, pero no interfiere visualmente.
IMPORTANCE_LOW	Muestra en todas partes, pero no es intrusivo.
IMPORTANCE_MIN	Solo se muestra en la sombra, debajo del pliegue.
IMPORTANCE_NONE	una notificación sin importancia; no se muestra en la sombra

Examples

Canal de notificaciones

¿Qué son los canales de notificación?

Los canales de notificación nos permiten a los desarrolladores de aplicaciones agrupar nuestras notificaciones en grupos, canales, y el usuario tiene la capacidad de modificar la configuración de notificación para todo el canal a la vez. Por ejemplo, para cada canal, los usuarios pueden bloquear completamente todas las notificaciones, anular el nivel de importancia o permitir que se muestre una credencial de notificación. Esta nueva característica ayuda a mejorar en gran medida la experiencia del usuario de una aplicación.

Crear canales de notificación

```
 import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Context;
import android.content.ContextWrapper;
import android.graphics.Color;

public class NotificationUtils extends ContextWrapper {


private NotificationManager mManager;
public static final String ANDROID_CHANNEL_ID = "com.sai.ANDROID";
public static final String IOS_CHANNEL_ID = "com.sai.IOS";
public static final String ANDROID_CHANNEL_NAME = "ANDROID CHANNEL";
public static final String IOS_CHANNEL_NAME = "IOS CHANNEL";

public NotificationUtils(Context base) {
    super(base);
    createChannels();
}

public void createChannels() {

    // create android channel
    NotificationChannel androidChannel = new NotificationChannel(ANDROID_CHANNEL_ID,
            ANDROID_CHANNEL_NAME, NotificationManager.IMPORTANCE_DEFAULT);
    // Sets whether notifications posted to this channel should display notification lights
    androidChannel.enableLights(true);
    // Sets whether notification posted to this channel should vibrate.
    androidChannel.enableVibration(true);
    // Sets the notification light color for notifications posted to this channel
    androidChannel.setLightColor(Color.BLUE);
    // Sets whether notifications posted to this channel appear on the lockscreen or not
    androidChannel.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);

    getManager().createNotificationChannel(androidChannel);

    // create ios channel
    NotificationChannel iosChannel = new NotificationChannel(IOS_CHANNEL_ID,
            IOS_CHANNEL_NAME, NotificationManager.IMPORTANCE_HIGH);
    iosChannel.enableLights(true);
    iosChannel.enableVibration(true);
    iosChannel.setLightColor(Color.GRAY);
    iosChannel.setLockscreenVisibility(Notification.VISIBILITY_PUBLIC);
    getManager().createNotificationChannel(iosChannel);


  }


private NotificationManager getManager() {
    if (mManager == null) {
        mManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    }
    return mManager;
}}
```

En el código anterior, creamos dos instancias de NotificationChannel, pasando un nombre de

canal a un nombre único y también un nivel de importancia en su constructor. Para cada canal de notificación, aplicamos las siguientes características.

1. Sonar
2. Luces
3. Vibración
4. Notificación para mostrar en pantalla de bloqueo.

Finalmente, obtuvimos el NotificationManager del sistema y luego registramos el canal llamando al método createNotificationChannel (), pasando el canal que hemos creado.

Podemos crear múltiples canales de notificación a la vez con createNotificationChannels (), pasando una lista de Java de instancias de NotificationChannel. Puede obtener todos los canales de notificación para una aplicación con getNotificationChannels () y obtener un canal específico con getNotificationChannel (), pasando solo la identificación del canal como argumento.

Nivel de importancia en los canales de notificación.

Método	Descripción
IMPORTANCE_MAX	no usado
IMPORTANCIA: ALTA	Se muestra por todas partes, hace ruido y asoma.
IMPORTANCE_DEFAULT	Se muestra en todas partes, hace ruido, pero no interfiere visualmente.
IMPORTANCE_LOW	Muestra en todas partes, pero no es intrusivo, el valor es 0
IMPORTANCE_MIN	Solo se muestra en la sombra, debajo del pliegue.
IMPORTANCE_NONE	una notificación sin importancia; no se muestra en la sombra

Crear notificación y publicar en el canal

Hemos creado dos notificaciones una usando NotificationUtils otra usando NotificationBuilder.

```
public Notification.Builder getAndroidChannelNotification(String title, String body) {
    return new Notification.Builder(getApplicationContext(), ANDROID_CHANNEL_ID)
            .setContentTitle(title)
            .setContentText(body)
            .setSmallIcon(android.R.drawable.stat_notify_more)
            .setAutoCancel(true);
}

public Notification.Builder getIosChannelNotification(String title, String body) {
    return new Notification.Builder(getApplicationContext(), IOS_CHANNEL_ID)
            .setContentTitle(title)
            .setContentText(body)
            .setSmallIcon(android.R.drawable.stat_notify_more)
            .setAutoCancel(true);
}
```

También podemos configurar NotificationChannel usando Notification.Builder (). Para eso podemos usar ***setChannel (String channelId)*** .

Actualizar la configuración del canal de notificación

Una vez que crea un canal de notificación, el usuario se encarga de su configuración y comportamiento. Puede llamar a createNotificationChannel () nuevamente para cambiar el nombre de un canal de notificación existente o actualizar su descripción. El siguiente código de ejemplo describe cómo puede redirigir a un usuario a la configuración de un canal de notificación al crear una intención de iniciar una actividad. En este caso, la intención requiere datos extendidos, incluido el ID del canal de notificación y el nombre del paquete de su aplicación.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    //...
    Button buttonAndroidNotifSettings = (Button)
findViewById(R.id.btn_android_notif_settings);
    buttonAndroidNotifSettings.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View view) {
            Intent i = new Intent(Settings.ACTION_CHANNEL_NOTIFICATION_SETTINGS);
            i.putExtra(Settings.EXTRA_APP_PACKAGE, getPackageName());
            i.putExtra(Settings.EXTRA_CHANNEL_ID, NotificationUtils.ANDROID_CHANNEL_ID);
            startActivity(i);
        }
    });
}
```

Archivo XML:

```
<!--...-->
<Button
    android:id="@+id/btn_android_notif_settings"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Notification Settings"/>
<!--...-->
```

Eliminando el canal de notificación

Puede eliminar los canales de notificación llamando a deleteNotificationChannel ().

```
NotificationManager mNotificationManager =
        (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
// The id of the channel.
String id = "my_channel_01";
NotificationChannel mChannel = mNotificationManager.getNotificationChannel(id);
mNotificationManager.deleteNotificationChannel(mChannel);
```

Ahora crea MainActivity y xml

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/activity_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:layout_margin="16dp"
        tools:context="com.chikeandroid.tutsplusalerts.MainActivity">

    <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Tuts+ Android Channel"
                android:layout_gravity="center_horizontal"
                android:textAppearance="@style/TextAppearance.AppCompat.Title"/>

        <EditText
                android:id="@+id/et_android_title"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Title"/>

        <EditText
                android:id="@+id/et_android_author"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Author"/>

        <Button
                android:id="@+id/btn_send_android"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Send"/>
    </LinearLayout>

    <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_marginTop="20dp">

        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Tuts+ IOS Channel"
                android:layout_gravity="center_horizontal"
                android:textAppearance="@style/TextAppearance.AppCompat.Title"/>

        <EditText
                android:id="@+id/et_ios_title"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Title"
```

```
                />

        <EditText
                android:id="@+id/et_ios_author"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Author"/>
        <Button
                android:id="@+id/btn_send_ios"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Send"/>
    </LinearLayout>

</LinearLayout>
```

MainActivity.java

Vamos a editar nuestra MainActivity para que podamos obtener el título y el autor de los componentes de EditText y luego enviarlos al canal de Android. Obtenemos Notification.Builder para el canal de Android que creamos en NotificationUtils y luego notificamos a NotificationManager.

importar android.app.Notification; importar android.os.Bundle; importar android.support.v7.app.AppCompatActivity; importar android.text.TextUtils; importar android.view.View; importar android.widget.Button; importar android.widget.EditText;

```
public class MainActivity extends AppCompatActivity {

    private NotificationUtils mNotificationUtils;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mNotificationUtils = new NotificationUtils(this);

        final EditText editTextTitleAndroid = (EditText) findViewById(R.id.et_android_title);
        final EditText editTextAuthorAndroid = (EditText)
findViewById(R.id.et_android_author);
        Button buttonAndroid = (Button) findViewById(R.id.btn_send_android);

        buttonAndroid.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String title = editTextTitleAndroid.getText().toString();
                String author = editTextAuthorAndroid.getText().toString();

                if(!TextUtils.isEmpty(title) && !TextUtils.isEmpty(author)) {
                    Notification.Builder nb = mNotificationUtils.
                            getAndroidChannelNotification(title, "By " + author);

                    mNotificationUtils.getManager().notify(107, nb.build());
                }
            }
        });
    }
```

```
}
```

Lea Notificacion canal android o en línea:
https://riptutorial.com/es/android/topic/10018/notificacion-canal-android-o

Capítulo 172: Notificaciones

Examples

Creando una notificación simple

Este ejemplo muestra cómo crear una notificación simple que inicia una aplicación cuando el usuario hace clic en ella.

Especifique el contenido de la notificación:

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.ic_launcher) // notification icon
        .setContentTitle("Simple notification") // title
        .setContentText("Hello word") // body message
        .setAutoCancel(true); // clear notification when clicked
```

Crea la intención de disparar al hacer clic:

```
Intent intent = new Intent(this, MainActivity.class);
PendingIntent pi = PendingIntent.getActivity(this, 0, intent, Intent.FLAG_ACTIVITY_NEW_TASK);
mBuilder.setContentIntent(pi);
```

Finalmente, construya la notificación y muéstrela.

```
NotificationManager mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(0, mBuilder.build());
```

Heads Up Notification with Ticker para dispositivos más antiguos

Aquí se explica cómo realizar una Notificación de Heads Up para dispositivos con capacidad y utilizar un Ticker para dispositivos más antiguos.

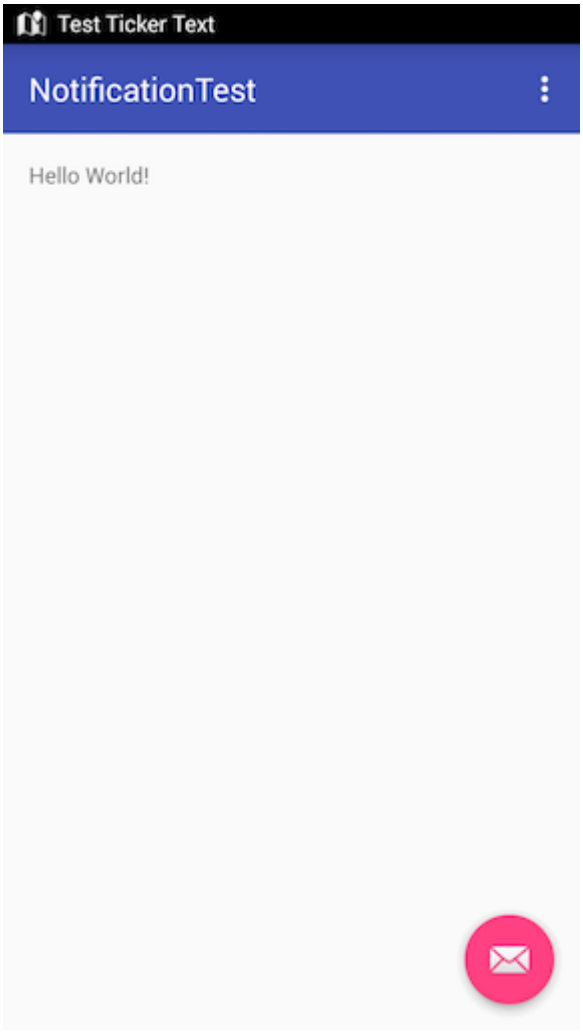
```
// Tapping the Notification will open up MainActivity
Intent i = new Intent(this, MainActivity.class);

// an action to use later
// defined as an app constant:
// public static final String MESSAGE_CONSTANT = "com.example.myapp.notification";
i.setAction(MainActivity.MESSAGE_CONSTANT);
// you can use extras as well
i.putExtra("some_extra", "testValue");

i.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT | Intent.FLAG_ACTIVITY_SINGLE_TOP);
PendingIntent notificationIntent = PendingIntent.getActivity(this, 999, i,
PendingIntent.FLAG_UPDATE_CURRENT);
```

```
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this.getApplicationContext());
builder.setContentIntent(notificationIntent);
builder.setAutoCancel(true);
builder.setLargeIcon(BitmapFactory.decodeResource(this.getResources(),
android.R.drawable.ic_menu_view));
builder.setSmallIcon(android.R.drawable.ic_dialog_map);
builder.setContentText("Test Message Text");
builder.setTicker("Test Ticker Text");
builder.setContentTitle("Test Message Title");

// set high priority for Heads Up Notification
builder.setPriority(NotificationCompat.PRIORITY_HIGH);
builder.setVisibility(NotificationCompat.VISIBILITY_PUBLIC);

// It won't show "Heads Up" unless it plays a sound
if (Build.VERSION.SDK_INT >= 21) builder.setVibrate(new long[0]);

NotificationManager mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(999, builder.build());
```

Esto es lo que parece en Android Marshmallow con la Notificación de Heads Up:

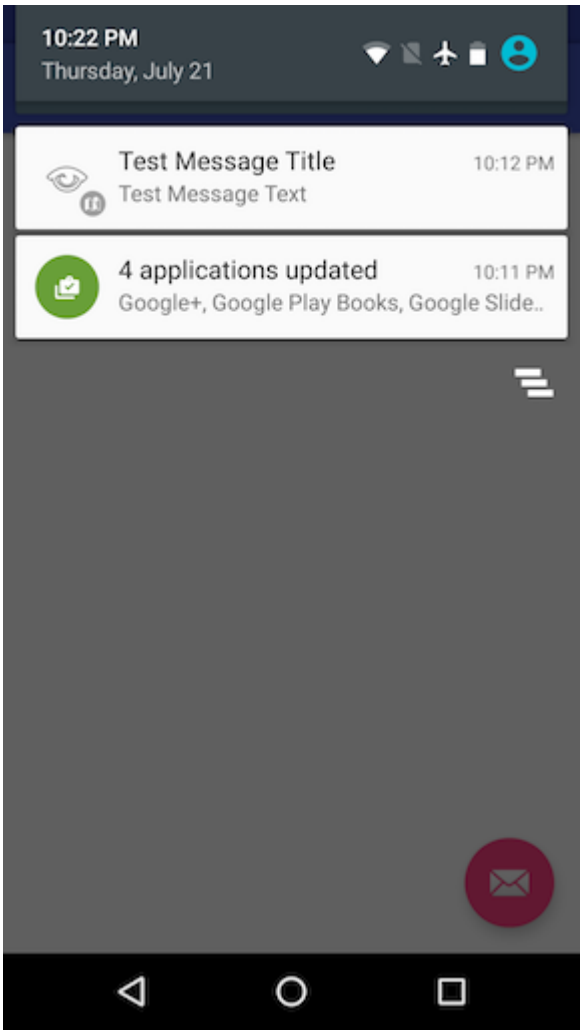


Aquí está lo que parece en Android KitKat con el Ticker:

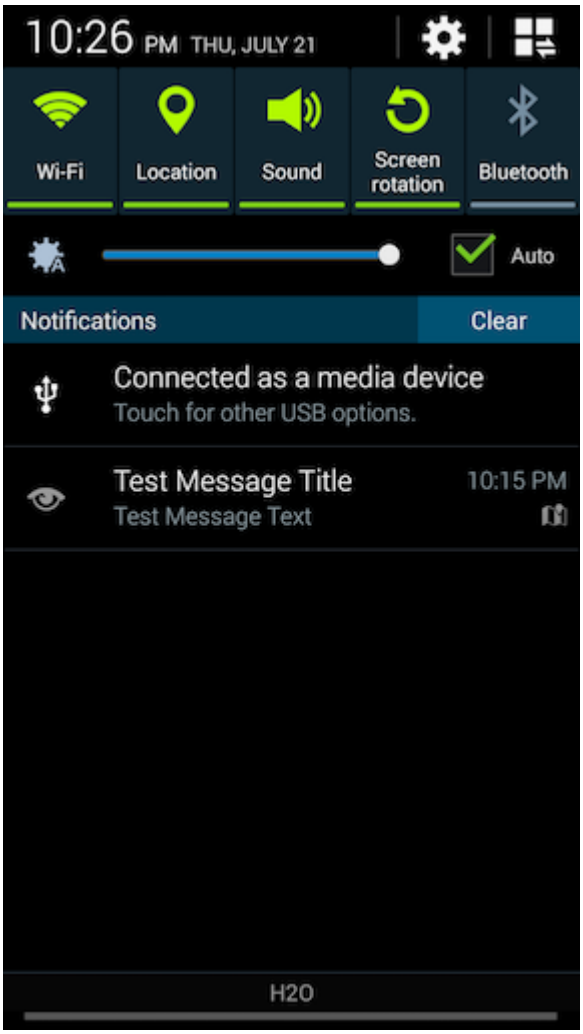


En todas las versiones de Android, la `Notification` se muestra en el cajón de notificaciones.

Android 6.0 Marshmallow:



Android 4.4.x KitKat:



Establecer diferentes prioridades en la notificación

```
NotificationCompat.Builder mBuilder =

        (NotificationCompat.Builder) new NotificationCompat.Builder(context)

        .setSmallIcon(R.drawable.some_small_icon)
        .setContentTitle("Title")
        .setContentText("This is a test notification with MAX priority")
        .setPriority(Notification.PRIORITY_MAX);
```

Cuando la notificación contiene imagen y desea expandir automáticamente la imagen cuando se recibe la notificación, use "PRIORITY_MAX", puede usar otros niveles de prioridad según los requisitos

Información de diferentes niveles de prioridad:

PRIORITY_MAX : se usa para notificaciones críticas y urgentes que alertan al usuario sobre una condición que es crítica en el tiempo o que debe resolverse antes de que puedan continuar con una tarea en particular.

PRIORITY_HIGH : se usa principalmente para comunicaciones importantes, como mensajes o eventos de chat con contenido que es particularmente interesante para el usuario. Las notificaciones de alta prioridad activan la visualización de la notificación de heads-up.

PRIORITY_DEFAULT : se usa para todas las notificaciones que no corresponden a ninguna de las otras prioridades descritas aquí.

PRIORITY_LOW : se utiliza para las notificaciones sobre las que desea que se informe al usuario, pero que sean menos urgentes. Las notificaciones de baja prioridad tienden a aparecer en la parte inferior de la lista, lo que las convierte en una buena opción para cosas como actualizaciones sociales públicas o no dirigidas: el usuario ha solicitado ser notificado de ellas, pero estas notificaciones nunca deben tener prioridad sobre las urgentes o comunicación directa.

PRIORITY_MIN : se usa para obtener información contextual o de fondo, como información meteorológica o información de ubicación contextual. Las notificaciones de prioridad mínima no aparecen en la barra de estado. El usuario los descubre al expandir el tono de notificación.

Referencias: Pautas de diseño de materiales - notificaciones

Programación de notificaciones

A veces es necesario mostrar una notificación en un momento específico, una tarea que desafortunadamente no es trivial en el sistema Android, ya que no existe un método `setTime()` o similar para las notificaciones. Este ejemplo describe los pasos necesarios para programar notificaciones utilizando el `AlarmManager` :

1. **Agregue un `BroadcastReceiver`** que escuche los `Intent` emitidos por el `AlarmManager` Android.

Este es el lugar donde construyes tu notificación en base a los extras provistos con la `Intent` :

```
   public class NotificationReceiver extends BroadcastReceiver {
       @Override
       public void onReceive(Context context, Intent intent) {
           // Build notification based on Intent
           Notification notification = new NotificationCompat.Builder(context)
               .setSmallIcon(R.drawable.ic_notification_small_icon)
               .setContentTitle(intent.getStringExtra("title", ""))
               .setContentText(intent.getStringExtra("text", ""))
               .build();
           // Show notification
           NotificationManager manager = (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
           manager.notify(42, notification);
       }
   }
```

2. **Registre `BroadcastReceiver`** en su archivo `AndroidManifest.xml` (de lo contrario, el receptor no recibirá ninguna `Intent` del `AlarmManager`):

   ```
   <receiver
       android:name=".NotificationReceiver"
       android:enabled="true" />
   ```

3. **Programe una notificación** pasando un `PendingIntent` para su `BroadcastReceiver` con los extras de `Intent` necesarios al `AlarmManager` sistema. Su `BroadcastReceiver` recibirá la `Intent` una vez que haya llegado el momento y mostrará la notificación. El siguiente método

programa una notificación:

```
public static void scheduleNotification(Context context, long time, String title, String
text) {
    Intent intent = new Intent(context, NotificationReceiver.class);
    intent.putExtra("title", title);
    intent.putExtra("text", text);
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
    // Schdedule notification
    AlarmManager manager = (AlarmManager)
context.getSystemService(Context.ALARM_SERVICE);
    manager.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP, time, pending);
}
```

Tenga en cuenta que el `42` por encima tiene que ser único para cada notificación programada, de lo contrario el `PendingIntent` s reemplazará unos a otros efectos no deseados que causan!

4. **Cancele una notificación** al reconstruir el `PendingIntent` asociado y cancelarlo en el `AlarmManager` sistema. El siguiente método cancela una notificación:

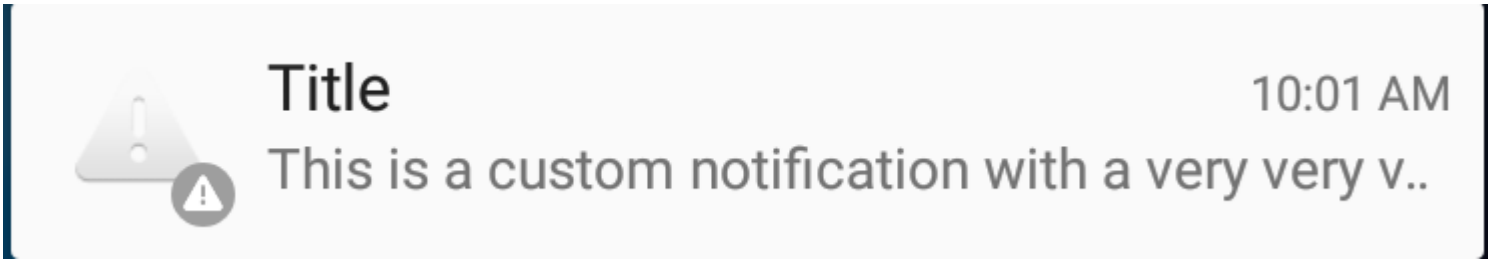
```
public static void cancelNotification(Context context, String title, String text) {
    Intent intent = new Intent(context, NotificationReceiver.class);
    intent.putExtra("title", title);
    intent.putExtra("text", text);
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
    // Cancel notification
    AlarmManager manager = (AlarmManager)
context.getSystemService(Context.ALARM_SERVICE);
    manager.cancel(pending);
}
```

¡Tenga en cuenta que el `42` anterior debe coincidir con el número del paso 3!

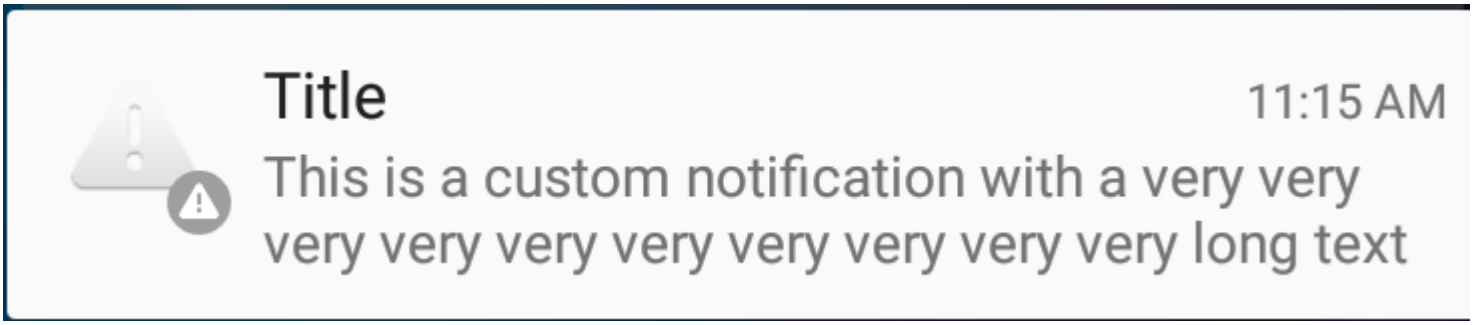
Establecer notificación personalizada: muestra el contenido completo del texto

Si desea que se muestre un texto largo en el contexto, debe configurar un contenido personalizado.

Por ejemplo, tienes esto:



Pero deseas que tu texto se muestre completamente:



Todo lo que necesita hacer es **agregar un estilo** a su contenido como a continuación:

```
  private void generateNotification(Context context) {
        String message = "This is a custom notification with a very very very very very very
very very very very very long text";
        Bitmap largeIcon = BitmapFactory.decodeResource(getResources(),
android.R.drawable.ic_dialog_alert);

        NotificationCompat.Builder builder = new NotificationCompat.Builder(context);

        builder.setContentTitle("Title").setContentText(message)
                .setSmallIcon(android.R.drawable.ic_dialog_alert)
                .setLargeIcon(largeIcon)
                .setAutoCancel(true)
                .setWhen(System.currentTimeMillis())
                .setStyle(new NotificationCompat.BigTextStyle().bigText(message));

        Notification notification = builder.build();
        NotificationManagerCompat notificationManager =
NotificationManagerCompat.from(context);
        notificationManager.notify(101, notification);
    }
```

Establecer el icono de notificación personalizado usando la biblioteca `Picasso`.

```
    PendingIntent pendingIntent = PendingIntent.getActivity(context,
    uniqueIntentId, intent, PendingIntent.FLAG_CANCEL_CURRENT);

    final RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
R.layout.remote_view_notification);
    remoteViews.setImageViewResource(R.id.remoteview_notification_icon,
R.mipmap.ic_navigation_favorites);

    Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
    NotificationCompat.Builder notificationBuilder =
            new NotificationCompat.Builder(context)
                    .setSmallIcon(R.mipmap.ic_navigation_favorites) //just dummy icon
                    .setContent(remoteViews) // here we apply our view
                    .setAutoCancel(true)
                    .setContentIntent(pendingIntent)
                    .setPriority(NotificationCompat.PRIORITY_DEFAULT);

    final Notification notification = notificationBuilder.build();
```

```
    if (android.os.Build.VERSION.SDK_INT >= 16) {
        notification.bigContentView = remoteViews;
    }

    NotificationManager notificationManager =
            (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);

    notificationManager.notify(uniqueIntentId, notification);


    //don't forget to include picasso to your build.gradle file.
    Picasso.with(context)
            .load(avatar)
            .into(remoteViews, R.id.remoteview_notification_icon, uniqueIntentId,
notification);
```

Y luego define un diseño dentro de tu carpeta de diseños:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:color/white"
    android:orientation="vertical">



    <ImageView
        android:id="@+id/remoteview_notification_icon"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:layout_marginRight="2dp"
        android:layout_weight="0"
        android:scaleType="centerCrop"/>
</LinearLayout>
```

Obtener dinámicamente el tamaño de píxel correcto para el icono grande

Si está creando una imagen, decodificando una imagen o cambiando el tamaño de una imagen para que se ajuste al área de imagen de notificación grande, puede obtener las dimensiones de píxeles correctas de la siguiente manera:

```
Resources resources = context.getResources();
int width  = resources.getDimensionPixelSize(android.R.dimen.notification_large_icon_width);
int height = resources.getDimensionPixelSize(android.R.dimen.notification_large_icon_height);
```

Notificación continua con botón de acción

```
// Cancel older notification with same id,
NotificationManager notificationMgr =
(NotificationManager)context.getSystemService(Context.NOTIFICATION_SERVICE);
notificationMgr.cancel(CALL_NOTIFY_ID);// any constant value
```

```
// Create Pending Intent,
Intent notificationIntent = null;
PendingIntent contentIntent = null;
notificationIntent = new Intent (context, YourActivityName);
contentIntent = PendingIntent.getActivity(context, 0, notificationIntent,
PendingIntent.FLAG_UPDATE_CURRENT);

// Notification builder
builder = new NotificationCompat.Builder(context);
builder.setContentText("Ongoing Notification..");
builder.setContentTitle("ongoing notification sample");
builder.setSmallIcon(R.drawable.notification_icon);
builder.setUsesChronometer(true);
builder.setDefaults(Notification.DEFAULT_LIGHTS);
builder.setContentIntent(contentIntent);
builder.setOngoing(true);

// Add action button in the notification
Intent intent = new Intent("action.name");
PendingIntent pIntent = PendingIntent.getBroadcast(context, 1, intent, 0);
builder.addAction(R.drawable.action_button_icon, "Action button name",pIntent);

// Notify using notificationMgr
Notification finalNotification = builder.build();
notificationMgr.notify(CALL_NOTIFY_ID, finalNotification);
```

Registre un receptor de difusión para la misma acción para manejar el evento de clic del botón de acción.

Lea Notificaciones en línea: https://riptutorial.com/es/android/topic/1347/notificaciones

Capítulo 173: Obtención de dimensiones de vista calculadas

Observaciones

Tenga en cuenta que una instancia de `ViewTreeObserver` asociada con una instancia de `View` puede dejar de ser válida mientras esa `View` aún esté activa. Desde el `View.getViewTreeObserver` View.getViewTreeObserver:

```
// The returned ViewTreeObserver observer is not guaranteed to remain
// valid for the lifetime of this View. If the caller of this method keeps
// a long-lived reference to ViewTreeObserver, it should always check for
// the return value of {@link ViewTreeObserver#isAlive()}.
```

Por lo tanto, si anteriormente ha agregado un servicio de escucha a una instancia de `ViewTreeObserver` y ahora desea eliminarlo, es más fácil llamar a `getViewTreeObserver` en la instancia de `View` correspondiente para recibir una instancia nueva de `ViewTreeObserver` . (Verificación de `isAlive` en una instancia existente es más trabajo para un pequeño beneficio; si `ViewTreeObserver` ya no está activo, ¡de todos modos obtendrá esa nueva referencia!)

Examples

Cálculo de dimensiones iniciales de vista en una actividad

```
package com.example;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.util.Log;
import android.view.View;
import android.view.ViewTreeObserver;

public class ExampleActivity extends Activity {

    @Override
    protected void onCreate(@Nullable final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_example);

        final View viewToMeasure = findViewById(R.id.view_to_measure);

        // viewToMeasure dimensions are not known at this point.
        // viewToMeasure.getWidth() and viewToMeasure.getHeight() both return 0,
        // regardless of on-screen size.

        viewToMeasure.getViewTreeObserver().addOnPreDrawListener(new
ViewTreeObserver.OnPreDrawListener() {
            @Override
            public boolean onPreDraw() {
                // viewToMeasure is now measured and laid out, and displayed dimensions are
```

```
known.
                logComputedViewDimensions(viewToMeasure.getWidth(),
viewToMeasure.getHeight());

                // Remove this listener, as we have now successfully calculated the desired
dimensions.
                viewToMeasure.getViewTreeObserver().removeOnPreDrawListener(this);

                // Always return true to continue drawing.
                return true;
            }
        });
    }

    private void logComputedViewDimensions(final int width, final int height) {
        Log.d("example", "viewToMeasure has width " + width);
        Log.d("example", "viewToMeasure has height " + height);
    }

}
```

Lea Obtención de dimensiones de vista calculadas en línea:
https://riptutorial.com/es/android/topic/115/obtencion-de-dimensiones-de-vista-calculadas

Capítulo 174: Obtención de nombres de fuentes del sistema y uso de las fuentes

Introducción

Los siguientes ejemplos muestran cómo recuperar los nombres predeterminados de las fuentes del sistema que se almacenan en el directorio */ system / fonts /* y cómo usar una fuente del sistema para establecer el tipo de letra de un elemento `TextView` .

Examples

Obtención de nombres de fuentes del sistema

```
ArrayList<String> fontNames = new ArrayList<String>();
File temp = new File("/system/fonts/");
String fontSuffix = ".ttf";

for(File font : temp.listFiles()) {
    String fontName = font.getName();
    if(fontName.endsWith(fontSuffix)) {
        fontNames.add(fontName.subSequence(0,fontName.lastIndexOf(fontSuffix)).toString());
    }
}
```

Aplicando una fuente del sistema a un TextView

En el siguiente código es necesario sustituir `fontsname` por el nombre de la fuente que desea utilizar:

```
TextView lblexample = (TextView) findViewById(R.id.lblexample);
lblexample.setTypeface(Typeface.createFromFile("/system/fonts/" + "fontsname" + ".ttf"));
```

Lea Obtención de nombres de fuentes del sistema y uso de las fuentes en línea: https://riptutorial.com/es/android/topic/10930/obtencion-de-nombres-de-fuentes-del-sistema-y-uso-de-las-fuentes

Capítulo 175: OkHttp

Examples

Interceptor de registro

`Interceptors` se utilizan para interceptar llamadas `OkHttp` . La razón para interceptar podría ser monitorear, reescribir y reintentar llamadas. Puede ser utilizado para solicitud de salida o respuesta entrante tanto.

```
class LoggingInterceptor implements Interceptor {
  @Override public Response intercept(Interceptor.Chain chain) throws IOException {
    Request request = chain.request();

    long t1 = System.nanoTime();
    logger.info(String.format("Sending request %s on %s%n%s",
        request.url(), chain.connection(), request.headers()));

    Response response = chain.proceed(request);

    long t2 = System.nanoTime();
    logger.info(String.format("Received response for %s in %.1fms%n%s",
        response.request().url(), (t2 - t1) / 1e6d, response.headers()));

    return response;
  }
}
```

Reescritura de respuestas

```
private static final Interceptor REWRITE_CACHE_CONTROL_INTERCEPTOR = new Interceptor() {
  @Override public Response intercept(Interceptor.Chain chain) throws IOException {
    Response originalResponse = chain.proceed(chain.request());
    return originalResponse.newBuilder()
        .header("Cache-Control", "max-age=60")
        .build();
  }
};
```

Ejemplo de uso básico

Me gusta envolver mi `OkHttp` en una clase llamada `HttpClient` por ejemplo, y en esta clase tengo métodos para cada uno de los principales verbos HTTP, `post` , `get` , `put` y `delete` , más comúnmente. (Por lo general, incluyo una interfaz, con el fin de mantenerla implementada, para poder cambiar fácilmente a una implementación diferente, si es necesario):

```
public class HttpClient implements HttpClientInterface{

private static final String TAG = OkHttpClient.class.getSimpleName();
public static final MediaType JSON
```

```
        = MediaType.parse("application/json; charset=utf-8");

OkHttpClient httpClient = new OkHttpClient();

@Override
public String post(String url, String json) throws IOException {
    Log.i(TAG, "Sending a post request with body:\n" + json + "\n to URL: " + url);

    RequestBody body = RequestBody.create(JSON, json);
    Request request = new Request.Builder()
            .url(url)
            .post(body)
            .build();
    Response response = httpClient.newCall(request).execute();
    return response.body().string();
}
```

La sintaxis es la misma para `put` , `get` y `delete` excepto por una palabra (`.put(body)`), por lo que también podría ser desagradable publicar ese código. El uso es bastante simple, simplemente llame al método apropiado en alguna `url` con algo de carga `json` y el método devolverá una cadena como resultado que luego podrá usar y analizar. Supongamos que la respuesta será un `json` , podemos crear un `JSONObject` fácilmente a partir de él:

```
String response = httpClient.post(MY_URL, JSON_PAYLOAD);
JSONObject json = new JSONObject(response);
// continue to parse the response according to it's structure
```

Llamada sincrónica Get

```
private final OkHttpClient client = new OkHttpClient();

  public void run() throws Exception {
    Request request = new Request.Builder()
        .url(yourUrl)
        .build();

    Response response = client.newCall(request).execute();
    if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

    Headers responseHeaders = response.headers();

    System.out.println(response.body().string());
  }
```

Asynchronous Get Call

```
private final OkHttpClient client = new OkHttpClient();

  public void run() throws Exception {
    Request request = new Request.Builder()
        .url(yourUrl)
        .build();

    client.newCall(request).enqueue(new Callback() {
      @Override public void onFailure(Call call, IOException e) {
```

```
        e.printStackTrace();
      }

      @Override
      public void onResponse(Call call, Response response) throws IOException {
        if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

        Headers responseHeaders = response.headers();

        System.out.println(response.body().string());
      }
    });
  }
```

Parámetros de formulario

```
private final OkHttpClient client = new OkHttpClient();

  public void run() throws Exception {
    RequestBody formBody = new FormBody.Builder()
        .add("search", "Jurassic Park")
        .build();
    Request request = new Request.Builder()
        .url("https://en.wikipedia.org/w/index.php")
        .post(formBody)
        .build();

    Response response = client.newCall(request).execute();
    if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

    System.out.println(response.body().string());
  }
```

Publicar una solicitud multiparte

```
 private static final String IMGUR_CLIENT_ID = "...";
  private static final MediaType MEDIA_TYPE_PNG = MediaType.parse("image/png");

  private final OkHttpClient client = new OkHttpClient();

  public void run() throws Exception {
    // Use the imgur image upload API as documented at https://api.imgur.com/endpoints/image
    RequestBody requestBody = new MultipartBody.Builder()
        .setType(MultipartBody.FORM)
        .addFormDataPart("title", "Square Logo")
        .addFormDataPart("image", "logo-square.png",
            RequestBody.create(MEDIA_TYPE_PNG, new File("website/static/logo-square.png")))
        .build();

    Request request = new Request.Builder()
        .header("Authorization", "Client-ID " + IMGUR_CLIENT_ID)
        .url("https://api.imgur.com/3/image")
        .post(requestBody)
        .build();

    Response response = client.newCall(request).execute();
    if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);
```

```
    System.out.println(response.body().string());
  }
```

Configurando OkHttp

Agarrar a través de Maven:

```
<dependency>
  <groupId>com.squareup.okhttp3</groupId>
  <artifactId>okhttp</artifactId>
  <version>3.6.0</version>
</dependency>
```

o Gradle:

```
compile 'com.squareup.okhttp3:okhttp:3.6.0'
```

Lea OkHttp en línea: https://riptutorial.com/es/android/topic/3625/okhttp

Capítulo 176: Okio

Examples

Descargar / Implementar

Descarga la última versión de JAR o graba a través de Maven:

```
<dependency>
    <groupId>com.squareup.okio</groupId>
    <artifactId>okio</artifactId>
    <version>1.12.0</version>
</dependency>
```

o Gradle:

```
compile 'com.squareup.okio:okio:1.12.0'
```

Decodificador PNG

La decodificación de los fragmentos de un archivo PNG demuestra Okio en la práctica.

```
private static final ByteString PNG_HEADER = ByteString.decodeHex("89504e470d0a1a0a");

public void decodePng(InputStream in) throws IOException {
  try (BufferedSource pngSource = Okio.buffer(Okio.source(in))) {
    ByteString header = pngSource.readByteString(PNG_HEADER.size());
    if (!header.equals(PNG_HEADER)) {
      throw new IOException("Not a PNG.");
    }

    while (true) {
      Buffer chunk = new Buffer();

      // Each chunk is a length, type, data, and CRC offset.
      int length = pngSource.readInt();
      String type = pngSource.readUtf8(4);
      pngSource.readFully(chunk, length);
      int crc = pngSource.readInt();

      decodeChunk(type, chunk);
      if (type.equals("IEND")) break;
    }
  }
}

private void decodeChunk(String type, Buffer chunk) {
  if (type.equals("IHDR")) {
    int width = chunk.readInt();
    int height = chunk.readInt();
    System.out.printf("%08x: %s %d x %d%n", chunk.size(), type, width, height);
  } else {
    System.out.printf("%08x: %s%n", chunk.size(), type);
```

```
    }
}
```

ByteStrings y Buffers

ByteStrings y Buffers

Okio se basa en dos tipos que combinan muchas capacidades en una API sencilla:

ByteString es una secuencia inmutable de bytes. Para datos de personajes, String es fundamental. ByteString es el hermano perdido de String, lo que facilita el tratamiento de los datos binarios como un valor. Esta clase es ergonómica: sabe cómo codificarse y decodificarse como hexadecimal, base64 y UTF-8.

Buffer es una secuencia mutable de bytes. Al igual que ArrayList, no es necesario que dimensione su búfer por adelantado. Lee y escribe buffers como una cola: escriba datos al final y léalos desde el frente. No hay obligación de gestionar posiciones, límites o capacidades.

Internamente, `ByteString` y `Buffer` hacen algunas cosas inteligentes para ahorrar CPU y memoria. Si codifica una cadena UTF-8 como ByteString, almacena en caché una referencia a esa cadena para que si la decodifica más tarde, no haya trabajo que hacer.

`Buffer` se implementa como una lista de segmentos vinculados. Cuando mueve los datos de un búfer a otro, se reasigna la propiedad de los segmentos en lugar de copiar los datos. Este enfoque es particularmente útil para los programas multiproceso: un subproceso que habla a la red puede intercambiar datos con un subproceso de trabajo sin ningún tipo de copia o ceremonia.

Lea Okio en línea: https://riptutorial.com/es/android/topic/9952/okio

Capítulo 177: Optimización del núcleo de Android

Examples

Configuración de RAM baja

Android ahora es compatible con dispositivos con 512 MB de RAM. Esta documentación está destinada a ayudar a los OEM a optimizar y configurar Android 4.4 para dispositivos con poca memoria. Varias de estas optimizaciones son lo suficientemente genéricas para que también puedan aplicarse a versiones anteriores.

Habilitar bandera de dispositivo bajo de RAM

Estamos introduciendo una nueva API llamada ActivityManager.isLowRamDevice () para que las aplicaciones determinen si deberían desactivar las funciones específicas de uso intensivo de memoria que funcionan mal en dispositivos con poca memoria.

Para dispositivos de 512 MB, se espera que devuelva esta API: "verdadero" Puede activarse mediante la siguiente propiedad del sistema en el archivo make del dispositivo.

```
PRODUCT_PROPERTY_OVERRIDES += ro.config.low_ram=true
```

Deshabilitar JIT

El uso de la memoria JIT en todo el sistema depende de la cantidad de aplicaciones que se ejecutan y de la huella del código de esas aplicaciones. El JIT establece un tamaño máximo de caché de código traducido y toca las páginas que contiene según sea necesario. Los costos de JIT en algún lugar entre 3M y 6M en un sistema de ejecución típico.

Las aplicaciones grandes tienden a maximizar la memoria caché de código bastante rápidamente (lo que por defecto ha sido 1M). En promedio, el uso del caché JIT se ejecuta en algún lugar entre 100K y 200K bytes por aplicación. Reducir el tamaño máximo de la memoria caché puede ayudar de alguna manera con el uso de la memoria, pero si se establece demasiado bajo, enviará el JIT a un modo de paliza. Para los dispositivos con muy poca memoria, recomendamos que el JIT se desactive por completo.

Esto se puede lograr agregando la siguiente línea al archivo make del producto:

```
PRODUCT_PROPERTY_OVERRIDES += dalvik.vm.jit.codecachesize=0
```

Cómo agregar un regulador de CPU

El gobernador de la CPU en sí mismo es solo un archivo C, que se encuentra en kernel_source / drivers / cpufreq /, por ejemplo: cpufreq_smartass2.c. Usted mismo es responsable de encontrar

el gobernador (busque en un repositorio de kernel existente para su dispositivo). Pero para poder llamar y compilar este archivo en su kernel, deberá realizar los siguientes cambios:

1. Copie su archivo de gobernador (cpufreq_govname.c) y vaya a kernel_source / drivers / cpufreq, ahora péguelo.
2. y abra Kconfig (esta es la interfaz del diseño del menú de configuración) cuando agregue un kernel, desea que aparezca en su configuración. Puedes hacerlo agregando la elección de gobernador.

```
config CPU_FREQ_GOV_GOVNAMEHERE
tristate "'gov_name_lowercase' cpufreq governor"
depends on CPU_FREQ
help
governor' - a custom governor!
```

por ejemplo, para smartassV2.

```
config CPU_FREQ_GOV_SMARTASS2
 tristate "'smartassV2' cpufreq governor"
 depends on CPU_FREQ
 help
 'smartassV2' - a "smart" optimized governor!
```

Además de agregar la opción, también debe declarar la posibilidad de que el gobernador sea elegido como gobernador predeterminado.

```
 config CPU_FREQ_DEFAULT_GOV_GOVNAMEHERE
 bool "gov_name_lowercase"
 select CPU_FREQ_GOV_GOVNAMEHERE
 help
 Use the CPUFreq governor 'govname' as default.
```

por ejemplo, para smartassV2.

```
config CPU_FREQ_DEFAULT_GOV_SMARTASS2
 bool "smartass2"
 select CPU_FREQ_GOV_SMARTASS2
 help
 Use the CPUFreq governor 'smartassV2' as default.
```

- ¿No puedes encontrar el lugar correcto para ponerlo? Simplemente busque `"CPU_FREQ_GOV_CONSERVATIVE"` , y coloque el código debajo, lo mismo cuenta para `"CPU_FREQ_DEFAULT_GOV_CONSERVATIVE"`

Ahora que Kconfig ha terminado, puede guardar y cerrar el archivo.

3. Mientras `/drivers/cpufreq` carpeta `/drivers/cpufreq` , abra Makefile. En Makefile, agregue la línea correspondiente a su CPU Governor. por ejemplo:

```
obj-$(CONFIG_CPU_FREQ_GOV_SMARTASS2)    += cpufreq_smartass2.o
```

Tenga en cuenta que no llama al archivo C nativo, sino al archivo O! que es el archivo C compilado. Guarda el archivo.

4. Mover a: `kernel_source/includes/linux` . Ahora abre `cpufreq.h` Desplázate hacia abajo hasta que veas algo como:

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_ONDEMAND)
 extern struct cpufreq_governor cpufreq_gov_ondemand;
 #define CPUFREQ_DEFAULT_GOVERNOR    (&amp;cpufreq_gov_ondemand)
```

(otros gobernadores de la CPU también se enumeran allí)

Ahora agregue su entrada con el Gobernador de CPU seleccionado, ejemplo:

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_SMARTASS2)
 extern struct cpufreq_governor cpufreq_gov_smartass2;
 #define CPUFREQ_DEFAULT_GOVERNOR (&amp;cpufreq_gov_smartass2)
```

Guarde el archivo y ciérrelo.

La configuración inicial del regulador de CPU ahora está completa. cuando haya realizado todos los pasos correctamente, debería poder elegir su gobernador en el menú (`menuconfig` , `xconfig` , `gconfig` , `nconfig`). Una vez comprobado en el menú se incluirá en el kernel.

Confirme que es casi lo mismo que en las instrucciones anteriores: agregue smartassV2 y lulzactive Governor commit

Programadores de E / S

Puede mejorar su kernel agregando nuevos programadores de E / S si es necesario. A nivel mundial, los gobernadores y planificadores son los mismos; ambos proporcionan una manera de cómo debería funcionar el sistema. Sin embargo, para los programadores se trata del flujo de datos de entrada / salida, excepto por la configuración de la CPU. Los programadores de E / S deciden cómo se programará una próxima actividad de E / S. Los programadores estándar, como *noop* o *cfq,* tienen un rendimiento muy razonable.

Los programadores de E / S se pueden encontrar en *kernel_source / block* .

1. Copie su archivo del programador de E / S (por ejemplo, *sio-iosched.c*) y navegue hasta *kernel_source / block* . Pegue el archivo del planificador allí.

2. Ahora abra *Kconfig.iosched* y agregue su elección a *Kconfig* , por ejemplo para *SIO* :

```
config IOSCHED_SIO
  tristate "Simple I/O scheduler"
  default y
  ---help---
    The Simple I/O scheduler is an extremely simple scheduler,
    based on noop and deadline, that relies on deadlines to
    ensure fairness. The algorithm does not do any sorting but
    basic merging, trying to keep a minimum overhead. It is aimed
```

```
    mainly for aleatory access devices (eg: flash devices).
```

3. A continuación, establezca la opción de elección predeterminada de la siguiente manera:

   ```
   default "sio" if DEFAULT_SIO
   ```

   Guarda el archivo.

4. Abra el *archivo Makefile* en *kernel_source / block /* y simplemente agregue la siguiente línea para *SIO* :

   ```
   obj-$(CONFIG_IOSCHED_SIO)    += sio-iosched.o
   ```

   Guarda el archivo y listo! Los programadores de E / S ahora deberían aparecer en la configuración del menú.

Compromiso similar en GitHub: se agregó el planificador de E / S simple .

Lea Optimización del núcleo de Android en línea:
https://riptutorial.com/es/android/topic/9106/optimizacion-del-nucleo-de-android

Capítulo 178: Optimización del rendimiento

Introducción

El rendimiento de su aplicación es un elemento crucial de la experiencia del usuario. Intente evitar los patrones de mal desempeño, como trabajar en el hilo de la interfaz de usuario y aprenda a escribir aplicaciones rápidas y con capacidad de respuesta.

Examples

Guardar búsquedas de vista con el patrón ViewHolder

Especialmente en un `ListView` , puede tener problemas de rendimiento al hacer demasiadas llamadas `findViewById()` durante el desplazamiento. Al usar el patrón `ViewHolder` , puede guardar estas búsquedas y mejorar el rendimiento de su `ListView` .

Si su elemento de lista contiene un solo `TextView` , cree una clase `ViewHolder` para almacenar la instancia:

```
static class ViewHolder {
    TextView myTextView;
}
```

Al crear su elemento de lista, adjunte un objeto `ViewHolder` al elemento de lista:

```
public View getView(int position, View convertView, ViewGroup parent) {
    Item i = getItem(position);
    if(convertView == null) {
        convertView = LayoutInflater.from(getContext()).inflate(R.layout.list_item, parent,
false);

        // Create a new ViewHolder and save the TextView instance
        ViewHolder holder = new ViewHolder();
        holder.myTextView = (TextView)convertView.findViewById(R.id.my_text_view);
        convertView.setTag(holder);
    }

    // Retrieve the ViewHolder and use the TextView
    ViewHolder holder = (ViewHolder)convertView.getTag();
    holder.myTextView.setText(i.getText());

    return convertView;
}
```

Usando este patrón, solo se llamará a `findViewById()` cuando se crea una nueva `View` y el `ListView` puede reciclar sus vistas de manera mucho más eficiente.

Lea Optimización del rendimiento en línea:
https://riptutorial.com/es/android/topic/8711/optimizacion-del-rendimiento

Capítulo 179: ORMLite en Android

Examples

Ejemplo de Android OrmLite sobre SQLite

ORMLite es un paquete de asignación relacional de objetos que proporciona una funcionalidad simple y liviana para la persistencia de objetos Java en bases de datos SQL, evitando la complejidad y la sobrecarga de paquetes ORM más estándar.

Hablando para Android, OrmLite se implementa en la base de datos compatible con el sistema, SQLite. Hace llamadas directas a la API para acceder a SQLite.

Configuración de Gradle

Para empezar, debes incluir el paquete en la versión de compilación.

```
 // https://mvnrepository.com/artifact/com.j256.ormlite/ormlite-android
compile group: 'com.j256.ormlite', name: 'ormlite-android', version: '5.0'
POJO configuration
```

A continuación, debe configurar un POJO para persistir en la base de datos. Aquí se debe tener cuidado con las anotaciones:

- Agregue la anotación @DatabaseTable en la parte superior de cada clase. También puedes usar @Entity.
- Agregue la anotación @DatabaseField justo antes de cada campo a persistir. También puedes usar @Column y otros.
- Agregue un constructor sin argumentos a cada clase con al menos visibilidad del paquete.

```
@DatabaseTable(tableName = "form_model")
public class FormModel implements Serializable {

   @DatabaseField(generatedId = true)
   private Long id;
   @DatabaseField(dataType = DataType.SERIALIZABLE)
   ArrayList<ReviewItem> reviewItems;

   @DatabaseField(index = true)
   private String username;

   @DatabaseField
   private String createdAt;

   public FormModel() {
   }

   public FormModel(ArrayList<ReviewItem> reviewItems, String username, String createdAt) {
       this.reviewItems = reviewItems;
```

```
        this.username = username;
        this.createdAt = createdAt;
    }
}
```

En el ejemplo anterior hay una tabla (form_model) con 4 campos.

El campo id es un índice generado automáticamente.

nombre de usuario es un índice de la base de datos.

Se puede encontrar más información sobre la anotación en la documentación oficial .

Ayudante de base de datos

Para continuar, deberá crear una clase auxiliar de base de datos que extienda la clase OrmLiteSqliteOpenHelper.

Esta clase crea y actualiza la base de datos cuando su aplicación está instalada y también puede proporcionar las clases DAO utilizadas por sus otras clases.

DAO significa Objeto de acceso a datos, proporciona toda la funcionalidad de scrum y se especializa en el manejo de una sola clase persistente.

La clase auxiliar debe implementar los siguientes dos métodos:

- onCreate (SQLiteDatabase sqliteDatabase, ConnectionSource connectionSource);

  onCreate crea la base de datos cuando su aplicación se instala por primera vez

- onUpgrade (base de datos SQLiteDatabase, ConnectionSource connectionSource, int oldVersion, int newVersion);

  onUpgrade maneja la actualización de las tablas de la base de datos cuando actualiza su aplicación a una nueva versión

Ejemplo de clase de Asistente de base de datos:

```
  public class OrmLite extends OrmLiteSqliteOpenHelper {

        //Database name
        private static final String DATABASE_NAME = "gaia";
        //Version of the database. Changing the version will call {@Link OrmLite.onUpgrade}
        private static final int DATABASE_VERSION = 2;

        /**
         * The data access object used to interact with the Sqlite database to do C.R.U.D
operations.
         */
        private Dao<FormModel, Long> todoDao;
```

```
        public OrmLite(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION,
                    /**
                     * R.raw.ormlite_config is a reference to the ormlite_config2.txt file in
the
                     * /res/raw/ directory of this project
                     * */
                    R.raw.ormlite_config2);
        }

        @Override
        public void onCreate(SQLiteDatabase database, ConnectionSource connectionSource) {
            try {

                /**
                 * creates the database table
                 */
                TableUtils.createTable(connectionSource, FormModel.class);

            } catch (SQLException e) {
                e.printStackTrace();
            } catch (java.sql.SQLException e) {
                e.printStackTrace();
            }
        }
        /*
            It is called when you construct a SQLiteOpenHelper with version newer than the
version of the opened database.
         */
        @Override
        public void onUpgrade(SQLiteDatabase database, ConnectionSource connectionSource,
                              int oldVersion, int newVersion) {
            try {
                /**
                 * Recreates the database when onUpgrade is called by the framework
                 */
                TableUtils.dropTable(connectionSource, FormModel.class, false);
                onCreate(database, connectionSource);

            } catch (SQLException | java.sql.SQLException e) {
                e.printStackTrace();
            }
        }

        /**
         * Returns an instance of the data access object
         * @return
         * @throws SQLException
         */
        public Dao<FormModel, Long> getDao() throws SQLException {
            if(todoDao == null) {
                try {
                    todoDao = getDao(FormModel.class);
                } catch (java.sql.SQLException e) {
                    e.printStackTrace();
                }
            }
            return todoDao;
        }
    }
```

Objeto persistente a SQLite

Finalmente, la clase que persiste el objeto hasta la base de datos.

```
     public class ReviewPresenter {
    Dao<FormModel, Long> simpleDao;


    public ReviewPresenter(Application application) {
        this.application = (GaiaApplication) application;
        simpleDao = this.application.getHelper().getDao();
    }

    public void storeFormToSqLite(FormModel form) {

        try {
            simpleDao.create(form);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        List<FormModel> list = null;
        try {
// query for all of the data objects in the database
            list = simpleDao.queryForAll();
        } catch (SQLException e) {
            e.printStackTrace();
        }
// our string builder for building the content-view
        StringBuilder sb = new StringBuilder();
        int simpleC = 1;
        for (FormModel simple : list) {
            sb.append('#').append(simpleC).append(":
").append(simple.getUsername()).append('\n');
            simpleC++;
        }
        System.out.println(sb.toString());
    }

    //Query to database to get all forms by username
    public List<FormModel> getAllFormsByUsername(String username) {
        List<FormModel> results = null;
        try {
            results = simpleDao.queryBuilder().where().eq("username",
PreferencesManager.getInstance().getString(Constants.USERNAME)).query();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return results;
    }
}
```

El descriptor de acceso del DOA en el constructor de la clase anterior se define como:

```
 private OrmLite dbHelper = null;

/*
Provides the SQLite Helper Object among the application
```

```
 */
public OrmLite getHelper() {
    if (dbHelper == null) {
        dbHelper = OpenHelperManager.getHelper(this, OrmLite.class);
    }
    return dbHelper;
}
```

Lea ORMLite en Android en línea: https://riptutorial.com/es/android/topic/7571/ormlite-en-android

Capítulo 180: Otto Event Bus

Observaciones

Otto está en desuso a favor de `RxJava` y `RxAndroid` . Estos proyectos permiten el mismo modelo de programación controlado por eventos que Otto, pero son más capaces y ofrecen un mejor control de los hilos.

Examples

Pasando un evento

Este ejemplo describe cómo pasar un evento usando el Otto Event Bus .

Para usar Otto Event Bus en **Android Studio** , debe insertar la siguiente declaración en su módulo de archivos de gradle:

```
dependencies {
    compile 'com.squareup:otto:1.3.8'
}
```

El evento que nos gustaría pasar es un simple objeto Java:

```
public class DatabaseContentChangedEvent {
    public String message;

    public DatabaseContentChangedEvent(String message) {
        this.message = message;
    }
}
```

Necesitamos un autobús para enviar eventos. Esto es típicamente un singleton:

```
import com.squareup.otto.Bus;

public final class BusProvider {
    private static final Bus mBus = new Bus();

    public static Bus getInstance() {
        return mBus;
    }

    private BusProvider() {
    }
}
```

Para enviar un evento solo necesitamos nuestro BusProvider y su método de `post` . Aquí enviamos un evento si se completa la acción de una AsyncTask:

```
public abstract class ContentChangingTask extends AsyncTask<Object, Void, Void> {

    ...

    @Override
    protected void onPostExecute(Void param) {
        BusProvider.getInstance().post(
            new DatabaseContentChangedEvent("Content changed")
        );
    }
}
```

Recibiendo un evento

Para recibir un evento es necesario implementar un método con el tipo de evento como parámetro y anotarlo con `@Subscribe` . Además, debe registrar / anular el registro de la instancia de su objeto en el `BusProvider` (ver ejemplo *Enviando un evento*):

```
public class MyFragment extends Fragment {
    private final static String TAG = "MyFragment";

    ...

    @Override
    public void onResume() {
        super.onResume();
        BusProvider.getInstance().register(this);
    }

    @Override
    public void onPause() {
        super.onPause();
        BusProvider.getInstance().unregister(this);
    }

    @Subscribe
    public void onDatabaseContentChanged(DatabaseContentChangedEvent event) {
        Log.i(TAG, "onDatabaseContentChanged: "+event.message);
    }
}
```

Importante: para recibir ese evento debe existir una instancia de la clase. Este no suele ser el caso cuando desea enviar un resultado de una actividad a otra actividad. Así que revisa tu caso de uso para el autobús del evento.

Lea Otto Event Bus en línea: https://riptutorial.com/es/android/topic/6068/otto-event-bus

Capítulo 181: Paginación en RecyclerView

Introducción

La paginación es un problema común con muchas aplicaciones móviles que necesitan tratar con listas de datos. La mayoría de las aplicaciones móviles ahora están comenzando a adoptar el modelo de "página sin fin", donde el desplazamiento se carga automáticamente en el nuevo contenido. El adaptador sin fin CWAC facilita mucho el uso de este patrón en aplicaciones de Android

Examples

MainActivity.java

```
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.widget.TextView;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.VolleyLog;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";
    private Toolbar toolbar;

    private TextView tvEmptyView;
    private RecyclerView mRecyclerView;
    private DataAdapter mAdapter;
    private LinearLayoutManager mLayoutManager;
    private int mStart=0,mEnd=20;
    private List<Student> studentList;
    private List<Student> mTempCheck;
    public static int pageNumber;
    public int total_size=0;
```

```
protected Handler handler;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    pageNumber = 1;
    toolbar = (Toolbar) findViewById(R.id.toolbar);
    tvEmptyView = (TextView) findViewById(R.id.empty_view);
    mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
    studentList = new ArrayList<>();
    mTempCheck=new ArrayList<>();
    handler = new Handler();
    if (toolbar != null) {
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle("Android Students");

    }

    mRecyclerView.setHasFixedSize(true);
    mLayoutManager = new LinearLayoutManager(this);
    mRecyclerView.setLayoutManager(mLayoutManager);
    mAdapter = new DataAdapter(studentList, mRecyclerView);
    mRecyclerView.setAdapter(mAdapter);
    GetGroupData("" + mStart, "" + mEnd);
    mAdapter.setOnLoadMoreListener(new OnLoadMoreListener() {
        @Override
        public void onLoadMore() {
            if( mTempCheck.size()> 0) {
                studentList.add(null);
                mAdapter.notifyItemInserted(studentList.size() - 1);
                int start = pageNumber * 20;
                start = start + 1;
                ++ pageNumber;
                mTempCheck.clear();
                GetData("" + start,""+ mEnd);
            }
        }
    });
}

public void GetData(final String LimitStart, final String LimitEnd) {
    Map<String, String> params = new HashMap<>();
    params.put("LimitStart", LimitStart);
    params.put("Limit", LimitEnd);
    Custom_Volly_Request jsonObjReq = new Custom_Volly_Request(Request.Method.POST,
            "Your php file link", params,
            new Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                    Log.d("ResponseSuccess",response.toString());
                    // handle the data from the servoce
                }
            }, new Response.ErrorListener() {

        @Override
        public void onErrorResponse(VolleyError error) {
            VolleyLog.d("ResponseErrorVolly: " + error.getMessage());

        }));
```

```
    }
    // load initial data
    private void loadData(int start,int end,boolean notifyadapter) {
        for (int i = start; i <= end; i++) {
            studentList.add(new Student("Student " + i, "androidstudent" + i + "@gmail.com"));
            if(notifyadapter)
                mAdapter.notifyItemInserted(studentList.size());
        }
    }
}
```

OnLoadMoreListener.java

interfaz pública OnLoadMoreListener {void onLoadMore (); }

DataAdapter.java

```
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import java.util.List;

public class DataAdapter extends RecyclerView.Adapter {
    private final int VIEW_ITEM = 1;
    private final int VIEW_PROG = 0;

    private List<Student> studentList;

    // The minimum amount of items to have below your current scroll position
    // before loading more.
    private int visibleThreshold = 5;
    private int lastVisibleItem, totalItemCount;
    private boolean loading;
    private OnLoadMoreListener onLoadMoreListener;



    public DataAdapter(List<Student> students, RecyclerView recyclerView) {
        studentList = students;
        if (recyclerView.getLayoutManager() instanceof LinearLayoutManager) {
            final LinearLayoutManager linearLayoutManager = (LinearLayoutManager)
recyclerView.getLayoutManager();
                    recyclerView.addOnScrollListener(new RecyclerView.OnScrollListener() {
                        @Override
                        public void onScrolled(RecyclerView recyclerView, int dx, int dy) {
                            super.onScrolled(recyclerView, dx, dy);
                            totalItemCount = linearLayoutManager.getItemCount();
                            lastVisibleItem =
linearLayoutManager.findLastVisibleItemPosition();
                            if (! loading && totalItemCount <= (lastVisibleItem +
visibleThreshold)) {
                                if (onLoadMoreListener != null) {
```

```
                                    onLoadMoreListener.onLoadMore();
                                }
                                loading = true;
                            }
                        }
                    });
        }
    }

    @Override
    public int getItemViewType(int position) {

            return studentList.get(position) != null ? VIEW_ITEM : VIEW_PROG;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        RecyclerView.ViewHolder vh;
        if (viewType == VIEW_ITEM) {
            View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_row,
parent, false);
            vh = new StudentViewHolder(v);
        } else {
            View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.progress_item,
parent, false);
            vh = new ProgressViewHolder(v);
        }
        return vh;
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
        if (holder instanceof StudentViewHolder) {
            Student singleStudent=studentList.get(position);
            ((StudentViewHolder) holder).tvName.setText(singleStudent.getName());
            ((StudentViewHolder) holder).tvEmailId.setText(singleStudent.getEmailId());
            ((StudentViewHolder) holder).student= singleStudent;
        } else {
            ((ProgressViewHolder) holder).progressBar.setIndeterminate(true);
        }
    }

    public void setLoaded(boolean state) {
        loading = state;
    }

    @Override
    public int getItemCount() {
        return studentList.size();
    }

    public void setOnLoadMoreListener(OnLoadMoreListener onLoadMoreListener) {
        this.onLoadMoreListener = onLoadMoreListener;
    }


    //
    public static class StudentViewHolder extends RecyclerView.ViewHolder {
        public TextView tvName;

        public TextView tvEmailId;
```

```
        public Student student;

        public StudentViewHolder(View v) {
            super(v);
            tvName = (TextView) v.findViewById(R.id.tvName);
            tvEmailId = (TextView) v.findViewById(R.id.tvEmailId);


        }
    }

    public static class ProgressViewHolder extends RecyclerView.ViewHolder {
        public ProgressBar progressBar;

        public ProgressViewHolder(View v) {
            super(v);
            progressBar = (ProgressBar) v.findViewById(R.id.progressBar1);
        }
    }

}
```

Lea Paginación en RecyclerView en línea: https://riptutorial.com/es/android/topic/9243/paginacion-en-recyclerview

Capítulo 182: Pantallas de apoyo con diferentes resoluciones, tamaños

Observaciones

Términos y conceptos

Tamaño de pantalla

Tamaño físico real, medido según la diagonal de la pantalla. Para simplificar, Android agrupa todos los tamaños de pantalla reales en cuatro tamaños generalizados: pequeño, normal, grande y extra grande.

Densidad de pantalla

La cantidad de píxeles dentro de un área física de la pantalla; Generalmente referido como dpi (puntos por pulgada). Por ejemplo, una pantalla de densidad "baja" tiene menos píxeles dentro de un área física determinada, en comparación con una pantalla de densidad "normal" o "alta". Para simplificar, Android agrupa todas las densidades de pantalla reales en seis densidades generalizadas: baja, media, alta, extra alta, extra extra alta y extra extra extra alta.

Orientación

La orientación de la pantalla desde el punto de vista del usuario. Esto es horizontal o vertical, lo que significa que la relación de aspecto de la pantalla es ancha o alta, respectivamente. Tenga en cuenta que no solo los diferentes dispositivos operan en diferentes orientaciones de manera predeterminada, sino que la orientación puede cambiar en el tiempo de ejecución cuando el usuario gira el dispositivo. Resolución El número total de píxeles físicos en una pantalla. Al agregar soporte para múltiples pantallas, las aplicaciones no funcionan directamente con resolución; las aplicaciones deben ocuparse únicamente del tamaño y la densidad de la pantalla, como lo especifican los grupos de tamaño y densidad generalizados. Píxel independiente de densidad (dp) Una unidad de píxeles virtual que debe usar al definir el diseño de la interfaz de usuario, para expresar las dimensiones o la posición del diseño de forma independiente de la densidad. El píxel independiente de la densidad es equivalente a un píxel físico en una pantalla de 160 ppp, que es la densidad de referencia asumida por el sistema para una pantalla de densidad "media". En tiempo de ejecución, el sistema maneja de forma transparente cualquier escala de las unidades dp, según sea necesario, en función de la densidad real de la pantalla en uso. La conversión de unidades dp a píxeles de pantalla es simple: px = dp * (dpi / 160). Por ejemplo, en una pantalla de 240 ppp, 1 dp equivale a 1.5 píxeles físicos. Siempre debe usar unidades

dp al definir la interfaz de usuario de su aplicación, para garantizar una visualización adecuada de su interfaz de usuario en pantallas con diferentes densidades.

Unidades

- **px**

  Píxeles: corresponde a los píxeles reales en la pantalla.

- **en**

  Pulgadas - basado en el tamaño físico de la pantalla. 1 pulgada = 2.54 centímetros

- **mm**

  Milímetros - basado en el tamaño físico de la pantalla.

- **pt**

  Puntos: 1/72 de pulgada según el tamaño físico de la pantalla.

- **dp o dip**

  Píxeles independientes de la densidad: una unidad abstracta que se basa en la densidad física de la pantalla. Estas unidades son relativas a una pantalla de 160 ppp, por lo que un dp es un píxel en una pantalla de 160 ppp. La proporción de dp a píxel cambiará con la densidad de la pantalla, pero no necesariamente en proporción directa. Nota: El compilador acepta tanto "dip" como "dp", aunque "dp" es más consistente con "sp".

- **sp**

  Píxeles independientes de la escala: esto es como la unidad dp, pero también se escala según la preferencia de tamaño de fuente del usuario. Se recomienda utilizar esta unidad cuando especifique tamaños de fuente, por lo que se ajustarán tanto para la densidad de la pantalla como para las preferencias del usuario. De entender la independencia de densidad en Android:

Unidad	Descripción	Unidades por pulgada física	Densidad Independiente	Mismo tamaño físico en cada pantalla
px	Pixeles	Varía	No	No
en	Pulgadas	1	Sí	Sí
mm	Milimetros	25.4	Sí	Sí
pt	Puntos	72	Sí	Sí
dp	Densidad de píxeles independientes	~ 160	Sí	No
sp	Escala de píxeles independientes	~ 160	Sí	No

Referencias:

- https://developer.android.com/guide/practices/screens_support.html
- http://developer.android.com/guide/topics/resources/more-resources.html

Examples

Uso de calificadores de configuración

Android admite varios calificadores de configuración que le permiten controlar cómo el sistema selecciona sus recursos alternativos según las características de la pantalla del dispositivo actual. Un calificador de configuración es una cadena que puede adjuntar a un directorio de recursos en su proyecto de Android y especifica la configuración para la cual se diseñan los recursos.

Para utilizar un calificador de configuración:

1. Cree un nuevo directorio en el directorio res / de su proyecto y asígnele un nombre con el formato: `<resources_name>-<qualifier>` . `<resources_name>` es el nombre del recurso estándar (como drawable o layout).
2. `<qualifier>` es un calificador de configuración, que especifica la configuración de la pantalla para la cual se utilizarán estos recursos (como hdpi o xlarge).

Por ejemplo, los siguientes directorios de recursos de aplicaciones proporcionan diferentes diseños de diseño para diferentes tamaños de pantalla y diferentes elementos dibujables. Utilice el `mipmap/` carpetas para los iconos de inicio.

```
res/layout/my_layout.xml               // layout for normal screen size ("default")
res/layout-large/my_layout.xml         // layout for large screen size
res/layout-xlarge/my_layout.xml        // layout for extra-large screen size
res/layout-xlarge-land/my_layout.xml   // layout for extra-large in landscape orientation
```

```
res/drawable-mdpi/graphic.png         // bitmap for medium-density
res/drawable-hdpi/graphic.png         // bitmap for high-density
res/drawable-xhdpi/graphic.png        // bitmap for extra-high-density
res/drawable-xxhdpi/graphic.png       // bitmap for extra-extra-high-density

res/mipmap-mdpi/my_icon.png         // launcher icon for medium-density
res/mipmap-hdpi/my_icon.png         // launcher icon for high-density
res/mipmap-xhdpi/my_icon.png        // launcher icon for extra-high-density
res/mipmap-xxhdpi/my_icon.png       // launcher icon for extra-extra-high-density
res/mipmap-xxxhdpi/my_icon.png      // launcher icon for extra-extra-extra-high-density
```

Convertir dp y sp a píxeles

Cuando necesita establecer un valor de píxel para algo como `Paint.setTextSize` pero aún desea que se escale según el dispositivo, puede convertir los valores dp y sp.

```
DisplayMetrics metrics = Resources.getSystem().getDisplayMetrics();
float pixels = TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, 12f, metrics);

DisplayMetrics metrics = Resources.getSystem().getDisplayMetrics();
float pixels = TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, 12f, metrics);
```

Alternativamente, puede convertir un recurso de dimensión en píxeles si tiene un contexto para cargar el recurso.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="size_in_sp">12sp</dimen>
    <dimen name="size_in_dp">12dp</dimen>
</resources>

// Get the exact dimension specified by the resource
float pixels = context.getResources().getDimension(R.dimen.size_in_sp);
float pixels = context.getResources().getDimension(R.dimen.size_in_dp);

// Get the dimension specified by the resource for use as a size.
// The value is rounded down to the nearest integer but is at least 1px.
int pixels = context.getResources().getDimensionPixelSize(R.dimen.size_in_sp);
int pixels = context.getResources().getDimensionPixelSize(R.dimen.size_in_dp);

// Get the dimension specified by the resource for use as an offset.
// The value is rounded down to the nearest integer and can be 0px.
int pixels = context.getResources().getDimensionPixelOffset(R.dimen.size_in_sp);
int pixels = context.getResources().getDimensionPixelOffset(R.dimen.size_in_dp);
```

Tamaño del texto y diferentes tamaños de pantalla de Android

A veces, es mejor tener solo tres opciones

```
 style="@android:style/TextAppearance.Small"
 style="@android:style/TextAppearance.Medium"
 style="@android:style/TextAppearance.Large"
```

Use pequeñas y grandes para diferenciarse del tamaño de pantalla normal.

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="@android:style/TextAppearance.Small"/>
```

Para normal, no tienes que especificar nada.

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

Usando esto, puede evitar probar y especificar dimensiones para diferentes tamaños de pantalla.

Lea Pantallas de apoyo con diferentes resoluciones, tamaños en línea: https://riptutorial.com/es/android/topic/1086/pantallas-de-apoyo-con-diferentes-resoluciones--tamanos

Capítulo 183: Parcelable

Introducción

Parcelable es una interfaz específica de Android en la que implementa la serialización usted mismo. Fue creado para ser mucho más eficiente que Serializable y para solucionar algunos problemas con el esquema de serialización Java predeterminado.

Observaciones

Es importante recordar que el orden en el que escribe los campos en una Parcela DEBE SER EL MISMO PEDIDO de que los lea del paquete al construir su objeto personalizado.

La interfaz parcelable tiene un límite estricto de tamaño de 1 MB. Eso significa que cualquier objeto o combinación de objetos que coloque en una parcela que ocupe más de 1 MB de espacio se corromperá en el otro lado. Esto puede ser difícil de descubrir, así que tenga en cuenta qué tipo de objetos planea hacer parcelables. Si tienen árboles de dependencia grandes, considere otra manera de pasar los datos.

Examples

Haciendo un objeto personalizado parcelable.

```
/**
 * Created by Alex Sullivan on 7/21/16.
 */
public class Foo implements Parcelable
{
    private final int myFirstVariable;
    private final String mySecondVariable;
    private final long myThirdVariable;

    public Foo(int myFirstVariable, String mySecondVariable, long myThirdVariable)
    {
        this.myFirstVariable = myFirstVariable;
        this.mySecondVariable = mySecondVariable;
        this.myThirdVariable = myThirdVariable;
    }

    // Note that you MUST read values from the parcel IN THE SAME ORDER that
    // values were WRITTEN to the parcel! This method is our own custom method
    // to instantiate our object from a Parcel. It is used in the Parcelable.Creator variable
we declare below.
    public Foo(Parcel in)
    {
        this.myFirstVariable = in.readInt();
        this.mySecondVariable = in.readString();
        this.myThirdVariable = in.readLong();
    }
```

```
    // The describe contents method can normally return 0. It's used when
    // the parceled object includes a file descriptor.
    @Override
    public int describeContents()
    {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags)
    {
        dest.writeInt(myFirstVariable);
        dest.writeString(mySecondVariable);
        dest.writeLong(myThirdVariable);
    }

    // Note that this seemingly random field IS NOT OPTIONAL. The system will
    // look for this variable using reflection in order to instantiate your
    // parceled object when read from an Intent.
    public static final Parcelable.Creator<Foo> CREATOR = new Parcelable.Creator<Foo>()
    {
        // This method is used to actually instantiate our custom object
        // from the Parcel. Convention dictates we make a new constructor that
        // takes the parcel in as its only argument.
        public Foo createFromParcel(Parcel in)
        {
            return new Foo(in);
        }

        // This method is used to make an array of your custom object.
        // Declaring a new array with the provided size is usually enough.
        public Foo[] newArray(int size)
        {
            return new Foo[size];
        }
    };
}
```

Objeto parcelable que contiene otro objeto parcelable

Un ejemplo de una clase que contiene una clase parcelable dentro de:

```
public class Repository implements Parcelable {
    private String name;
    private Owner owner;
    private boolean isPrivate;

    public Repository(String name, Owner owner, boolean isPrivate) {
        this.name = name;
        this.owner = owner;
        this.isPrivate = isPrivate;
    }

    protected Repository(Parcel in) {
        name = in.readString();
        owner = in.readParcelable(Owner.class.getClassLoader());
        isPrivate = in.readByte() != 0;
    }
```

```
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeParcelable(owner, flags);
        dest.writeByte((byte) (isPrivate ? 1 : 0));
    }

    @Override
    public int describeContents() {
        return 0;
    }

    public static final Creator<Repository> CREATOR = new Creator<Repository>() {
        @Override
        public Repository createFromParcel(Parcel in) {
            return new Repository(in);
        }

        @Override
        public Repository[] newArray(int size) {
            return new Repository[size];
        }
    };

    //getters and setters

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Owner getOwner() {
        return owner;
    }

    public void setOwner(Owner owner) {
        this.owner = owner;
    }

     public boolean isPrivate() {
        return isPrivate;
    }

    public void setPrivate(boolean isPrivate) {
        this.isPrivate = isPrivate;
    }
}
```

Propietario es solo una clase parcelable normal.

Usando Enums con Parcelable

```
/**
 * Created by Nick Cardoso on 03/08/16.
 * This is not a complete parcelable implementation, it only highlights the easiest
 * way to read and write your Enum values to your parcel
```

```
 */
public class Foo implements Parcelable {

    private final MyEnum myEnumVariable;
    private final MyEnum mySaferEnumVariableExample;

    public Foo(Parcel in) {

        //the simplest way
        myEnumVariable = MyEnum.valueOf( in.readString() );

        //with some error checking
        try {
            mySaferEnumVariableExample= MyEnum.valueOf( in.readString() );
        } catch (IllegalArgumentException e) { //bad string or null value
            mySaferEnumVariableExample= MyEnum.DEFAULT;
        }

    }

    ...

    @Override
    public void writeToParcel(Parcel dest, int flags) {

        //the simple way
        dest.writeString(myEnumVariable.name());

        //avoiding NPEs with some error checking
        dest.writeString(mySaferEnumVariableExample == null? null :
mySaferEnumVariableExample.name());

    }

}

public enum MyEnum {
    VALUE_1,
    VALUE_2,
    DEFAULT
}
```

Esto es preferible a (por ejemplo) usar un ordinal, porque insertar nuevos valores en su enumeración no afectará los valores almacenados previamente

Lea Parcelable en línea: https://riptutorial.com/es/android/topic/1849/parcelable

Capítulo 184: Patrones de diseño

Introducción

Los patrones de diseño se formalizan según las mejores prácticas que el programador puede usar para resolver problemas comunes al diseñar una aplicación o sistema.

Los patrones de diseño pueden acelerar el proceso de desarrollo al proporcionar paradigmas de desarrollo probados y comprobados.

La reutilización de los patrones de diseño ayuda a prevenir problemas sutiles que pueden causar problemas mayores, y también mejora la legibilidad del código para los programadores y arquitectos que están familiarizados con los patrones.

Examples

Ejemplo de clase Singleton

Patrón de Java Singleton

Para implementar el patrón Singleton, tenemos diferentes enfoques, pero todos ellos tienen los siguientes conceptos comunes.

- Constructor privado para restringir la creación de instancias de la clase de otras clases.
- Variable estática privada de la misma clase que es la única instancia de la clase.
- Método estático público que devuelve la instancia de la clase, este es el acceso global
- punto para el mundo exterior para obtener la instancia de la clase singleton.

```
/**
 * Singleton class.
 */
public final class Singleton {

  /**
   * Private constructor so nobody can instantiate the class.
   */
  private Singleton() {}

  /**
   * Static to class instance of the class.
   */
  private static final Singleton INSTANCE = new Singleton();

  /**
   * To be called by user to obtain instance of the class.
   *
   * @return instance of the singleton.
   */
  public static Singleton getInstance() {
    return INSTANCE;
```

```
    }
}
```

Patrón observador

El patrón de observador es un patrón común, que se usa ampliamente en muchos contextos. Se puede tomar un ejemplo real de YouTube: cuando te gusta un canal y deseas recibir todas las noticias y ver nuevos videos de este canal, debes suscribirte a ese canal. Luego, cada vez que este canal publique alguna noticia, usted (y todos los demás suscriptores) recibirán una notificación.

Un observador tendrá dos componentes. Una es una emisora (canal) y la otra es un receptor (usted o cualquier otro suscriptor). La emisora manejará todas las instancias receptoras que se hayan suscrito. Cuando la emisora dispare un nuevo evento, lo anunciará a todas las instancias del receptor. Cuando el receptor recibe un evento, tendrá que reaccionar a ese evento, por ejemplo, encendiendo YouTube y reproduciendo el nuevo video.

Implementando el patrón observador.

1. La emisora debe proporcionar métodos que permitan a los receptores suscribirse y darse de baja. Cuando la emisora dispara un evento, los suscriptores deben ser notificados de que se ha producido un evento:

```
class Channel{
    private List<Subscriber> subscribers;
    public void subscribe(Subscriber sub) {
        // Add new subscriber.
    }
    public void unsubscribe(Subscriber sub) {
        // Remove subscriber.
    }
    public void newEvent() {
        // Notification event for all subscribers.
    }
}
```

2. El receptor debe implementar un método que maneje el evento desde la emisora:

```
interface Subscriber {
    void doSubscribe(Channel channel);
    void doUnsubscribe(Channel channel);
    void handleEvent();  // Process the new event.
}
```

Lea Patrones de diseño en línea: https://riptutorial.com/es/android/topic/9949/patrones-de-diseno

Capítulo 185: Pérdidas de memoria

Examples

Fugas de memoria comunes y cómo solucionarlos.

1. Arregla tus contextos:

Intente usar el contexto apropiado: por ejemplo, dado que se puede ver un Toast en muchas actividades en lugar de en solo una, use `getApplicationContext()` para `getApplicationContext()` , y como los servicios pueden seguir ejecutándose aunque una actividad haya terminado, inicie un servicio con:

```
Intent myService = new Intent(getApplicationContext(), MyService.class);
```

Use esta tabla como una guía rápida para el contexto apropiado:

	Application	Activity	Service	ContentProvider	Bro
Show a Dialog	NO	YES	NO	NO	
Start an Activity	NO[1]	YES	NO[1]	NO[1]	
Layout Inflation	NO[2]	YES	NO[2]	NO[2]	
Start a Service	YES	YES	YES	YES	
Bind to a Service	YES	YES	YES	YES	
Send a Broadcast	YES	YES	YES	YES	
Register BroadcastReceiver	YES	YES	YES	YES	
Load Resource Values	YES	YES	YES	YES	

Artículo original sobre contexto aquí .

2. Referencia estática al contexto.

Un error grave de pérdida de memoria es mantener una referencia estática a `View` . Cada `View` tiene una referencia interna al `Context` . Lo que significa que una actividad antigua con toda su jerarquía de vistas no se recogerá como basura hasta que la aplicación finalice. Tendrá su aplicación dos veces en la memoria cuando gire la pantalla.

Asegúrese de que no haya ninguna referencia estática a Vista, Contexto o cualquiera de sus descendientes.

3. Comprueba que realmente estás terminando tus servicios.

Por ejemplo, tengo un intentService que utiliza la API del servicio de ubicación de Google. Y me olvidé de llamar a `googleApiClient.disconnect();` :

```
//Disconnect from API onDestroy()
if (googleApiClient.isConnected()) {
    LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient,
GoogleLocationService.this);
    googleApiClient.disconnect();
}
```

4. Comprobar el uso de la imagen y de los mapas de bits:

Si está utilizando la **biblioteca** de Square, **Picasso** , descubrí que estaba perdiendo memoria al no usar `.fit()` , lo que redujo drásticamente mi huella de memoria de 50 MB en promedio a menos de 19 MB:

```
Picasso.with(ActivityExample.this)                    //Activity context
                .load(object.getImageUrl())
                .fit()                                 //This avoided the OutOfMemoryError
                .centerCrop()                          //makes image to not stretch
                .into(imageView);
```

5. Si está utilizando receptores de difusión, anúltelos.

6. Si está utilizando `java.util.Observer` (patrón de observador):

Asegúrese de utilizar `deleteObserver(observer);`

Evite las actividades con fugas con AsyncTask

> **Una advertencia** : AsyncTask tiene muchos errores aparte de la pérdida de memoria que se describe aquí. Así que tenga cuidado con esta API, o evítela por completo si no comprende completamente las implicaciones. Hay muchas alternativas (Thread, EventBus, RxAndroid, etc).

Un error común con `AsyncTask` es capturar una referencia fuerte a la `Activity` del host (o `Fragment`):

```
class MyActivity extends Activity {
  private AsyncTask<Void, Void, Void> myTask = new AsyncTask<Void, Void, Void>() {
    // Don't do this! Inner classes implicitly keep a pointer to their
    // parent, which in this case is the Activity!
  }
}
```

Esto es un problema porque `AsyncTask` puede sobrevivir fácilmente a la `Activity` principal, por ejemplo, si ocurre un cambio de configuración mientras se ejecuta la tarea.

La forma correcta de hacer esto es convertir su tarea en una clase `static` , que no capture al padre, y manteniendo una referencia débil a la `Activity` host:

```
class MyActivity extends Activity {
  static class MyTask extends AsyncTask<Void, Void, Void> {
    // Weak references will still allow the Activity to be garbage-collected
    private final WeakReference<MyActivity> weakActivity;

    MyTask(MyActivity myActivity) {
      this.weakActivity = new WeakReference<>(myActivity);
    }

    @Override
    public Void doInBackground(Void... params) {
      // do async stuff here
    }

    @Override
    public void onPostExecute(Void result) {
      // Re-acquire a strong reference to the activity, and verify
      // that it still exists and is active.
      MyActivity activity = weakActivity.get();
      if (activity == null
          || activity.isFinishing()
          || activity.isDestroyed()) {
        // activity is no longer valid, don't do anything!
        return;
      }

      // The activity is still valid, do main-thread stuff here
    }
  }
}
```

Callback anónimo en actividades

Cada vez que creas una clase anónima, conserva una referencia implícita a su clase principal. Así que cuando escribes:

```
public class LeakyActivity extends Activity
{

...

    foo.registerCallback(new BarCallback()
    {
        @Override
        public void onBar()
        {
            // do something
        }
    });
}
```

De hecho, está enviando una referencia a su instancia de LeakyActivity a foo. Cuando el usuario navega fuera de su LeakyActivity, esta referencia puede evitar que la instancia de LeakyActivity se recoja. Esta es una fuga grave ya que las actividades contienen una referencia a toda su

jerarquía de vistas y, por lo tanto, son objetos bastante grandes en la memoria.

Cómo evitar esta fuga:

Por supuesto, puede evitar el uso de devoluciones de llamada anónimas en actividades por completo. También puede anular el registro de todas sus devoluciones de llamada con respecto al ciclo de vida de la actividad. al igual que:

```
public class NonLeakyActivity extends Activity
{
    private final BarCallback mBarCallback = new BarCallback()
    {
        @Override
        public void onBar()
        {
            // do something
        }
    });

    @Override
    protected void onResume()
    {
        super.onResume();
        foo.registerCallback(mBarCallback);
    }

    @Override
    protected void onPause()
    {
        super.onPause();
        foo.unregisterCallback(mBarCallback);
    }
}
```

Contexto de actividad en clases estáticas

A menudo querrá envolver algunas clases de Android en clases de utilidad más fáciles de usar. Esas clases de utilidad a menudo requieren un contexto para acceder al sistema operativo Android o los recursos de sus aplicaciones. Un ejemplo común de esto es un contenedor para la clase SharedPreferences. Para acceder a las preferencias compartidas de los androides se debe escribir:

```
context.getSharedPreferences(prefsName, mode);
```

Y así uno puede estar tentado de crear la siguiente clase:

```
public class LeakySharedPrefsWrapper
{
    private static Context sContext;

    public static void init(Context context)
    {
        sContext = context;
    }

    public int getInt(String name,int defValue)
```

```
    {
        return sContext.getSharedPreferences("a name",
Context.MODE_PRIVATE).getInt(name,defValue);
    }
}
```

Ahora, si llama a `init()` con su contexto de actividad, el LeakySharedPrefsWrapper conservará una referencia a su actividad, evitando que se recoja basura.

Como evitar:

Al llamar a las funciones de ayuda estática, puede enviar el contexto de la aplicación utilizando `context.getApplicationContext();`

Al crear funciones auxiliares estáticas, puede extraer el contexto de la aplicación del contexto que se le da (Al llamar a getApplicationContext () en el contexto de la aplicación, se devuelve el contexto de la aplicación). Así que la solución a nuestra envoltura es simple:

```
public static void init(Context context)
{
    sContext = context.getApplicationContext();
}
```

Si el contexto de la aplicación no es apropiado para su caso de uso, puede incluir un parámetro de contexto en cada función de utilidad, debe evitar mantener referencias a estos parámetros de contexto. En este caso la solución se vería así:

```
public int getInt(Context context,String name,int defValue)
{
    // do not keep a reference of context to avoid potential leaks.
    return context.getSharedPreferences("a name", Context.MODE_PRIVATE).getInt(name,defValue);
}
```

Detecta pérdidas de memoria con la biblioteca LeakCanary

LeakCanary es una biblioteca de código abierto de Java para detectar pérdidas de memoria en sus construcciones de depuración.

Solo agrega las dependencias en el `build.gradle` :

```
 dependencies {
   debugCompile 'com.squareup.leakcanary:leakcanary-android:1.5.1'
   releaseCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
   testCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
 }
```

Luego en tu clase de `Application` :

```
public class ExampleApplication extends Application {

  @Override public void onCreate() {
    super.onCreate();
```

```
    if (LeakCanary.isInAnalyzerProcess(this)) {
      // This process is dedicated to LeakCanary for heap analysis.
      // You should not init your app in this process.
      return;
    }

    LeakCanary.install(this);
  }
}
```

Ahora LeakCanary mostrará automáticamente una notificación cuando se detecte una pérdida de memoria de actividad en su compilación de **depuración** .

NOTA: El código de publicación no contendrá ninguna referencia a LeakCanary más que las dos clases vacías que existen en la `leakcanary-android-no-op` .

Evite las actividades de filtración con oyentes

Si implementa o crea un escucha en una actividad, siempre preste atención al ciclo de vida del objeto que tiene el escucha registrado.

Considere una aplicación en la que tengamos varias actividades / fragmentos diferentes interesados en cuando un usuario está conectado o desconectado. Una forma de hacer esto sería tener una instancia singleton de un `UserController` que se pueda suscribir para recibir una notificación cuando cambie el estado del usuario:

```
public class UserController {
    private static UserController instance;
    private List<StateListener> listeners;

    public static synchronized UserController getInstance() {
        if (instance == null) {
            instance = new UserController();
        }
        return instance;
    }

    private UserController() {
        // Init
    }

    public void registerUserStateChangeListener(StateListener listener) {
        listeners.add(listener);
    }

    public void logout() {
        for (StateListener listener : listeners) {
            listener.userLoggedOut();
        }
    }

    public void login() {
        for (StateListener listener : listeners) {
            listener.userLoggedIn();
        }
```

```
    }

    public interface StateListener {
        void userLoggedIn();
        void userLoggedOut();
    }
}
```

Luego hay dos actividades, `SignInActivity` :

```
public class SignInActivity extends Activity implements UserController.StateListener{

    UserController userController;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        this.userController = UserController.getInstance();
        this.userController.registerUserStateChangeListener(this);
    }

    @Override
    public void userLoggedIn() {
        startMainActivity();
    }

    @Override
    public void userLoggedOut() {
        showLoginForm();
    }

    ...

    public void onLoginClicked(View v) {
        userController.login();
    }
}
```

Y `MainActivity` :

```
public class MainActivity extends Activity implements UserController.StateListener{
    UserController userController;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        this.userController = UserController.getInstance();
        this.userController.registerUserStateChangeListener(this);
    }

    @Override
    public void userLoggedIn() {
        showUserAccount();
    }

    @Override
    public void userLoggedOut() {
```

```
        finish();
    }

    ...

    public void onLogoutClicked(View v) {
        userController.logout();
    }
}
```

Lo que sucede con este ejemplo es que cada vez que el usuario inicia sesión y luego vuelve a `MainActivity` sesión, se `MainActivity` una instancia de `MainActivity` . La fuga se produce porque hay una referencia a la actividad en los `UserController#listeners` .

Tenga en cuenta: incluso si utilizamos una clase interna anónima como un oyente, la actividad todavía se filtraría:

```
...
this.userController.registerUserStateChangeListener(new UserController.StateListener() {
    @Override
    public void userLoggedIn() {
        showUserAccount();
    }

    @Override
    public void userLoggedOut() {
        finish();
    }
});
...
```

La actividad todavía se filtraría, porque la clase interna anónima tiene una referencia implícita a la clase externa (en este caso, la actividad). Es por esto que es posible llamar a los métodos de instancia en la clase externa desde la clase interna. De hecho, el único tipo de clases internas que no tienen una referencia a la clase externa son las clases internas **estáticas** .

En resumen, todas las instancias de clases internas no estáticas contienen una referencia implícita a la instancia de la clase externa que las creó.

Hay dos enfoques principales para resolver esto, ya sea agregando un método para eliminar un oyente de los auditores de `UserController#listeners` o usando una `WeakReference` para mantener la referencia de los oyentes.

Alternativa 1: Eliminar oyentes

Comencemos por crear un nuevo método `removeUserStateChangeListener(StateListener listener)` :

```
public class UserController {

    ...

    public void registerUserStateChangeListener(StateListener listener) {
        listeners.add(listener);
```

```
    }

    public void removeUserStateChangeListener(StateListener listener) {
        listeners.remove(listener);
    }

    ...
}
```

Entonces llamemos a este método en el método `onDestroy` la actividad:

```
public class MainActivity extends Activity implements UserController.StateListener{
    ...

    @Override
    protected void onDestroy() {
        super.onDestroy();
        userController.removeUserStateChangeListener(this);
    }
}
```

Con esta modificación, las instancias de `MainActivity` ya no se filtran cuando el usuario inicia y `MainActivity` sesión. Sin embargo, si la documentación no está clara, es probable que el próximo desarrollador que comience a usar `UserController` se pierda la obligación de anular el registro del oyente cuando se destruye la actividad, lo que nos lleva al segundo método de evitar este tipo de fugas.

Alternativa 2: Usar referencias débiles

En primer lugar, comencemos por explicar qué es una referencia débil. Una referencia débil, como su nombre indica, contiene una referencia débil a un objeto. En comparación con un campo de instancia normal, que es una referencia fuerte, una referencia débil no impide que el recolector de basura, GC, elimine los objetos. En el ejemplo anterior, esto permitiría que `MainActivity` se recoja después de que se haya destruido si el `UserController` usó `WeakReference` para hacer referencia a los oyentes.

En resumen, una referencia débil le dice al GC que si nadie más tiene una referencia fuerte a este objeto, siga adelante y elimínelo.

Permítanos modificar el `UserController` para usar una lista de `WeakReference` para hacer un seguimiento de sus oyentes:

```
public class UserController {

    ...
    private List<WeakReference<StateListener>> listeners;
    ...

    public void registerUserStateChangeListener(StateListener listener) {
        listeners.add(new WeakReference<>(listener));
    }

    public void removeUserStateChangeListener(StateListener listenerToRemove) {
```

```
        WeakReference referencesToRemove = null;
        for (WeakReference<StateListener> listenerRef : listeners) {
            StateListener listener = listenerRef.get();
            if (listener != null && listener == listenerToRemove) {
                referencesToRemove = listenerRef;
                break;
            }
        }
        listeners.remove(referencesToRemove);
    }

    public void logout() {
        List referencesToRemove = new LinkedList();
        for (WeakReference<StateListener> listenerRef : listeners) {
            StateListener listener = listenerRef.get();
            if (listener != null) {
                listener.userLoggedOut();
            } else {
                referencesToRemove.add(listenerRef);
            }
        }
    }

    public void login() {
        List referencesToRemove = new LinkedList();
        for (WeakReference<StateListener> listenerRef : listeners) {
            StateListener listener = listenerRef.get();
            if (listener != null) {
                listener.userLoggedIn();
            } else {
                referencesToRemove.add(listenerRef);
            }
        }
    }
    ...
}
```

Con esta modificación, no importa si se eliminan o no los escuchas, ya que `UserController` no tiene referencias sólidas a ninguno de los oyentes. Sin embargo, escribir este código repetitivo cada vez es engorroso. Por lo tanto, vamos a crear una clase genérica llamada `WeakCollection` :

```
public class WeakCollection<T> {
    private LinkedList<WeakReference<T>> list;

    public WeakCollection() {
        this.list = new LinkedList<>();
    }
    public void put(T item){
        //Make sure that we don't re add an item if we already have the reference.
        List<T> currentList = get();
        for(T oldItem : currentList){
            if(item == oldItem){
                return;
            }
        }
        list.add(new WeakReference<T>(item));
    }

    public List<T> get() {
```

```
        List<T> ret = new ArrayList<>(list.size());
        List<WeakReference<T>> itemsToRemove = new LinkedList<>();
        for (WeakReference<T> ref : list) {
            T item = ref.get();
            if (item == null) {
                itemsToRemove.add(ref);
            } else {
                ret.add(item);
            }
        }
        for (WeakReference ref : itemsToRemove) {
            this.list.remove(ref);
        }
        return ret;
    }

    public void remove(T listener) {
        WeakReference<T> refToRemove = null;
        for (WeakReference<T> ref : list) {
            T item = ref.get();
            if (item == listener) {
                refToRemove = ref;
            }
        }
        if(refToRemove != null){
            list.remove(refToRemove);
        }
    }
}
```

Ahora volvamos a escribir `UserController` para utilizar `WeakCollection<T>` lugar:

```
public class UserController {
    ...
    private WeakCollection<StateListener> listenerRefs;
    ...

    public void registerUserStateChangeListener(StateListener listener) {
        listenerRefs.put(listener);
    }

    public void removeUserStateChangeListener(StateListener listenerToRemove) {
        listenerRefs.remove(listenerToRemove);
    }

    public void logout() {
        for (StateListener listener : listenerRefs.get()) {
            listener.userLoggedOut();
        }
    }

    public void login() {
        for (StateListener listener : listenerRefs.get()) {
            listener.userLoggedIn();
        }
    }

    ...
}
```

Como se muestra en el ejemplo de código anterior, `WeakCollection<T>` elimina todo el código de `WeakReference` necesario para usar `WeakReference` lugar de una lista normal. Para colmo: si se pierde una llamada a `UserController#removeUserStateChangeListener(StateListener)` , el oyente y todos los objetos a los que hace referencia no se perderán.

Evite las pérdidas de memoria con la clase anónima, el controlador, la tarea del temporizador y el hilo

En Android, todos los desarrolladores utilizan `Anonymous Class` (Runnable) al menos una vez en un proyecto. Cualquier `Anonymous Class` tiene una referencia a su padre (actividad). Si realizamos una tarea de larga duración, la actividad principal no se destruirá hasta que la tarea finalice.
El ejemplo usa el controlador y la clase `Runnable` Anónimo. La memoria se perderá cuando abandonemos la actividad antes de que `Runnable` .

```
new Handler().postDelayed(new Runnable() {
      @Override
      public void run() {
          // do abc long 5s or so
      }
  }, 10000); // run "do abc" after 10s. It same as timer, thread...
```

¿Como lo resolvemos?

1. No hagas operaciones largas con la `Anonymous Class` o necesitamos una `Static class` para ello y le pasamos `WeakReference` (como actividad, vista ...). `Thread` es el mismo con la `Anonymous Class` .
2. Cancelar el `Handler` , `Timer` cuando se destruye la actividad.

Lea Pérdidas de memoria en línea: https://riptutorial.com/es/android/topic/2687/perdidas-de-memoria

Capítulo 186: Permisos de tiempo de ejecución en API-23 +

Introducción

Android Marshmallow introdujo el modelo Runtime Permission . Los permisos se clasifican en dos categorías, es decir, permisos normales y peligrosos . donde ahora el usuario otorga permisos peligrosos en tiempo de ejecución.

Observaciones

Desde el SDK 23, Android requiere permisos de tiempo de ejecución para permisos en dispositivos que ejecutan Android 6.0 y superior, dentro de lo que se clasifica como los Grupos de permisos peligrosos. Los grupos de permisos peligrosos son aquellos que se considera que comprometen la privacidad y / o seguridad del usuario.

La siguiente es una lista de grupos de permisos peligrosos:

Grupos de permisos peligrosos

Grupo de permisos
CALENDARIO
CÁMARA
CONTACTOS
UBICACIÓN
MICRÓFONO
TELÉFONO
Sensores
SMS
ALMACENAMIENTO

Cualquier permiso de estos grupos requiere la administración de permisos de tiempo de ejecución para dispositivos en Android 6.0 y superior con un sdk de destino de 23 o superior.

Permisos normales

La siguiente es una lista de permisos normales. Estos no se consideran peligrosos para la privacidad o seguridad del usuario, por lo que no requieren permisos de tiempo de ejecución para sdk 23 y superior.

ACCESS_LOCATION_EXTRA_COMMANDS
ACCESS_NETWORK_STATE
ACCESS_NOTIFICATION_POLICY
ACCESS_WIFI_STATE
BLUETOOTH

BLUETOOTH_ADMIN
BROADCAST_STICKY
CHANGE_NETWORK_STATE
CHANGE_WIFI_MULTICAST_STATE
CHANGE_WIFI_STATE
DISABLE_KEYGUARD
EXPAND_STATUS_BAR
GET_PACKAGE_SIZE
INSTALL_SHORTCUT
INTERNET
KILL_BACKGROUND_PROCESSES
MODIFY_AUDIO_SETTINGS
NFC
READ_SYNC_SETTINGS
READ_SYNC_STATS
RECEIVE_BOOT_COMPLETED
REORDER_TASKS
REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
REQUEST_INSTALL_PACKAGES
AJUSTAR ALARMA
SET_TIME_ZONE
ESTABLECER FONDO DE PANTALLA
SET_WALLPAPER_HINTS
TRANSMIT_IR
UNINSTALL_SHORTCUT
USE_FINGERPRINT
VIBRAR
WAKE_LOCK
WRITE_SYNC_SETTINGS

Examples

Android 6.0 permisos múltiples

Este ejemplo muestra cómo verificar los permisos en tiempo de ejecución en Android 6 y versiones posteriores.

```
public static final int MULTIPLE_PERMISSIONS = 10; // code you want.

String[] permissions = new String[] {
    Manifest.permission.WRITE_EXTERNAL_STORAGE,
    Manifest.permission.CAMERA,
    Manifest.permission.ACCESS_COARSE_LOCATION,
    Manifest.permission.ACCESS_FINE_LOCATION
};

@Override
void onStart() {
    if (checkPermissions()){
```

```
        // permissions granted.
    } else {
        // show dialog informing them that we lack certain permissions
    }
}

private boolean checkPermissions() {
    int result;
    List<String> listPermissionsNeeded = new ArrayList<>();
    for (String p:permissions) {
        result = ContextCompat.checkSelfPermission(getActivity(),p);
        if (result != PackageManager.PERMISSION_GRANTED) {
            listPermissionsNeeded.add(p);
        }
    }
    if (!listPermissionsNeeded.isEmpty()) {
        ActivityCompat.requestPermissions(this, listPermissionsNeeded.toArray(new
String[listPermissionsNeeded.size()]), MULTIPLE_PERMISSIONS);
        return false;
    }
    return true;
}

@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[]
grantResults) {
    switch (requestCode) {
        case MULTIPLE_PERMISSIONS:{
            if(grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED){
                // permissions granted.
            } else {
                // no permissions granted.
            }
            return;
        }
    }
}
```

Cumplimiento de permisos en difusiones, URI

Puede realizar una verificación de permisos al enviar un Intent a un receptor de difusión registrado. Los permisos que envía se cotejan con los registrados bajo la etiqueta. Restringen quién puede enviar transmisiones al receptor asociado.

Para enviar una solicitud de difusión con permisos, especifique el permiso como una cadena en la `Context.sendBroadcast(Intent intent, String permission)` , pero tenga en cuenta que la aplicación del receptor **DEBE** tener ese permiso para recibir su transmisión. El receptor debe instalarse primero antes que el remitente.

La firma del método es:

```
 void sendBroadcast (Intent intent, String receiverPermission)
 //for example to send a broadcast to Bcastreceiver receiver
 Intent broadcast = new Intent(this, Bcastreceiver.class);
 sendBroadcast(broadcast, "org.quadcore.mypermission");
```

y puede especificar en su manifiesto que el remitente de difusión debe incluir el permiso solicitado enviado a través de sendBroadcast:

```
<!--  Your special permission -->
<permission android:name="org.quadcore.mypermission"
   android:label="my_permission"
   android:protectionLevel="dangerous"></permission>
```

También declare el permiso en el manifiesto de la aplicación que debe recibir esta transmisión:

```
<!--  I use the permission ! -->
<uses-permission android:name="org.quadcore.mypermission"/>
<!-- along with the receiver -->
<receiver android:name="Bcastreceiver" android:exported="true" />
```

Nota: Tanto un receptor como una emisora pueden requerir un permiso, y cuando esto sucede, ambas comprobaciones de permisos deben pasar para que la Intención se entregue al destino asociado. La aplicación que define el permiso se debe instalar primero.

Encuentra la documentación completa aquí en Permisos.

Permisos de tiempo de ejecución múltiples de los mismos grupos de permisos

En el manifiesto tenemos cuatro permisos de tiempo de ejecución peligrosos de dos grupos.

```
<!-- Required to read and write to shredPref file. -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

<!-- Required to get location of device. -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

En la actividad donde se requieren los permisos. Tenga en cuenta que es importante verificar los permisos en cualquier actividad que requiera permisos, ya que los permisos se pueden revocar mientras la aplicación está en segundo plano y la aplicación se bloqueará.

```
final private int REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS = 124;


@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.act_layout);

    // A simple check of whether runtime permissions need to be managed
    if (Build.VERSION.SDK_INT >= 23) {
        checkMultiplePermissions();
    }
```

Solo necesitamos pedir permiso para uno de estos de cada grupo y todos los demás permisos de

este grupo se otorgan a menos que el usuario revoque el permiso.

```
private void checkMultiplePermissions() {

    if (Build.VERSION.SDK_INT >= 23) {
        List<String> permissionsNeeded = new ArrayList<String>();
        List<String> permissionsList = new ArrayList<String>();

        if (!addPermission(permissionsList, android.Manifest.permission.ACCESS_FINE_LOCATION))
{
            permissionsNeeded.add("GPS");
        }

        if (!addPermission(permissionsList,
android.Manifest.permission.READ_EXTERNAL_STORAGE)) {
            permissionsNeeded.add("Read Storage");
        }

        if (permissionsList.size() > 0) {
            requestPermissions(permissionsList.toArray(new String[permissionsList.size()]),
                    REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS);
            return;
        }
    }
}



private boolean addPermission(List<String> permissionsList, String permission) {
    if (Build.VERSION.SDK_INT >= 23)

        if (checkSelfPermission(permission) != PackageManager.PERMISSION_GRANTED) {
            permissionsList.add(permission);

            // Check for Rationale Option
            if (!shouldShowRequestPermissionRationale(permission))
                return false;
        }
    return true;
}
```

Se trata del resultado de que el usuario permita o no permita permisos. En este ejemplo, si los permisos no están permitidos, la aplicación se cancela.

```
@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
grantResults) {
    switch (requestCode) {
        case REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS: {

            Map<String, Integer> perms = new HashMap<String, Integer>();
            // Initial
            perms.put(android.Manifest.permission.ACCESS_FINE_LOCATION,
PackageManager.PERMISSION_GRANTED);
            perms.put(android.Manifest.permission.READ_EXTERNAL_STORAGE,
PackageManager.PERMISSION_GRANTED);

            // Fill with results
            for (int i = 0; i < permissions.length; i++)
```

```
                perms.put(permissions[i], grantResults[i]);
            if (perms.get(android.Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED
                    && perms.get(android.Manifest.permission.READ_EXTERNAL_STORAGE) ==
PackageManager.PERMISSION_GRANTED) {
                // All Permissions Granted
                return;
            } else {
                // Permission Denied
                if (Build.VERSION.SDK_INT >= 23) {
                    Toast.makeText(
                            getApplicationContext(),
                            "My App cannot run without Location and Storage " +
                                    "Permissions.\nRelaunch My App or allow permissions" +
                                    " in Applications Settings",
                            Toast.LENGTH_LONG).show();
                    finish();
                }
            }
        }
        break;
        default:
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

Más información https://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en

Usando PermissionUtil

PermissionUtil es una forma simple y conveniente de solicitar permisos en contexto. Puede proporcionar fácilmente lo que debería suceder en el caso de todos los permisos solicitados concedidos (`onAllGranted()`), cualquier solicitud fue denegada (`onAnyDenied()`) o en caso de que se necesite un racional (`onRational()`).

En cualquier lugar de su AppCompatActivity o Fragmento, desea solicitar la autorización del usuario.

```
mRequestObject =
PermissionUtil.with(this).request(Manifest.permission.WRITE_EXTERNAL_STORAGE).onAllGranted(
                new Func() {
                    @Override protected void call() {
                        //Happy Path
                    }
                }).onAnyDenied(
                new Func() {
                    @Override protected void call() {
                        //Sad Path
                    }
                }).ask(REQUEST_CODE_STORAGE);
```

Y agregar esto a `onRequestPermissionsResult`

```
if(mRequestObject!=null){
```

```
    mRequestObject.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

Agrega el permiso solicitado a tu AndroidManifest.xml también

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Incluya todo el código relacionado con permisos para una clase base abstracta y extienda la actividad de esta clase base para lograr un código más limpio / reutilizable

```
public abstract class BaseActivity extends AppCompatActivity {
    private Map<Integer, PermissionCallback> permissionCallbackMap = new HashMap<>();

    @Override
    protected void onStart() {
        super.onStart();
        ...
    }

    @Override
    public void setContentView(int layoutResId) {
        super.setContentView(layoutResId);
        bindViews();
    }

    ...

    @Override
    public void onRequestPermissionsResult(
            int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        PermissionCallback callback = permissionCallbackMap.get(requestCode);

        if (callback == null) return;

        // Check whether the permission request was rejected.
        if (grantResults.length < 0 && permissions.length > 0) {
            callback.onPermissionDenied(permissions);
            return;
        }

        List<String> grantedPermissions = new ArrayList<>();
        List<String> blockedPermissions = new ArrayList<>();
        List<String> deniedPermissions = new ArrayList<>();
        int index = 0;

        for (String permission : permissions) {
            List<String> permissionList = grantResults[index] ==
PackageManager.PERMISSION_GRANTED
                    ? grantedPermissions
                    : ! ActivityCompat.shouldShowRequestPermissionRationale(this, permission)
                        ? blockedPermissions
                        : deniedPermissions;
            permissionList.add(permission);
            index ++;
        }
```

```
        if (grantedPermissions.size() > 0) {
            callback.onPermissionGranted(
                    grantedPermissions.toArray(new String[grantedPermissions.size()]));
        }

        if (deniedPermissions.size() > 0) {
            callback.onPermissionDenied(
                    deniedPermissions.toArray(new String[deniedPermissions.size()]));
        }

        if (blockedPermissions.size() > 0) {
            callback.onPermissionBlocked(
                    blockedPermissions.toArray(new String[blockedPermissions.size()]));
        }

        permissionCallbackMap.remove(requestCode);
    }

    /**
     * Check whether a permission is granted or not.
     *
     * @param permission
     * @return
     */
    public boolean hasPermission(String permission) {
        return ContextCompat.checkSelfPermission(this, permission) ==
PackageManager.PERMISSION_GRANTED;
    }

    /**
     * Request permissions and get the result on callback.
     *
     * @param permissions
     * @param callback
     */
    public void requestPermission(String [] permissions, @NonNull PermissionCallback callback)
{
        int requestCode = permissionCallbackMap.size() + 1;
        permissionCallbackMap.put(requestCode, callback);
        ActivityCompat.requestPermissions(this, permissions, requestCode);
    }

    /**
     * Request permission and get the result on callback.
     *
     * @param permission
     * @param callback
     */
    public void requestPermission(String permission, @NonNull PermissionCallback callback) {
        int requestCode = permissionCallbackMap.size() + 1;
        permissionCallbackMap.put(requestCode, callback);
        ActivityCompat.requestPermissions(this, new String[] { permission }, requestCode);
    }
}
```

Ejemplo de uso en la actividad.

La actividad debe ampliar la clase base abstracta definida anteriormente de la siguiente manera:

```
private void requestLocationAfterPermissionCheck() {
    if (hasPermission(Manifest.permission.ACCESS_FINE_LOCATION)) {
        requestLocation();
        return;
    }

    // Call the base class method.
    requestPermission(Manifest.permission.ACCESS_FINE_LOCATION, new PermissionCallback() {
        @Override
        public void onPermissionGranted(String[] grantedPermissions) {
            requestLocation();
        }

        @Override
        public void onPermissionDenied(String[] deniedPermissions) {
            // Do something.
        }

        @Override
        public void onPermissionBlocked(String[] blockedPermissions) {
            // Do something.
        }
    });
}
```

Lea Permisos de tiempo de ejecución en API-23 + en línea:
https://riptutorial.com/es/android/topic/1525/permisos-de-tiempo-de-ejecucion-en-api-23-plus

Capítulo 187: Picasso

Introducción

Picasso es una librería de imágenes para Android. Es creado y mantenido por Square . Simplifica el proceso de visualización de imágenes desde ubicaciones externas. La biblioteca maneja cada etapa del proceso, desde la solicitud HTTP inicial hasta el almacenamiento en caché de la imagen. En muchos casos, solo se requieren unas pocas líneas de código para implementar esta biblioteca ordenada.

Observaciones

Picasso es una potente biblioteca de descarga y almacenamiento de imágenes para Android. Sigue este ejemplo para agregar la biblioteca a tu proyecto.

Sitios web:

- Fuente
- Doc
- Registro de cambios

Examples

Añadiendo la biblioteca de Picasso a tu proyecto de Android

De la documentación oficial :

Gradle

```
dependencies {
 compile "com.squareup.picasso:picasso:2.5.2"
}
```

Maven

```
<dependency>
  <groupId>com.squareup.picasso</groupId>
  <artifactId>picasso</artifactId>
  <version>2.5.2</version>
</dependency>
```

Marcador de posición y manejo de errores

Picasso admite tanto marcadores de posición de descarga como de error como características opcionales. También proporciona devoluciones de llamada para el manejo del resultado de la descarga.

```
Picasso.with(context)
  .load("YOUR IMAGE URL HERE")
  .placeholder(Your Drawable Resource)   //this is optional the image to display while the url
image is downloading
  .error(Your Drawable Resource)         //this is also optional if some error has occurred in
downloading the image this image would be displayed
  .into(imageView, new Callback(){
     @Override
        public void onSuccess() {}

        @Override
        public void onError() {}
   });
```

Se reintentará una solicitud tres veces antes de que se muestre el marcador de posición de error.

Redimensionamiento y rotación

```
Picasso.with(context)
 .load("YOUR IMAGE URL HERE")
 .placeholder(DRAWABLE RESOURCE)   // optional
 .error(DRAWABLE RESOURCE)         // optional
 .resize(width, height)            // optional
 .rotate(degree)                   // optional
 .into(imageView);
```

Avatares circulares con Picasso.

Aquí hay un ejemplo de la clase Picasso Circle Transform basada en el original , con la adición de un borde fino, y también incluye la funcionalidad de un separador opcional para apilar:

```
import android.graphics.Bitmap;
import android.graphics.BitmapShader;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Style;

import com.squareup.picasso.Transformation;

public class CircleTransform implements Transformation {

    boolean mCircleSeparator = false;

    public CircleTransform(){
    }

    public CircleTransform(boolean circleSeparator){
        mCircleSeparator = circleSeparator;
    }
```

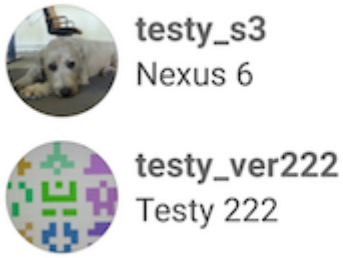
```
    @Override
    public Bitmap transform(Bitmap source) {
        int size = Math.min(source.getWidth(), source.getHeight());

        int x = (source.getWidth() - size) / 2;
        int y = (source.getHeight() - size) / 2;

        Bitmap squaredBitmap = Bitmap.createBitmap(source, x, y, size, size);

        if (squaredBitmap != source) {
            source.recycle();
        }

        Bitmap bitmap = Bitmap.createBitmap(size, size, source.getConfig());

        Canvas canvas = new Canvas(bitmap);
        BitmapShader shader = new BitmapShader(squaredBitmap, BitmapShader.TileMode.CLAMP,
BitmapShader.TileMode.CLAMP);
        Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG | Paint.DITHER_FLAG |
Paint.FILTER_BITMAP_FLAG);
        paint.setShader(shader);

        float r = size/2f;
        canvas.drawCircle(r, r, r-1, paint);

        // Make the thin border:
        Paint paintBorder = new Paint();
        paintBorder.setStyle(Style.STROKE);
        paintBorder.setColor(Color.argb(84,0,0,0));
        paintBorder.setAntiAlias(true);
        paintBorder.setStrokeWidth(1);
        canvas.drawCircle(r, r, r-1, paintBorder);

        // Optional separator for stacking:
        if (mCircleSeparator) {
            Paint paintBorderSeparator = new Paint();
            paintBorderSeparator.setStyle(Style.STROKE);
            paintBorderSeparator.setColor(Color.parseColor("#ffffff"));
            paintBorderSeparator.setAntiAlias(true);
            paintBorderSeparator.setStrokeWidth(4);
            canvas.drawCircle(r, r, r+1, paintBorderSeparator);
        }

        squaredBitmap.recycle();
        return bitmap;
    }

    @Override
    public String key() {
        return "circle";
    }
}
```

Aquí es cómo usarlo cuando se carga una imagen (suponiendo que `this` es un contexto de actividad, y la `url` es una cadena con la URL de la imagen a cargar):

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform())
```

```
    .into(ivAvatar);
```

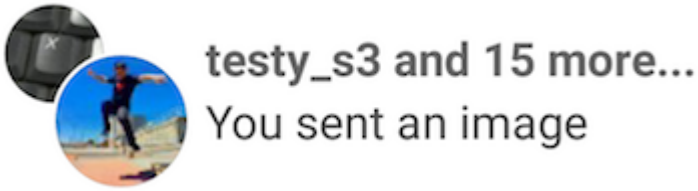
Resultado:



Para su uso con el separador, dar `true` que el constructor de la imagen superior:

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform(true))
    .into(ivAvatar);
```

Resultado (dos ImageViews en un FrameLayout):



Deshabilitar el caché en Picasso

```
Picasso.with(context)
.load(uri)
.networkPolicy(NetworkPolicy.NO_CACHE)
.memoryPolicy(MemoryPolicy.NO_CACHE)
.placeholder(R.drawable.placeholder)
.into(imageView);
```

Cargando imagen desde almacenamiento externo

```
String filename = "image.png";
String imagePath = getExternalFilesDir() + "/" + filename;

Picasso.with(context)
    .load(new File(imagePath))
    .into(imageView);
```

Descargando imagen como Bitmap usando Picasso

Si desea descargar la imagen como `Bitmap` utilizando `Picasso` siguiente código le ayudará:

```
Picasso.with(mContext)
        .load(ImageUrl)
        .into(new Target() {
            @Override
            public void onBitmapLoaded(Bitmap bitmap, Picasso.LoadedFrom from) {
               // Todo: Do something with your bitmap here
            }

            @Override
            public void onBitmapFailed(Drawable errorDrawable) {
            }

            @Override
            public void onPrepareLoad(Drawable placeHolderDrawable) {
            }
        });
```

Cancelando solicitudes de imagen usando Picasso

En ciertos casos, debemos cancelar una solicitud de descarga de imágenes en Picasso antes de que se complete la descarga.

Esto podría suceder por varios motivos, por ejemplo, si la vista principal cambia a otra vista antes de que se complete la descarga de la imagen.

En este caso, puede cancelar la solicitud de descarga de imágenes usando el método `cancelRequest()` :

```
ImageView imageView;

//......

Picasso.with(imageView.getContext()).cancelRequest(imageView);
```

Usando Picasso como ImageGetter para Html.fromHtml

Usando Picasso como ImageGetter para Html.fromHtml

```
public class PicassoImageGetter implements Html.ImageGetter {

private TextView textView;

private Picasso picasso;

public PicassoImageGetter(@NonNull Picasso picasso, @NonNull TextView textView) {
    this.picasso = picasso;
    this.textView = textView;
}

@Override
public Drawable getDrawable(String source) {
    Log.d(PicassoImageGetter.class.getName(), "Start loading url " + source);

    BitmapDrawablePlaceHolder drawable = new BitmapDrawablePlaceHolder();
```

```
    picasso
            .load(source)
            .error(R.drawable.connection_error)
            .into(drawable);

    return drawable;
}

private class BitmapDrawablePlaceHolder extends BitmapDrawable implements Target {

    protected Drawable drawable;

    @Override
    public void draw(final Canvas canvas) {
        if (drawable != null) {
            checkBounds();
            drawable.draw(canvas);
        }
    }

    public void setDrawable(@Nullable Drawable drawable) {
        if (drawable != null) {
            this.drawable = drawable;
            checkBounds();
        }
    }

    private void checkBounds() {
        float defaultProportion = (float) drawable.getIntrinsicWidth() / (float)
drawable.getIntrinsicHeight();
        int width = Math.min(textView.getWidth(), drawable.getIntrinsicWidth());
        int height = (int) ((float) width / defaultProportion);

        if (getBounds().right != textView.getWidth() || getBounds().bottom != height) {

            setBounds(0, 0, textView.getWidth(), height); //set to full width

            int halfOfPlaceHolderWidth = (int) ((float) getBounds().right / 2f);
            int halfOfImageWidth = (int) ((float) width / 2f);

            drawable.setBounds(
                    halfOfPlaceHolderWidth - halfOfImageWidth, //centering an image
                    0,
                    halfOfPlaceHolderWidth + halfOfImageWidth,
                    height);

            textView.setText(textView.getText()); //refresh text
        }
    }

    //------------------------------------------------------------------//

    @Override
    public void onBitmapLoaded(Bitmap bitmap, Picasso.LoadedFrom from) {
        setDrawable(new BitmapDrawable(Application.getContext().getResources(), bitmap));
    }

    @Override
    public void onBitmapFailed(Drawable errorDrawable) {
        setDrawable(errorDrawable);
    }
```

```
    @Override
    public void onPrepareLoad(Drawable placeHolderDrawable) {
        setDrawable(placeHolderDrawable);
    }

    //------------------------------------------------------------------//

}
}
```

El uso es simple:

```
Html.fromHtml(textToParse, new PicassoImageGetter(picasso, textViewTarget), null);
```

Pruebe primero la memoria caché del disco sin conexión, luego conéctese y busque la imagen

Primero agregue el OkHttp al archivo de compilación gradle del módulo de la aplicación

```
compile 'com.squareup.picasso:picasso:2.5.2'
compile 'com.squareup.okhttp:okhttp:2.4.0'
compile 'com.jakewharton.picasso:picasso2-okhttp3-downloader:1.0.2'
```

Luego haz una clase extendiendo la aplicación.

```
import android.app.Application;

import com.squareup.picasso.OkHttpDownloader;
import com.squareup.picasso.Picasso;

public class Global extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        Picasso.Builder builder = new Picasso.Builder(this);
        builder.downloader(new OkHttpDownloader(this,Integer.MAX_VALUE));
        Picasso built = builder.build();
        built.setIndicatorsEnabled(true);
        built.setLoggingEnabled(true);
        Picasso.setSingletonInstance(built);

    }
}
```

Agréguelo al archivo Manifiesto de la siguiente manera:

```
<application
        android:name=".Global"
        .. >

</application>
```

Uso normal

```
Picasso.with(getActivity())
.load(imageUrl)
.networkPolicy(NetworkPolicy.OFFLINE)
.into(imageView, new Callback() {
    @Override
    public void onSuccess() {
        //Offline Cache hit
    }

    @Override
    public void onError() {
        //Try again online if cache failed
        Picasso.with(getActivity())
                .load(imageUrl)
                .error(R.drawable.header)
                .into(imageView, new Callback() {
            @Override
            public void onSuccess() {
                 //Online download
            }

            @Override
            public void onError() {
                Log.v("Picasso","Could not fetch image");
            }
        });
    }
});
```

Enlace a la respuesta original

Lea Picasso en línea: https://riptutorial.com/es/android/topic/2172/picasso

Capítulo 188: Ping ICMP

Introducción

La solicitud de ping de ICMP se puede realizar en Android creando un nuevo proceso para ejecutar la solicitud de ping. El resultado de la solicitud puede evaluarse al completar la solicitud de ping desde su proceso.

Examples

Realiza un solo ping.

Este ejemplo intenta una única solicitud de ping. El comando ping dentro de la `runtime.exec` método `runtime.exec` se puede modificar a cualquier comando ping válido que pueda realizar usted mismo en la línea de comandos.

```
try {
    Process ipProcess = runtime.exec("/system/bin/ping -c 1 8.8.8.8");
    int exitValue = ipProcess.waitFor();
    ipProcess.destroy();

    if(exitValue == 0){
        // Success
    } else {
        // Failure
    }
} catch (IOException | InterruptedException e) {
    e.printStackTrace();
}
```

Lea Ping ICMP en línea: https://riptutorial.com/es/android/topic/9434/ping-icmp

Capítulo 189: Pintar

Introducción

Una pintura es uno de los cuatro objetos necesarios para dibujar, junto con un Lienzo (contiene llamadas de dibujo), un Mapa de bits (contiene los píxeles) y una primitiva de dibujo (Rect, Ruta, Mapa de bits ...)

Examples

Creando una pintura

Puedes crear una nueva pintura con uno de estos 3 constructores:

- `new Paint()` Crear con la configuración predeterminada
- `new Paint(int flags)` Crear con banderas
- `new Paint(Paint from)` Copiar configuraciones de otra pintura

En general, se sugiere nunca crear un objeto de pintura, o cualquier otro objeto en onDraw (), ya que puede llevar a problemas de rendimiento. (Android Studio probablemente te lo advertirá) En su lugar, hazlo global e inicialízalo en el constructor de tu clase de la siguiente manera:

```
public class CustomView extends View {

    private Paint paint;

    public CustomView(Context context) {
        super(context);
        paint = new Paint();
        //...
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        paint.setColor(0xFF000000);
        // ...
    }
}
```

Configuración de la pintura para el texto

Ajustes de dibujo de texto

- `setTypeface(Typeface typeface)` Establece la fuente. Ver tipografía
- `setTextSize(int size)` Establece el tamaño de la fuente, en píxeles.
- `setColor(int color)` Establece el color del dibujo de pintura, incluido el color del texto. También puede usar `setARGB(int a, int r, int g, int b` y `setAlpha(int alpha)`

- `setLetterSpacing(float size)` Establece el espaciado entre los caracteres, en ems. El valor predeterminado es 0, un valor negativo ajustará el texto, mientras que uno positivo lo expandirá.
- `setTextAlign(Paint.Align align)` Establece la alineación del texto en relación con su origen. `Paint.Align.LEFT` lo dibujará a la derecha del origen, `RIGHT` lo dibujará a la izquierda y `CENTER` lo dibujará centrado en el origen (horizontalmente)
- `setTextSkewX(float skewX)` Esto podría considerarse como una cursiva falsa. SkewX representa el desplazamiento horizontal de la parte inferior del texto. (use -0.25 para cursiva)
- `setStyle(Paint.Style style)` Llenar el texto `FILL` , poner texto `STROKE` , o ambos `FILL_AND_STROKE`

Tenga en cuenta que puede usar `TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, size, getResources().getDisplayMetrics())` para convertir de SP o DP a píxeles.

Texto de medición

- `float width = paint.measureText(String text)` Mide el ancho del texto
- `float height = paint.ascent()` Mide la altura del texto
- `paint.getTextBounds(String text, int start, int end, Rect bounds` Almacena las dimensiones del texto. Usted ha asignado el Rect, no puede ser nulo:

```
String text = "Hello world!";
Rect bounds = new Rect();
paint.getTextBounds(text, 0, text.length(), bounds);
```

Hay otros métodos para medir, sin embargo, estos tres deben ajustarse a la mayoría de los propósitos.

Configuración de pintura para dibujar formas.

- `setStyle(Paint.Style style)` forma rellena `FILL` , forma Stroke `STROKE` , o ambos `FILL_AND_STROKE`
- `setColor(int color)` Establece el color de dibujo de la pintura. También puede usar `setARGB(int a, int r, int g, int b` y `setAlpha(int alpha)`
- `setStrokeCap(Paint.Cap cap)` Establezca los límites de línea, ya sea `ROUND` , `SQUARE` o `BUTT` (ninguno) Vea esto .
- `setStrokeJoin(Paint.Join join)` Establece uniones de línea, ya sea `MITER` (puntiaguda), `ROUND` o `BEVEL` . Ver esto
- `setStrokeMiter(float miter)` Establezca el límite de unión de inglete. Esto puede evitar que la unión en inglete se realice indefinidamente, convirtiéndola en una unión en bisel después de x píxeles. Ver esto
- `setStrokeWidth(float width)` Establezca el ancho del trazo. `0` dibujará en modo rayita, independiente de la matriz del lienzo. (siempre 1 píxel)

Poniendo banderas

Puede establecer las siguientes banderas en el constructor, o con `setFlags(int flags)`

- `Paint.ANTI_ALIAS_FLAG` Habilitar antialiasing, suaviza el dibujo.
- `Paint.DITHER_FLAG` Habilitar el dithering. Si la precisión del color es mayor que la del dispositivo, esto sucederá .
- `Paint.EMBEDDED_BITMAP_TEXT_FLAG` Habilita el uso de fuentes de mapa de bits.
- `Paint.FAKE_BOLD_TEXT_FLAG` dibujará texto con un efecto en negrita falso, se puede usar en lugar de usar una tipografía en negrita. Algunas fuentes tienen un estilo audaz, falso negrita no
- `Paint.FILTER_BITMAP_FLAG` Afecta el muestreo de mapas de bits cuando se transforma.
- `Paint.HINTING_OFF` , `Paint.HINTING_ON` Alterna sugerencias de fuentes, vea esto
- `Paint.LINEAR_TEXT_FLAG` Deshabilita la escala de fuente, las operaciones de dibujo se escalan en su lugar
- `Paint.SUBPIXEL_TEXT_FLAG` texto se computará utilizando la precisión de subpíxel.
- `Paint.STRIKE_THRU_TEXT_FLAG` texto dibujado será tachado
- `Paint.UNDERLINE_TEXT_FLAG` texto dibujado estará subrayado

Puedes agregar una bandera y eliminar banderas como esta:

```
Paint paint = new Paint();
paint.setFlags(paint.getFlags() | Paint.FLAG);   // Add flag
paint.setFlags(paint.getFlags() & ~Paint.FLAG);  // Remove flag
```

Intentar eliminar una bandera que no está allí o agregar una bandera que ya está allí no cambiará nada. También tenga en cuenta que la mayoría de los indicadores también se pueden establecer utilizando `set<Flag>(boolean enabled)` , por ejemplo `setAntialias(true)` .

Puede usar `paint.reset()` para restablecer la pintura a su configuración predeterminada. El único indicador predeterminado es `EMBEDDED_BITMAP_TEXT_FLAG` . Se establecerá incluso si utiliza `new Paint(0)` , tendrá

Lea Pintar en línea: https://riptutorial.com/es/android/topic/9141/pintar

Capítulo 190: Pista de audio

Examples

Generar tono de una frecuencia específica.

Para reproducir un sonido con un tono específico, primero tenemos que crear un sonido de onda sinusoidal. Esto se hace de la siguiente manera.

```
final int duration = 10; // duration of sound
final int sampleRate = 22050; // Hz (maximum frequency is 7902.13Hz (B8))
final int numSamples = duration * sampleRate;
final double samples[] = new double[numSamples];
final short buffer[] = new short[numSamples];
for (int i = 0; i < numSamples; ++i)
{
 samples[i] = Math.sin(2 * Math.PI * i / (sampleRate / note[0])); // Sine wave
 buffer[i] = (short) (samples[i] * Short.MAX_VALUE);  // Higher amplitude increases volume
}
```

Ahora tenemos que configurar AudioTrack para reproducir de acuerdo con el búfer generado. Se realiza de la siguiente manera.

```
AudioTrack audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC,
                    sampleRate, AudioFormat.CHANNEL_OUT_MONO,
                    AudioFormat.ENCODING_PCM_16BIT, buffer.length,
                    AudioTrack.MODE_STATIC);
```

Escribe el búfer generado y reproduce la pista.

```
audioTrack.write(buffer, 0, buffer.length);
audioTrack.play();
```

Espero que esto ayude :)

Lea Pista de audio en línea: https://riptutorial.com/es/android/topic/9155/pista-de-audio

Capítulo 191: Política de modo estricto: una herramienta para detectar el error en el tiempo de compilación.

Introducción

El modo estricto es una clase especial introducida en Android 2.3 para la depuración. Estas herramientas de desarrollo detectan las cosas que se hacen accidentalmente y nos las traen a nuestra atención para que podamos solucionarlas. Se usa más comúnmente para capturar el acceso accidental al disco o la red en el subproceso principal de las aplicaciones, donde se reciben las operaciones de la interfaz de usuario y se llevan a cabo las animaciones. StrictMode es básicamente una herramienta para detectar el error en el modo de tiempo de compilación.

Observaciones

StrictMode es básicamente una herramienta para detectar el error en el modo de tiempo de compilación. Usando esto podemos evitar las fugas de memoria en nuestras aplicaciones.

Examples

El siguiente fragmento de código es para configurar StrictMode para políticas de subprocesos. Este Código se debe establecer en los puntos de entrada a nuestra aplicación.

```
StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
    .detectDiskWrites()
    .penaltyLog() //Logs a message to LogCat
    .build())
```

El código siguiente trata las fugas de memoria, como las que se detectan cuando se llama a SQLLite para finalizar o no.

```
StrictMode.setVmPolicy(new StrictMode.VmPolicy.Builder()
    .detectActivityLeaks()
    .detectLeakedClosableObjects()
    .penaltyLog()
    .build());
```

Lea Política de modo estricto: una herramienta para detectar el error en el tiempo de compilación. en línea: https://riptutorial.com/es/android/topic/8756/politica-de-modo-estricto--una-herramienta-para-detectar-el-error-en-el-tiempo-de-compilacion-

Capítulo 192: Preferencias compartidas

Introducción

SharedPreferences proporciona una forma de guardar datos en el disco en forma de pares **clave-valor** .

Sintaxis

- **Método de contexto**
  - Public SharedPreferences getSharedPreferences (nombre de cadena, modo int)
- **Metodo de actividad**
  - Public SharedPreferences getPreferences ()
- **Métodos SharedPreferences**
  - Public SharedPreferences.Editor edit ()
  - booleano público contiene ()
  - Mapa público <String,?> getAll ()
  - public boolean getBoolean (String key, boolean defValue)
  - public float getFloat (String key, float defValue)
  - public int getInt (String key, int defValue)
  - public long getLong (String key, long defValue)
  - public String getString (String key, String defValue)
  - conjunto público getStringSet (clave de cadena, conjunto defValues)
  - public void registerOnSharedPreferenceChangeListener (SharedPreferences.OnSharedPreferenceChangeListener listener)
  - public void unregisterOnSharedPreferenceChangeListener (SharedPreferences.OnSharedPreferenceChangeListener listener)
- **SharedPreferences. Métodos del editor**
  - se aplica el vacío público ()
  - public boolean commit ()
  - Public SharedPreferences.Editor clear ()
  - Public SharedPreferences.Editor putBoolean (clave de cadena, valor booleano)
  - Public SharedPreferences.Editor putFloat (clave de cadena, valor flotante)
  - Public SharedPreferences.Editor putInt (clave de cadena, valor int)
  - Public SharedPreferences.Editor putLong (clave de cadena, valor largo)
  - Public SharedPreferences.Editor putString (clave de cadena, valor de cadena)
  - Public SharedPreferences.Editor putStringSet (clave de cadena, valores establecidos)
  - Public SharedPreferences.Editor remove (clave de cadena)

Parámetros

Parámetro	Detalles
llave	Una `String` no nula que identifica el parámetro. Puede contener espacios en blanco o no imprimibles. Esto solo se usa dentro de su aplicación (y en el archivo XML), por lo que no tiene que tener un espacio de nombre, pero es una buena idea tenerla como una constante en su código fuente. No lo localices.
desvalorizar	Todas las funciones de obtención toman un valor predeterminado, que se devuelve si la clave dada no está presente en las `SharedPreferences` . No se devuelve si la clave está presente pero el valor tiene un tipo incorrecto: en ese caso, se obtiene una `ClassCastException` .

Observaciones

- `SharedPreferences` no debe utilizarse para almacenar gran cantidad de datos. Para tales propósitos, es mucho mejor usar `SQLiteDatabase` .
- `SharedPreferences` son solo procesos únicos, a menos que use el modo en desuso `MODE_MULTI_PROCESS` . Entonces, si su aplicación tiene múltiples procesos, no podrá leer las `SharedPreferences` del proceso principal en otro proceso. En tales casos, debe usar otro mecanismo para compartir datos entre procesos, pero no use `MODE_MULTI_PROCESS` ya que no es confiable y está en desuso.
- Es mejor usar la instancia de `SharedPreferences` en la clase `Singleton` para acceder a todo el `context` la aplicación. Si desea usarlo solo para una Actividad en particular, vaya a `getPreferences()` .
- Evite almacenar información confidencial en texto sin `SharedPreferences` mientras usa `SharedPreferences` ya que se puede leer fácilmente.

Documentacion oficial

https://developer.android.com/reference/android/content/SharedPreferences.html

Examples

Leer y escribir valores en SharedPreferences

```
public class MyActivity extends Activity {

    private static final String PREFS_FILE = "NameOfYourPrefrenceFile";
    // PREFS_MODE defines which apps can access the file
```

```
    private static final int PREFS_MODE = Context.MODE_PRIVATE;
    // you can use live template "key" for quickly creating keys
    private static final String KEY_BOOLEAN = "KEY_FOR_YOUR_BOOLEAN";
    private static final String KEY_STRING = "KEY_FOR_YOUR_STRING";
    private static final String KEY_FLOAT = "KEY_FOR_YOUR_FLOAT";
    private static final String KEY_INT = "KEY_FOR_YOUR_INT";
    private static final String KEY_LONG = "KEY_FOR_YOUR_LONG";

    @Override
    protected void onStart() {
        super.onStart();

        // Get the saved flag (or default value if it hasn't been saved yet)
        SharedPreferences settings = getSharedPreferences(PREFS_FILE, PREFS_MODE);
        // read a boolean value (default false)
        boolean booleanVal = settings.getBoolean(KEY_BOOLEAN, false);
        // read an int value (Default 0)
        int intVal = settings.getInt(KEY_INT, 0);
        // read a string value (default "my string")
        String str = settings.getString(KEY_STRING, "my string");
        // read a long value (default 123456)
        long longVal = settings.getLong(KEY_LONG, 123456);
        // read a float value (default 3.14f)
        float floatVal = settings.getFloat(KEY_FLOAT, 3.14f);
    }

    @Override
    protected void onStop() {
        super.onStop();

        // Save the flag
        SharedPreferences settings = getSharedPreferences(PREFS_FILE, PREFS_MODE);
        SharedPreferences.Editor editor = settings.edit();
        // write a boolean value
        editor.putBoolean(KEY_BOOLEAN, true);
        // write an integer value
        editor.putInt(KEY_INT, 123);
        // write a string
        editor.putString(KEY_STRING, "string value");
        // write a long value
        editor.putLong(KEY_LONG, 456876451);
        // write a float value
        editor.putFloat(KEY_FLOAT, 1.51f);
        editor.apply();
    }
}
```

`getSharedPreferences()` es un método de la clase de `Context` , que la `Activity` extiende. Si necesita acceder al método `getSharedPreferences()` desde otras clases, puede usar `context.getSharedPreferences()` con una referencia de objeto de `Context` de una `Activity` , `View` o `Application` .

Quitando llaves

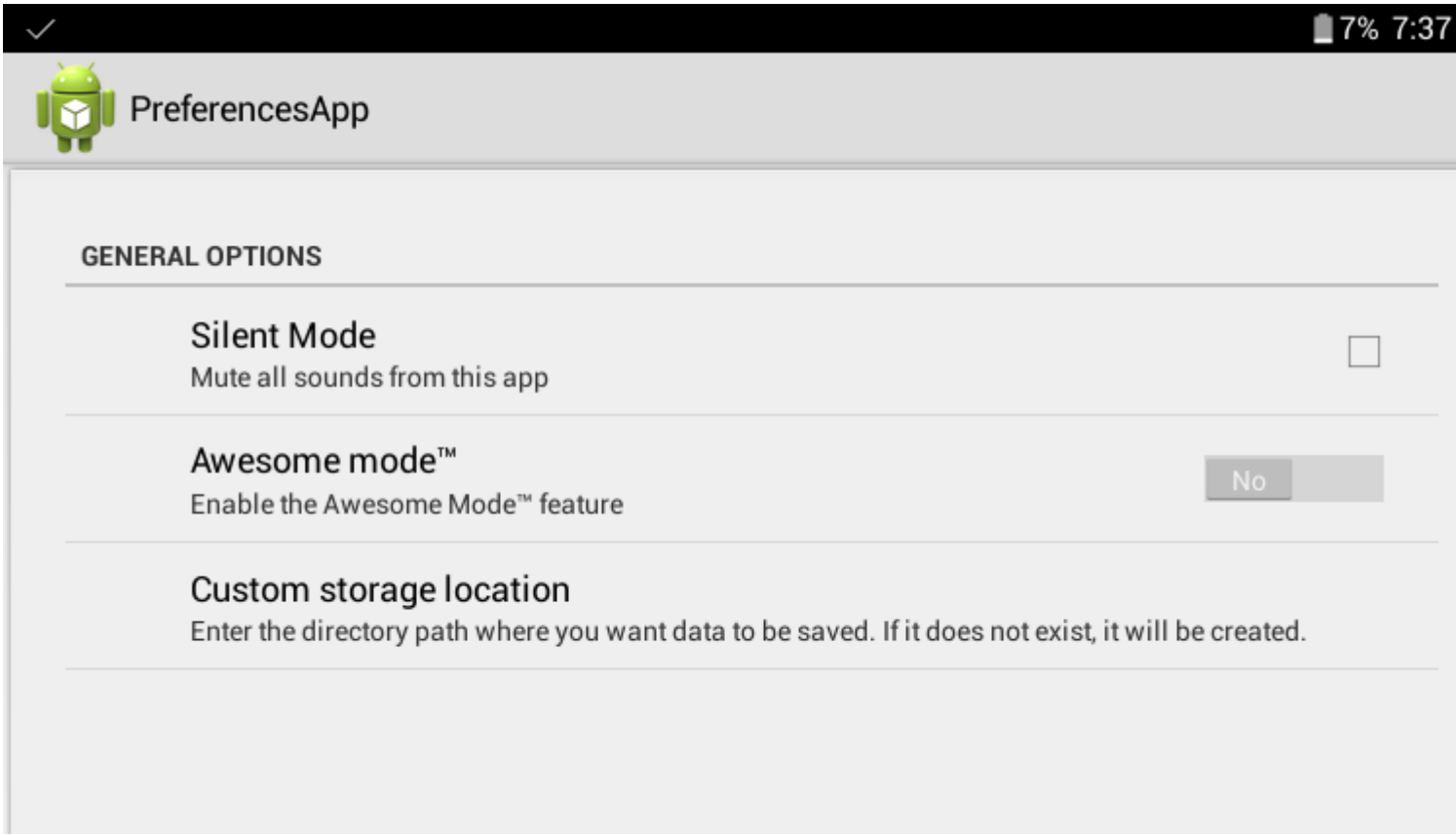
```
private static final String MY_PREF = "MyPref";

// ...
```

```
SharedPreferences prefs = ...;

// ...

SharedPreferences.Editor editor = prefs.edit();
editor.putString(MY_PREF, "value");
editor.remove(MY_PREF);
editor.apply();
```

Después de `apply()` , `prefs` contiene "clave" -> "valor", además de lo que ya contenía. Aunque parece que agregué "clave" y luego la eliminé, la eliminación realmente sucede primero. Los cambios en el `Editor` se aplican todos de una vez, no en el orden en que los agregó. Todas las eliminaciones suceden antes que todas las puestas.

Implementando una pantalla de configuración usando SharedPreferences

Un uso de `SharedPreferences` es implementar una pantalla de "Configuración" en su aplicación, donde el usuario puede configurar sus preferencias / opciones. Me gusta esto:



Una PreferenceScreen guarda las preferencias del usuario en `SharedPreferences` . Para crear una PreferenceScreen, necesitas algunas cosas:

Un archivo XML para definir las opciones disponibles:

Esto va en `/res/xml/preferences.xml` , y para la pantalla de configuración anterior, se ve así:

```
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory
```

```
        android:title="General options">
        <CheckBoxPreference
            android:key = "silent_mode"
            android:defaultValue="false"
            android:title="Silent Mode"
            android:summary="Mute all sounds from this app" />

        <SwitchPreference
            android:key="awesome_mode"
            android:defaultValue="false"
            android:switchTextOn="Yes"
            android:switchTextOff="No"
            android:title="Awesome mode™"
            android:summary="Enable the Awesome Mode™ feature"/>

        <EditTextPreference
            android:key="custom_storage"
            android:defaultValue="/sdcard/data/"
            android:title="Custom storage location"
            android:summary="Enter the directory path where you want data to be saved. If it
does not exist, it will be created."
            android:dialogTitle="Enter directory path (eg. /sdcard/data/ )"/>
    </PreferenceCategory>
</PreferenceScreen>
```

Esto define las opciones disponibles en la pantalla de configuración. Hay muchos otros tipos de `Preference` enumerados en la documentación de los Desarrolladores de Android en la Clase de Preferencias .

A continuación, necesitamos **una Actividad para alojar nuestra** interfaz de usuario de **Preferencias** . En este caso, es bastante corto, y se ve así:

```
package com.example.preferences;

import android.preference.PreferenceActivity;
import android.os.Bundle;

public class PreferencesActivity extends PreferenceActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

Extiende `PreferenceActivity` y proporciona la interfaz de usuario para la pantalla de preferencias. Puede iniciarse como una actividad normal, en este caso, con algo como:

```
Intent i = new Intent(this, PreferencesActivity.class);
startActivity(i);
```

No te olvides de agregar `PreferencesActivity` a tu `AndroidManifest.xml` .

Obtener los valores de las preferencias dentro de su aplicación es bastante simple, solo llame a `setDefaultValues()` primero, para establecer los valores predeterminados definidos en su

XML, y luego obtenga las `SharedPreferences` predeterminadas. Un ejemplo:

```
//set the default values we defined in the XML
PreferenceManager.setDefaultValues(this, R.xml.preferences, false);
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);

//get the values of the settings options
boolean silentMode = preferences.getBoolean("silent_mode", false);
boolean awesomeMode = preferences.getBoolean("awesome_mode", false);

String customStorage = preferences.getString("custom_storage", "");
```

Recupere todas las entradas almacenadas de un archivo de SharedPreferences particular

El método `getAll()` recupera todos los valores de las preferencias. Podemos usarlo, por ejemplo, para registrar el contenido actual de las `SharedPreferences` :

```
private static final String PREFS_FILE = "MyPrefs";

public static void logSharedPreferences(final Context context) {
    SharedPreferences sharedPreferences = context.getSharedPreferences(PREFS_FILE,
Context.MODE_PRIVATE);
    Map<String, ?> allEntries = sharedPreferences.getAll();
    for (Map.Entry<String, ?> entry : allEntries.entrySet()) {
        final String key = entry.getKey();
        final Object value = entry.getValue();
        Log.d("map values", key + ": " + value);
    }
}
```

La documentación le advierte sobre la modificación de la `Collection` devuelta por `getAll` :

> Tenga en cuenta que no debe modificar la colección devuelta por este método ni alterar ninguno de sus contenidos. La consistencia de sus datos almacenados no está garantizada si lo hace.

Escuchando cambios de SharedPreferences

```
SharedPreferences sharedPreferences = ...;
sharedPreferences.registerOnSharedPreferenceChangeListener(mOnSharedPreferenceChangeListener);


private final SharedPreferences.OnSharedPreferenceChangeListener
mOnSharedPreferenceChangeListener = new SharedPreferences.OnSharedPreferenceChangeListener() {
    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
        //TODO
    }
}
```

Tenga en cuenta:

- El oyente disparará solo si el valor se agregó o cambió, establecer el mismo valor no lo llamará;
- El oyente debe guardarse en una variable miembro y **NO** con una clase anónima, porque `registerOnSharedPreferenceChangeListener` almacena con una referencia débil, por lo que sería una recolección de basura;
- En lugar de usar una variable miembro, la clase también puede implementarla directamente y luego llamar a `registerOnSharedPreferenceChangeListener(this);`
- Recuerde anular el registro del oyente cuando ya no sea necesario con `unregisterOnSharedPreferenceChangeListener` .

Lectura y escritura de datos en SharedPreferences con Singleton

SharedPreferences Manager (Singleton) para leer y escribir todo tipo de datos.

```
import android.content.Context;
import android.content.SharedPreferences;
import android.util.Log;

import com.google.gson.Gson;

import java.lang.reflect.Type;

/**
 * Singleton Class for accessing SharedPreferences,
 * should be initialized once in the beginning by any application component using static
 * method initialize(applicationContext)
 */
public class SharedPrefsManager {

    private static final String TAG = SharedPrefsManager.class.getName();
    private SharedPreferences prefs;
    private static SharedPrefsManager uniqueInstance;
    public static final String PREF_NAME = "com.example.app";

    private SharedPrefsManager(Context appContext) {
        prefs = appContext.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
    }

    /**
     * Throws IllegalStateException if this class is not initialized
     *
     * @return unique SharedPrefsManager instance
     */
    public static SharedPrefsManager getInstance() {
        if (uniqueInstance == null) {
            throw new IllegalStateException(
                    "SharedPrefsManager is not initialized, call
initialize(applicationContext) " +
                            "static method first");
        }
        return uniqueInstance;
    }

    /**
     * Initialize this class using application Context,
     * should be called once in the beginning by any application Component
     *
```

```
     * @param appContext application context
     */
    public static void initialize(Context appContext) {
        if (appContext == null) {
            throw new NullPointerException("Provided application context is null");
        }
        if (uniqueInstance == null) {
            synchronized (SharedPrefsManager.class) {
                if (uniqueInstance == null) {
                    uniqueInstance = new SharedPrefsManager(appContext);
                }
            }
        }
    }

    private SharedPreferences getPrefs() {
        return prefs;
    }

    /**
     * Clears all data in SharedPreferences
     */
    public void clearPrefs() {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.clear();
        editor.commit();
    }

    public void removeKey(String key) {
        getPrefs().edit().remove(key).commit();
    }

    public boolean containsKey(String key) {
        return getPrefs().contains(key);
    }

    public String getString(String key, String defValue) {
        return getPrefs().getString(key, defValue);
    }

    public String getString(String key) {
        return getString(key, null);
    }

    public void setString(String key, String value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putString(key, value);
        editor.apply();
    }

    public int getInt(String key, int defValue) {
        return getPrefs().getInt(key, defValue);
    }

    public int getInt(String key) {
        return getInt(key, 0);
    }

    public void setInt(String key, int value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putInt(key, value);
```

```
        editor.apply();
    }

    public long getLong(String key, long defValue) {
        return getPrefs().getLong(key, defValue);
    }

    public long getLong(String key) {
        return getLong(key, 0L);
    }

    public void setLong(String key, long value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putLong(key, value);
        editor.apply();
    }

    public boolean getBoolean(String key, boolean defValue) {
        return getPrefs().getBoolean(key, defValue);
    }

    public boolean getBoolean(String key) {
        return getBoolean(key, false);
    }

    public void setBoolean(String key, boolean value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putBoolean(key, value);
        editor.apply();
    }

    public boolean getFloat(String key) {
        return getFloat(key, 0f);
    }

    public boolean getFloat(String key, float defValue) {
        return getFloat(key, defValue);
    }

    public void setFloat(String key, Float value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putFloat(key, value);
        editor.apply();
    }

    /**
     * Persists an Object in prefs at the specified key, class of given Object must implement
Model
     * interface
     *
     * @param key         String
     * @param modelObject Object to persist
     * @param <M>         Generic for Object
     */
    public <M extends Model> void setObject(String key, M modelObject) {
        String value = createJSONStringFromObject(modelObject);
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putString(key, value);
        editor.apply();
    }
```

```
    /**
     * Fetches the previously stored Object of given Class from prefs
     *
     * @param key                String
     * @param classOfModelObject Class of persisted Object
     * @param <M>                Generic for Object
     * @return Object of given class
     */
    public <M extends Model> M getObject(String key, Class<M> classOfModelObject) {
        String jsonData = getPrefs().getString(key, null);
        if (null != jsonData) {
            try {
                Gson gson = new Gson();
                M customObject = gson.fromJson(jsonData, classOfModelObject);
                return customObject;
            } catch (ClassCastException cce) {
                Log.d(TAG, "Cannot convert string obtained from prefs into collection of type
" +
                        classOfModelObject.getName() + "\n" + cce.getMessage());
            }
        }
        return null;
    }

    /**
     * Persists a Collection object in prefs at the specified key
     *
     * @param key            String
     * @param dataCollection Collection Object
     * @param <C>            Generic for Collection object
     */
    public <C> void setCollection(String key, C dataCollection) {
        SharedPreferences.Editor editor = getPrefs().edit();
        String value = createJSONStringFromObject(dataCollection);
        editor.putString(key, value);
        editor.apply();
    }

    /**
     * Fetches the previously stored Collection Object of given type from prefs
     *
     * @param key     String
     * @param typeOfC Type of Collection Object
     * @param <C>     Generic for Collection Object
     * @return Collection Object which can be casted
     */
    public <C> C getCollection(String key, Type typeOfC) {
        String jsonData = getPrefs().getString(key, null);
        if (null != jsonData) {
            try {
                Gson gson = new Gson();
                C arrFromPrefs = gson.fromJson(jsonData, typeOfC);
                return arrFromPrefs;
            } catch (ClassCastException cce) {
                Log.d(TAG, "Cannot convert string obtained from prefs into collection of type
" +
                        typeOfC.toString() + "\n" + cce.getMessage());
            }
        }
        return null;
    }
```

```
    public void registerPrefsListener(SharedPreferences.OnSharedPreferenceChangeListener
listener) {
        getPrefs().registerOnSharedPreferenceChangeListener(listener);
    }

    public void unregisterPrefsListener(

            SharedPreferences.OnSharedPreferenceChangeListener listener) {
        getPrefs().unregisterOnSharedPreferenceChangeListener(listener);
    }

    public SharedPreferences.Editor getEditor() {
        return getPrefs().edit();
    }

    private static String createJSONStringFromObject(Object object) {
        Gson gson = new Gson();
        return gson.toJson(object);
    }
}
```

`interface Model` implementada por las clases que van a `Gson` para evitar la ofuscación del programa.

```
public interface Model {

}
```

Reglas de progreso para la interfaz del `Model` :

```
-keep interface com.example.app.Model
-keep class * implements com.example.app.Model { *;}
```

Diferentes formas de instanciar un objeto de SharedPreferences

Puede acceder a SharedPreferences de varias maneras:

Obtenga el archivo predeterminado SharedPreferences:

```
import android.preference.PreferenceManager;
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
```

Obtener un archivo específico de SharedPreferences:

```
public static final String PREF_FILE_NAME = "PrefFile";
SharedPreferences prefs = getSharedPreferences(PREF_FILE_NAME, MODE_PRIVATE);
```

Obtenga SharedPreferences de otra aplicación:

```
// Note that the other app must declare prefs as MODE_WORLD_WRITEABLE
final ArrayList<HashMap<String,String>> LIST = new ArrayList<HashMap<String,String>>();
Context contextOtherApp = createPackageContext("com.otherapp", Context.MODE_WORLD_WRITEABLE);
```

```
SharedPreferences prefs = contextOtherApp.getSharedPreferences("pref_file_name",
Context.MODE_WORLD_READABLE);
```

getPreferences (int) VS getSharedPreferences (String, int)

`getPreferences(int)`

devuelve las preferencias guardadas por `Activity's class name` como se describe en los documentos :

> Recupere un objeto SharedPreferences para acceder a las preferencias que son privadas para esta actividad. Esto simplemente llama al método subyacente getSharedPreferences (String, int) pasando el nombre de clase de esta actividad como el nombre de las preferencias.

Mientras se usa el método getSharedPreferences (String name, int mode) , se devuelven las preferencias guardadas con el `name` dado. Como en los documentos:

> Recupere y mantenga el contenido del 'nombre' del archivo de preferencias, devolviendo un SharedPreferences a través del cual puede recuperar y modificar sus valores.

Por lo tanto, si el valor que se está `SharedPreferences` en `SharedPreferences` se debe usar en toda la aplicación, se debe usar `getSharedPreferences (String name, int mode)` con un nombre fijo. Al usar `getPreferences(int)` devuelven / `getPreferences(int)` las preferencias que pertenecen a la `Activity` que lo llama.

Cometer vs. Aplicar

El método `editor.apply()` es **asíncrono** , mientras que `editor.commit()` es **síncrono** .

Obviamente, debe llamar a `apply()` o `commit()` .

2.3

```
SharedPreferences settings = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean(PREF_CONST, true);
// This will asynchronously save the shared preferences without holding the current thread.
editor.apply();
```

```
SharedPreferences settings = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean(PREF_CONST, true);
// This will synchronously save the shared preferences while holding the current thread until
done and returning a success flag.
boolean result = editor.commit();
```

`apply()` se agregó en 2.3 (API 9), se compromete sin devolver un valor booleano que indique el éxito o el fracaso.

`commit()` devuelve true si el guardado funciona, falso de lo contrario.

`apply()` se agregó a medida que el equipo de desarrollo de Android notó que casi nadie se dio cuenta del valor de retorno, por lo que se aplica es más rápido ya que es asíncrono.

A diferencia de `commit()` , que escribe sus preferencias en el almacenamiento persistente sincrónicamente, `apply()` confirma sus cambios en las `SharedPreferences` en la memoria de inmediato, pero inicia una confirmación asíncrona en el disco y no se le notificará ningún error. Si otro editor en este `SharedPreferences` hace un `commit()` regular `commit()` mientras un `apply()` aún está pendiente, el `commit()` se bloqueará hasta que se completen todos los commit asíncronos (apply), así como cualquier otro commit de sincronización que pueda estar pendiente.

Tipos de datos soportados en SharedPreferences

`SharedPreferences` permite almacenar tipos de datos primitivos solamente (`boolean` , `float` , `long` , `int` , `String` y `string set`). No puede almacenar objetos más complejos en `SharedPreferences` y, como tal, realmente pretende ser un lugar para almacenar configuraciones de usuario o similar, no pretende ser una base de datos para mantener los datos del usuario (como guardar una lista de tareas pendientes de un usuario, por ejemplo).

Para almacenar algo en `SharedPreferences` utiliza una clave y un valor. La clave es cómo puede hacer referencia a lo que almacenó más tarde y los datos de valor que desea almacenar.

```
    String keyToUseToFindLater = "High Score";
    int newHighScore = 12938;
    //getting SharedPreferences & Editor objects
    SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPref.edit();
    //saving an int in the SharedPreferences file
    editor.putInt(keyToUseToFindLater, newHighScore);
    editor.commit();
```

Almacenar, recuperar, eliminar y borrar datos de SharedPreferences

Crear SharedPreferences BuyyaPref

```
SharedPreferences pref = getApplicationContext().getSharedPreferences("BuyyaPref",
MODE_PRIVATE);
Editor editor = pref.edit();
```

Almacenamiento de datos como par KEY / VALUE

```
editor.putBoolean("key_name1", true);           // Saving boolean - true/false
editor.putInt("key_name2", 10);         // Saving integer
editor.putFloat("key_name3", 10.1f);    // Saving float
editor.putLong("key_name4", 1000);      // Saving long
editor.putString("key_name5", "MyString");  // Saving string

// Save the changes in SharedPreferences
editor.commit(); // commit changes
```

Obtener datos de SharedPreferences

Si el valor para la clave no existe, devuelva el segundo valor param (en este caso, nulo, es como el valor predeterminado)

```
pref.getBoolean("key_name1", null);         // getting boolean
pref.getInt("key_name2", null);             // getting Integer
pref.getFloat("key_name3", null);           // getting Float
pref.getLong("key_name4", null);            // getting Long
pref.getString("key_name5", null);          // getting String
```

Eliminar valor clave de SharedPreferences

```
editor.remove("key_name3"); // will delete key key_name3
editor.remove("key_name4"); // will delete key key_name4

// Save the changes in SharedPreferences
editor.commit(); // commit changes
```

Borrar todos los datos de SharedPreferences

```
 editor.clear();
 editor.commit(); // commit changes
```

Soporte pre-Honeycomb con StringSet

Aquí está la clase de utilidad:

```
public class SharedPreferencesCompat {

    public static void putStringSet(SharedPreferences.Editor editor, String key, Set<String>
values) {
            if (Build.VERSION.SDK_INT >= 11) {
                while (true) {
                    try {
                        editor.putStringSet(key, values).apply();
                        break;
                    } catch (ClassCastException ex) {
                        // Clear stale JSON string from before system upgrade
                        editor.remove(key);
                    }
                }
            } else putStringSetToJson(editor, key, values);
    }

    public static Set<String> getStringSet(SharedPreferences prefs, String key, Set<String>
defaultReturnValue) {
        if (Build.VERSION.SDK_INT >= 11) {
            try {
                return prefs.getStringSet(key, defaultReturnValue);
            } catch (ClassCastException ex) {
                // If user upgraded from Gingerbread to something higher read the stale JSON
string
                return getStringSetFromJson(prefs, key, defaultReturnValue);
            }
```

```
        } else return getStringSetFromJson(prefs, key, defaultReturnValue);
    }

    private static Set<String> getStringSetFromJson(SharedPreferences prefs, String key,
Set<String> defaultReturnValue) {
        final String input = prefs.getString(key, null);
        if (input == null) return defaultReturnValue;

        try {
            HashSet<String> set = new HashSet<>();
            JSONArray json = new JSONArray(input);
            for (int i = 0, size = json.length(); i < size; i++) {
                String value = json.getString(i);
                set.add(value);
            }
            return set;
        } catch (JSONException e) {
            e.printStackTrace();
            return defaultReturnValue;
        }
    }

    private static void putStringSetToJson(SharedPreferences.Editor editor, String key,
Set<String> values) {
        JSONArray json = new JSONArray(values);
        if (Build.VERSION.SDK_INT >= 9)
            editor.putString(key, json.toString()).apply();
        else
            editor.putString(key, json.toString()).commit();
    }

    private SharedPreferencesCompat() {}
}
```

Un ejemplo para guardar preferencias como tipo de datos StringSet es:

```
Set<String> sets = new HashSet<>();
sets.add("John");
sets.add("Nicko");
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
SharedPreferencesCompat.putStringSet(preferences.edit(), "pref_people", sets);
```

Para recuperarlos de nuevo:

```
Set<String> people = SharedPreferencesCompat.getStringSet(preferences, "pref_people", new
HashSet<String>());
```

Referencia: preferencia de soporte de Android

Añadir filtro para EditTextPreference

Crea esta clase:

```
public class InputFilterMinMax implements InputFilter {

    private int min, max;
```

```
    public InputFilterMinMax(int min, int max) {
        this.min = min;
        this.max = max;
    }

    public InputFilterMinMax(String min, String max) {
        this.min = Integer.parseInt(min);
        this.max = Integer.parseInt(max);
    }

    @Override
    public CharSequence filter(CharSequence source, int start, int end, Spanned dest, int
dstart, int dend) {
        try {
            int input = Integer.parseInt(dest.toString() + source.toString());
            if (isInRange(min, max, input))
                return null;
        } catch (NumberFormatException nfe) { }
        return "";
    }

    private boolean isInRange(int a, int b, int c) {
        return b > a ? c >= a && c <= b : c >= b && c <= a;
    }
}
```

Utilizar :

```
EditText compressPic = ((EditTextPreference)
findPreference(getString("pref_key_compress_pic"))).getEditText();
compressPic.setFilters(new InputFilter[]{ new InputFilterMinMax(1, 100) });
```

Lea Preferencias compartidas en línea: https://riptutorial.com/es/android/topic/119/preferencias-compartidas

Capítulo 193: Procesador de anotaciones

Introducción

El procesador de anotaciones es una herramienta desarrollada en javac para escanear y procesar anotaciones en tiempo de compilación.

Las anotaciones son una clase de metadatos que se pueden asociar con clases, métodos, campos e incluso otras anotaciones. Hay dos formas de acceder a estas anotaciones en tiempo de ejecución a través de la reflexión y en tiempo de compilación a través de los procesadores de anotación.

Examples

@NonNull Annotation

```
public class Foo {
    private String name;
    public Foo(@NonNull String name){...};
    ...
}
```

Aquí @NonNull es una anotación que es procesada por el estudio de Android para advertirle que la función particular necesita un parámetro no nulo.

Tipos de anotaciones

Hay tres tipos de anotaciones.

1. **Anotación de marcador** - anotación que no tiene método

   ```
   @interface CustomAnnotation {}
   ```

2. **Anotación de valor único** : anotación que tiene un método

   ```
   @interface CustomAnnotation {
       int value();
   }
   ```

3. **Anotación multivalor** : anotación que tiene más de un método

   ```
   @interface CustomAnnotation{
       int value1();
       String value2();
       String value3();
   }
   ```

Creación y uso de anotaciones personalizadas

Para crear anotaciones personalizadas necesitamos decidir

- Destino: en el que funcionarán estas anotaciones como nivel de campo, nivel de método, nivel de tipo, etc.
- Retención - a qué nivel de anotación estará disponible.

Para esto, hemos construido en anotaciones personalizadas. Echa un vistazo a estos más utilizados:

@Objetivo

Element Types	Where the a
TYPE	class, interfac
FIELD	fields
METHOD	methods
CONSTRUCTOR	constructors
LOCAL_VARIABLE	local variable
ANNOTATION_TYPE	annotation ty
PARAMETER	parameter

@Retencion

RetentionPolicy	Availability
RetentionPolicy.SOURCE	refers to the source code, d class.
RetentionPolicy.CLASS	refers to the .class file, ava file.
RetentionPolicy.RUNTIME	refers to the runtime, availa

Creación de anotaciones personalizadas

```
@Retention(RetentionPolicy.SOURCE) // will not be available in compiled class
@Target(ElementType.METHOD) // can be applied to methods only
@interface CustomAnnotation{
      int value();
}
```

Uso de anotaciones personalizadas

```
class Foo{
  @CustomAnnotation(value = 1)  // will be used by an annotation processor
  public void foo(){..}
}
```

el valor proporcionado dentro de `@CustomAnnotation` será consumido por un Annotationprocessor puede ser generar código en tiempo de compilación, etc.

Lea Procesador de anotaciones en línea:
https://riptutorial.com/es/android/topic/10726/procesador-de-anotaciones

Capítulo 194: Programación de Android con Kotlin.

Introducción

Usar Kotlin con Android Studio es una tarea fácil, ya que Kotlin es desarrollado por JetBrains. Es la misma compañía que respalda a IntelliJ IDEA, un IDE base para Android Studio. Es por eso que casi no hay problemas con la compatibilidad.

Observaciones

Si desea obtener más información sobre el lenguaje de programación Kotlin, consulte la documentación .

Examples

Instalando el plugin de Kotlin

En primer lugar, deberás instalar el complemento Kotlin.

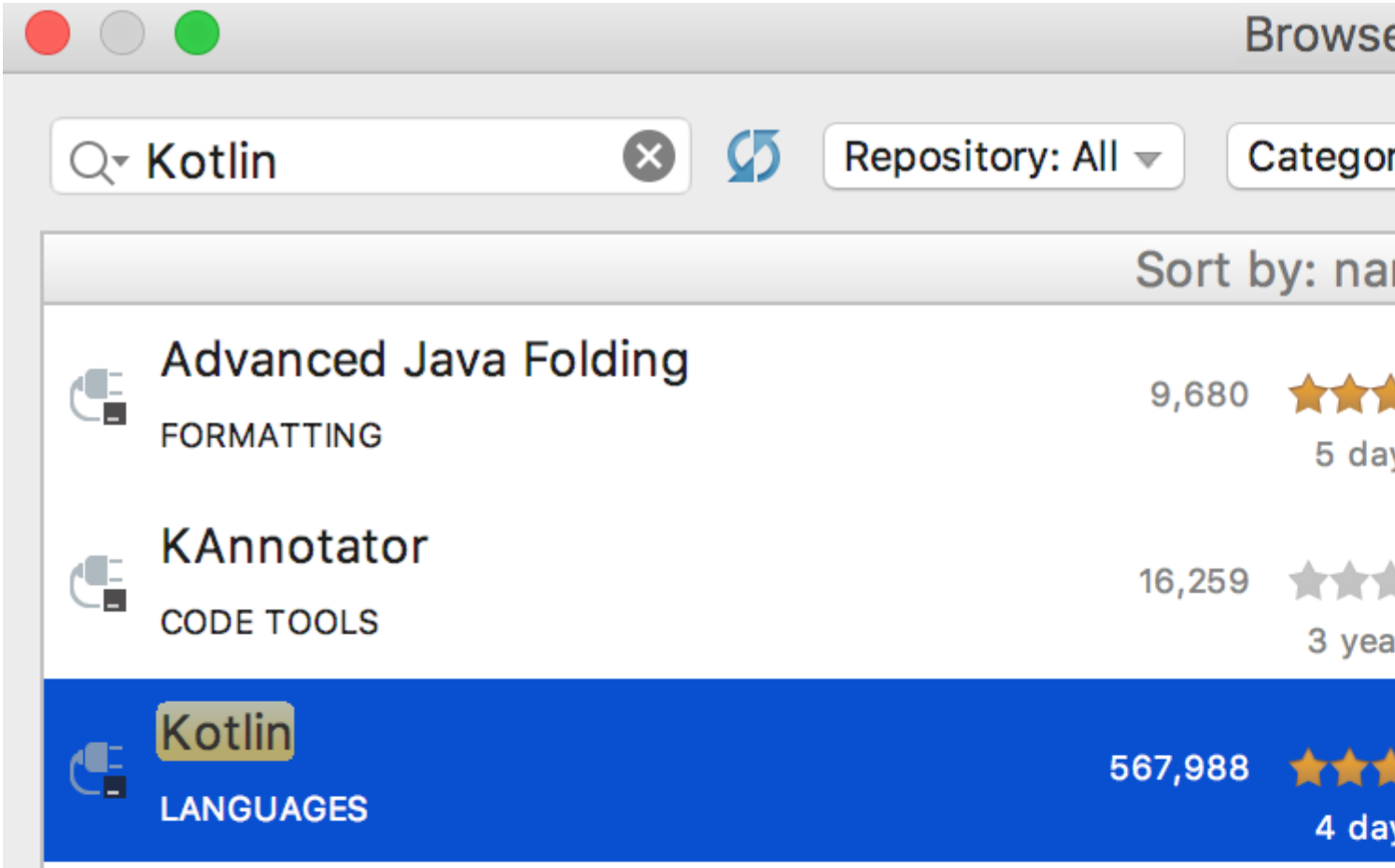
Para ventanas:

- Vaya a `File` → `Settings` → `Plugins` → `Install JetBrains plugin`

Para Mac:

- Navegue a `Android Studio` → `Preferences` → `Plugins` → `Install JetBrains plugin`

Y luego buscar e instalar Kotlin. Tendrá que reiniciar el IDE después de que esto se complete.


Browse
Kotlin
Repository: All
Categor
Sort by: na
Advanced Java Folding
FORMATTING
9,680
5 day
KAnnotator
CODE TOOLS
16,259
3 yea
Kotlin
LANGUAGES
567,988
4 day

y luego agregarle soporte de Kotlin o modificar su proyecto existente. Para hacerlo, tienes que:

1. **Agregue la dependencia a un archivo gradle raíz** : debe agregar la dependencia para el `kotlin-android` a un archivo `build.gradle` raíz.

```
buildscript {

    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.1'
        classpath 'org.jetbrains.kotlin:kotlin-gradle-plugin:1.1.2'
    }
}

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

2. **Aplique el Complemento de Android Kotlin** : simplemente agregue el `apply plugin: 'kotlin-android'` a un módulo `build.gradle` file.

3. **Agregue dependencia a stdlib de Kotlin** : agregue la dependencia a `'org.jetbrains.kotlin:kotlin-stdlib:1.1.2'` a la sección de dependencia en un archivo `build.gradle` módulo.

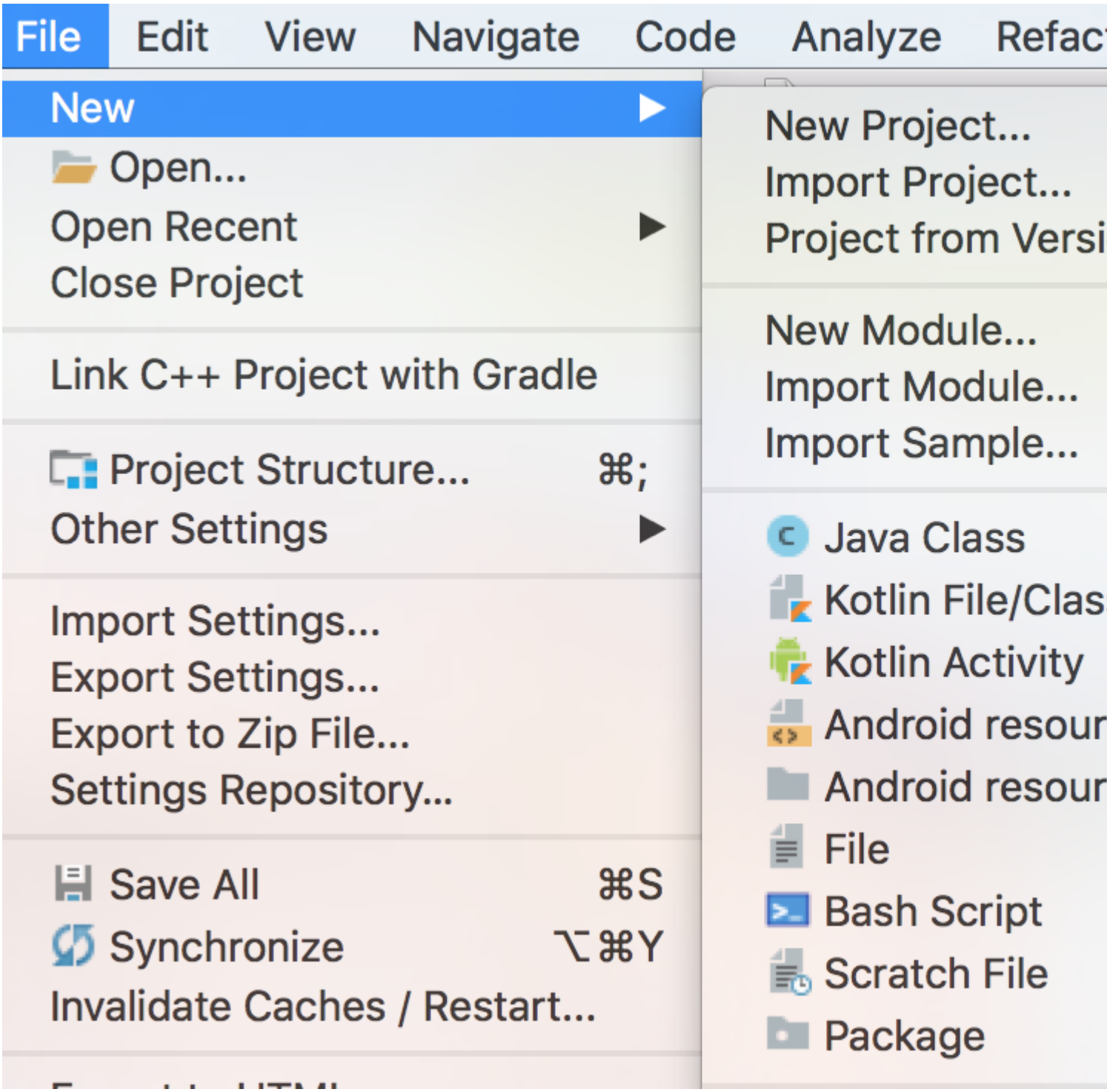
Para un nuevo proyecto, el archivo `build.gradle` podría `build.gradle` así:

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'

android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "org.example.example"
        minSdkVersion 16
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
```

```
dependencies {
    compile 'org.jetbrains.kotlin:kotlin-stdlib:1.1.1'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    compile 'com.android.support:appcompat-v7:25.3.1'

    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })

    testCompile 'junit:junit:4.12'
}
```

Creando una nueva actividad de Kotlin

1. Haga clic en `File` → `New` → `Kotlin Activity` .
2. Elige un tipo de actividad.
3. Seleccione nombre y otro parámetro para la Actividad.
4. Terminar.

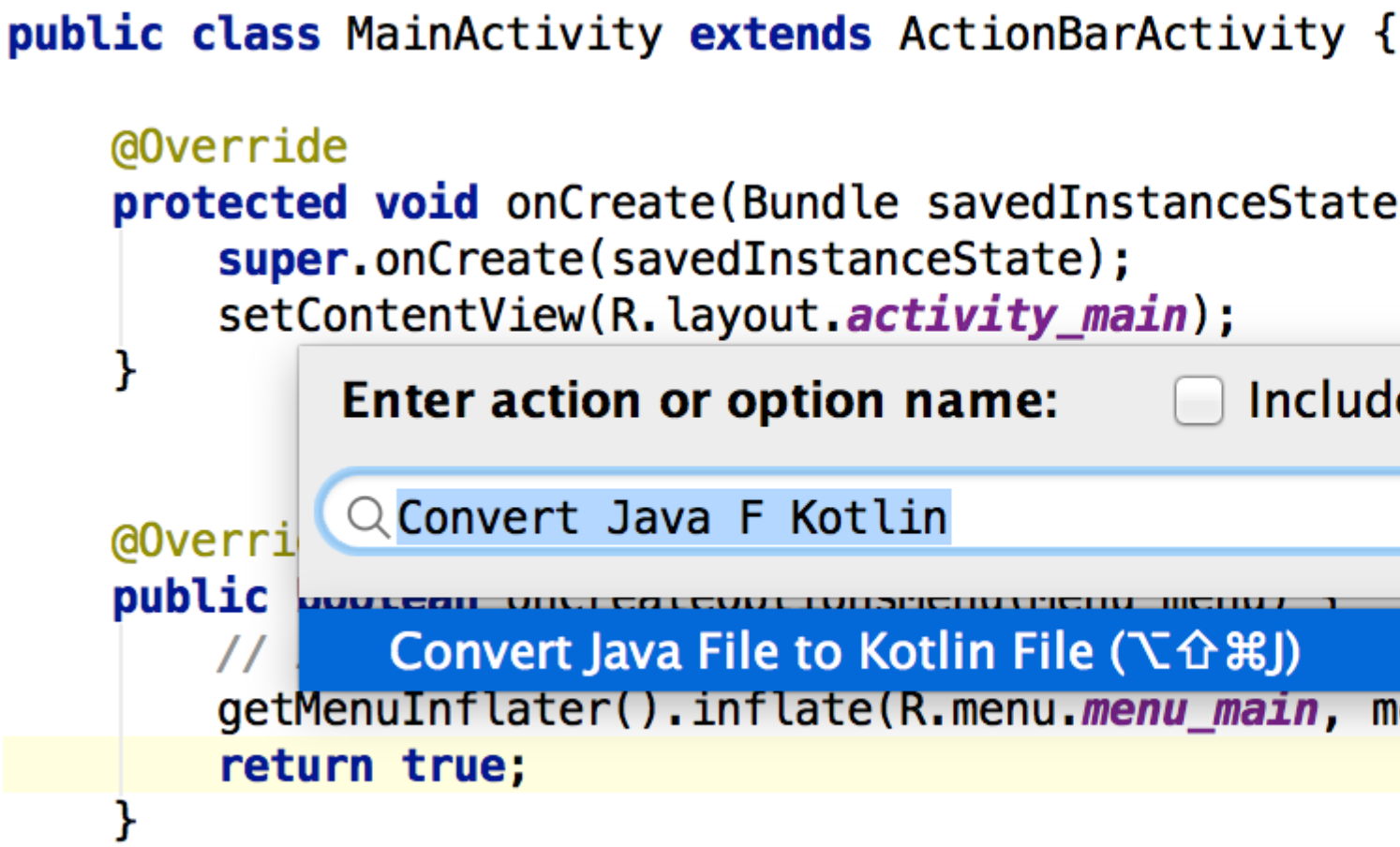


La clase final podría verse así:

```
import android.support.v7.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Convertir código Java existente a Kotlin

Kotlin Plugin para Android Studio admite la conversión de archivos Java existentes a archivos Kotlin. Elija un archivo Java e invoque la acción Convertir archivo Java a archivo Kotlin:



Comenzando una nueva actividad

```
fun startNewActivity(){
    val intent: Intent = Intent(context, Activity::class.java)
    startActivity(intent)
}
```

Puede agregar extras a la intención al igual que en Java.

```
fun startNewActivityWithIntents(){
    val intent: Intent = Intent(context, Activity::class.java)
    intent.putExtra(KEY_NAME, KEY_VALUE)
    startActivity(intent)
}
```

Lea Programación de Android con Kotlin. en línea: https://riptutorial.com/es/android/topic/9623/programacion-de-android-con-kotlin-

Capítulo 195: Programación de trabajos

Observaciones

Tenga cuidado con ejecutar un montón de código o hacer un trabajo pesado dentro de su `JobService` , por ejemplo en `onStartJob()` . El código se ejecutará en el subproceso **principal / UI** y, por lo tanto, puede llevar a una IU bloqueada, que ya no responda a la aplicación o incluso a que se bloquee la aplicación.

Debido a eso, debe descargar el trabajo, por ejemplo, utilizando un `Thread` o `AsyncTask` .

Examples

Uso básico

Crear un nuevo servicio de empleo

Esto se hace extendiendo la clase `JobService` e implementando / reemplazando los métodos requeridos en `onStartJob()` y `onStopJob()` .

```
public class MyJobService extends JobService
{
    final String TAG = getClass().getSimpleName();

    @Override
    public boolean onStartJob(JobParameters jobParameters) {
        Log.i(TAG, "Job started");

        // ... your code here ...

        jobFinished(jobParameters, false);  // signal that we're done and don't want to
reschedule the job
        return false;                       // finished: no more work to be done
    }

    @Override
    public boolean onStopJob(JobParameters jobParameters) {
        Log.w(TAG, "Job stopped");
        return false;
    }
}
```

Agregue el nuevo servicio de trabajo a su AndroidManifest.xml

El siguiente paso es *obligatorio* , de lo contrario no podrá ejecutar su trabajo:

Declare su clase `MyJobService` como un nuevo elemento `<service>` entre `<application> </application>` en su *AndroidManifest.xml* .

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service
            android:name=".MyJobService"
            android:permission="android.permission.BIND_JOB_SERVICE" />
    </application>
</manifest>
```

Configura y ejecuta el trabajo

Después de implementar un nuevo JobService y agregarlo a su *AndroidManifest.xml* , puede continuar con los pasos finales.

- `onButtonClick_startJob()` prepara y ejecuta un trabajo periódico. Además de los trabajos periódicos, `JobInfo.Builder` permite especificar muchas otras configuraciones y restricciones. Por ejemplo, puede definir que se requiere un *cargador enchufado* o una *conexión de red* para ejecutar el trabajo.
- `onButtonClick_stopJob()` cancela todos los trabajos en ejecución

```
public class MainActivity extends AppCompatActivity
{
    final String TAG = getClass().getSimpleName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onButtonClick_startJob(View v) {
        // get the jobScheduler instance from current context
        JobScheduler jobScheduler = (JobScheduler) getSystemService(JOB_SCHEDULER_SERVICE);

        // MyJobService provides the implementation for the job
        ComponentName jobService = new ComponentName(getApplicationContext(),
```

```
MyJobService.class);

        // define that the job will run periodically in intervals of 10 seconds
        JobInfo jobInfo = new JobInfo.Builder(1, jobService).setPeriodic(10 * 1000).build();

        // schedule/start the job
        int result = jobScheduler.schedule(jobInfo);
        if (result == JobScheduler.RESULT_SUCCESS)
            Log.d(TAG, "Successfully scheduled job: " + result);
        else
            Log.e(TAG, "RESULT_FAILURE: " + result);
    }

    public void onButtonClick_stopJob(View v) {
        JobScheduler jobScheduler = (JobScheduler) getSystemService(JOB_SCHEDULER_SERVICE);
        Log.d(TAG, "Stopping all jobs...");
        jobScheduler.cancelAll(); // cancel all potentially running jobs
    }
}
```

Después de llamar a `onButtonClick_startJob()` , el trabajo se ejecutará aproximadamente en intervalos de 10 segundos, incluso cuando la aplicación se encuentre en estado de *pausa* (el botón de inicio presionado por el usuario ya no está visible).

En lugar de cancelar todos los trabajos en `onButtonClick_stopJob()` dentro de `onButtonClick_stopJob()` , también puede llamar a `jobScheduler.cancel()` para cancelar un trabajo específico en función de su ID de trabajo.

Lea Programación de trabajos en línea:
https://riptutorial.com/es/android/topic/6907/programacion-de-trabajos

Capítulo 196: ProGuard - ofuscar y encoger su código

Examples

Reglas para algunas de las bibliotecas ampliamente utilizadas

Actualmente contiene reglas para las siguientes bibliotecas:

1. Cuchillo de mantequilla
2. RxJava
3. Biblioteca de soporte de Android
4. Biblioteca de soporte de diseño de Android
5. Reequipamiento
6. Gson y jackson
7. Otón
8. Crashlitycs
9. Picasso
10. Voleo
11. Okhttp3
12. Parcelable

```
#Butterknife
-keep class butterknife.** { *; }
-keepnames class * { @butterknife.Bind *;}

-dontwarn butterknife.internal.**
-keep class **$$ViewBinder { *; }

-keepclasseswithmembernames class * {
    @butterknife.* <fields>;
}

-keepclasseswithmembernames class * {
    @butterknife.* <methods>;
}

# rxjava
-keep class rx.schedulers.Schedulers {
    public static <methods>;
}
-keep class rx.schedulers.ImmediateScheduler {
    public <methods>;
}
-keep class rx.schedulers.TestScheduler {
    public <methods>;
}
-keep class rx.schedulers.Schedulers {
    public static ** test();
}
-keepclassmembers class rx.internal.util.unsafe.*ArrayQueue*Field* {
```

```
    long producerIndex;
    long consumerIndex;
}
-keepclassmembers class rx.internal.util.unsafe.BaseLinkedQueueProducerNodeRef {
    long producerNode;
    long consumerNode;
}

# Support library
-dontwarn android.support.**
-dontwarn android.support.v4.**
-keep class android.support.v4.** { *; }
-keep interface android.support.v4.** { *; }
-dontwarn android.support.v7.**
-keep class android.support.v7.** { *; }
-keep interface android.support.v7.** { *; }

# support design
-dontwarn android.support.design.**
-keep class android.support.design.** { *; }
-keep interface android.support.design.** { *; }
-keep public class android.support.design.R$* { *; }

# retrofit
-dontwarn okio.**
-keepattributes Signature
-keepattributes *Annotation*
-keep class com.squareup.okhttp.** { *; }
-keep interface com.squareup.okhttp.** { *; }
-dontwarn com.squareup.okhttp.**

-dontwarn rx.**
-dontwarn retrofit.**
-keep class retrofit.** { *; }
-keepclasseswithmembers class * {
    @retrofit.http.* <methods>;
}

-keep class sun.misc.Unsafe { *; }
#your package path where your gson models are stored
-keep class com.abc.model.** { *; }

# Keep these for GSON and Jackson
-keepattributes Signature
-keepattributes *Annotation*
-keepattributes EnclosingMethod
-keep class sun.misc.Unsafe { *; }
-keep class com.google.gson.** { *; }

#keep otto
-keepattributes *Annotation*
-keepclassmembers class ** {
    @com.squareup.otto.Subscribe public *;
    @com.squareup.otto.Produce public *;
}

# Crashlitycs 2.+
-keep class com.crashlytics.** { *; }
-keep class com.crashlytics.android.**
-keepattributes SourceFile, LineNumberTable, *Annotation*
# If you are using custom exceptions, add this line so that custom exception types are skipped
```

```
during obfuscation:
-keep public class * extends java.lang.Exception
# For Fabric to properly de-obfuscate your crash reports, you need to remove this line from
your ProGuard config:
# -printmapping mapping.txt

# Picasso
-dontwarn com.squareup.okhttp.**

# Volley
-keep class com.android.volley.toolbox.ImageLoader { *; }

# OkHttp3
-keep class okhttp3.** { *; }
-keep interface okhttp3.** { *; }
-dontwarn okhttp3.**

# Needed for Parcelable/SafeParcelable Creators to not get stripped
-keepnames class * implements android.os.Parcelable {
    public static final ** CREATOR;
}
```

Habilita ProGuard para tu compilación

Para habilitar las configuraciones de `ProGuard` para su aplicación, debe habilitarla en su archivo de nivel de módulo. necesitas establecer el valor de `minifyEnabled true` .

También puede habilitar `shrinkResources true` que eliminará los recursos que `ProGuard` marca como no utilizados.

```
buildTypes {
        release {
            minifyEnabled true
            shrinkResources true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
```

El código anterior aplicará sus configuraciones de ProGuard contenidas en `proguard-rules.pro` ("proguard-project.txt" en Eclipse) a su apk publicado.

Para que luego pueda determinar la línea en la que se produjo una excepción en un seguimiento de pila, "proguard-rules.pro" debe contener las siguientes líneas:

```
-renamesourcefileattribute SourceFile
-keepattributes SourceFile,LineNumberTable
```

Para habilitar Proguard en Eclipse, agregue `proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt` a "project.properties"

Eliminar las declaraciones de registro de seguimiento (y otras) en el momento de la compilación

Si desea eliminar las llamadas a ciertos métodos, asumiendo que devuelven un vacío y no tienen efectos secundarios (como en, llamarlos no cambia ningún valor del sistema, argumentos de referencia, estadísticas, etc.), entonces puede hacer que ProGuard los elimine de la salida después de la construcción se completa.

Por ejemplo, esto me parece útil para eliminar las declaraciones de registro de depuración / verbosa útiles en la depuración, pero generar las cadenas para ellas no es necesario en la producción.

```
# Remove the debug and verbose level Logging statements.
# That means the code to generate the arguments to these methods will also not be called.
# ONLY WORKS IF -dontoptimize IS _NOT_ USED in any ProGuard configs
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    public static *** v(...);
}
```

Nota: Si se usa `-dontoptimize` en cualquier configuración de ProGuard para que no esté minimizando / eliminando el código no utilizado, esto no eliminará las declaraciones. (Pero a quién no le gustaría quitar el código no utilizado, ¿verdad?)

Nota 2: esta llamada eliminará la llamada al registro, pero no protegerá su código. Las cuerdas permanecerán realmente en el apk generado. Lea más en este post .

Protegiendo su código de hackers

La ofuscación a menudo se considera como una solución mágica para la protección del código, al hacer que su código sea más difícil de entender si los hackers lo descompilan.

Pero si está pensando que eliminar el `Log.x(..)` realmente elimina la información que necesitan los hackers, tendrá una desagradable sorpresa.

Eliminar todas sus llamadas de registro con:

```
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    ...etc
}
```

efectivamente eliminará la llamada de registro en sí misma, pero generalmente *no* las cadenas que se ponen en ellas.

Si, por ejemplo, dentro de su llamada de registro, escribe un mensaje de registro común como: `Log.d(MyTag,"Score="+score);` , el compilador convierte el `+` en un 'nuevo StringBuilder ()' fuera de la llamada de registro. ProGuard no cambia este nuevo objeto.

Su código descompilado todavía tendrá un `StringBuilder` colgante para `"Score="` , añadido con la versión ofuscada para la variable de `score` (digamos que se convirtió a `b`).
Ahora el hacker sabe lo que es `b` , y da sentido a su código.

Una buena práctica para eliminar realmente estos residuos de su código es no ponerlos allí en primer lugar (use el formateador de cadenas en su lugar, con reglas de progreso para eliminarlos), o envolver sus llamadas de `Log` con:

```
    if (BuildConfig.DEBUG) {
        Log.d(TAG,".."+var);
    }
```

Propina:

¡Pruebe lo bien protegido que está su código ofuscado al descompilarlo usted mismo!

1. dex2jar - convierte el apk a jar
2. jd - descompila el frasco y lo abre en un editor de gui

Habilitando ProGuard con un archivo de configuración de ofuscación personalizado

ProGuard permite al desarrollador ofuscar, reducir y optimizar su código.

 # 1 El primer paso del procedimiento es habilitar el programa en la compilación .

Esto se puede hacer **configurando el comando 'minifyEnabled' en verdadero** en la compilación deseada

 # 2 El segundo paso es especificar qué archivos de proguard estamos usando para la compilación dada

Esto se puede hacer **configurando la línea 'proguardFiles' con los nombres de archivo adecuados**

```
buildTypes {
    debug {
        minifyEnabled false
    }
    testRelease {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-
rules-tests.pro'
    }
    productionRelease {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-
rules-tests.pro', 'proguard-rules-release.pro'
    }
}
```

 # 3 El desarrollador puede editar su archivo de programación con las reglas que desee.

Esto se puede hacer editando el archivo (por ejemplo, 'proguard-rules-tests.pro') y agregando las restricciones deseadas. *El siguiente archivo sirve como ejemplo de archivo proguard*

```
// default & basic optimization configurations
-optimizationpasses 5
-dontpreverify
-repackageclasses ''
-allowaccessmodification
-optimizations !code/simplification/arithmetic
-keepattributes *Annotation*

-verbose

-dump obfuscation/class_files.txt
-printseeds obfuscation/seeds.txt
-printusage obfuscation/unused.txt // unused classes that are stripped out in the process
-printmapping obfuscation/mapping.txt // mapping file that shows the obfuscated names of the
classes after proguad is applied

// the developer can specify keywords for the obfuscation (I myself use fruits for obfuscation
names once in a while :-) )
-obfuscationdictionary obfuscation/keywords.txt
-classobfuscationdictionary obfuscation/keywords.txt
-packageobfuscationdictionary obfuscation/keywords.txt
```

Finalmente, cada vez que el desarrollador ejecute y / o genere su nuevo archivo .APK, se aplicarán las configuraciones de programa personalizadas para cumplir con los requisitos.

Lea ProGuard - ofuscar y encoger su código en línea:
https://riptutorial.com/es/android/topic/4500/proguard---ofuscar-y-encoger-su-codigo

Capítulo 197: Proveedor de contenido

Observaciones

Los proveedores de contenido gestionan el acceso a un conjunto estructurado de datos. Encapsulan los datos y proporcionan mecanismos para definir la seguridad de los datos. Los proveedores de contenido son la interfaz estándar que conecta los datos en un proceso con el código que se ejecuta en otro proceso.

Cuando desea acceder a los datos en un proveedor de contenido, utiliza el objeto `ContentResolver` en el `Context` su aplicación para comunicarse con el proveedor como cliente. El objeto `ContentResolver` comunica con el objeto proveedor, una instancia de una clase que implementa `ContentProvider` . El objeto proveedor recibe solicitudes de datos de los clientes, realiza la acción solicitada y devuelve los resultados.

No necesita desarrollar su propio proveedor si no tiene la intención de compartir sus datos con otras aplicaciones. Sin embargo, necesita su propio proveedor para proporcionar sugerencias de búsqueda personalizadas en su propia aplicación. También necesita su propio proveedor si desea copiar y pegar datos o archivos complejos de su aplicación a otras aplicaciones.

El propio Android incluye proveedores de contenido que administran datos como audio, video, imágenes e información de contacto personal. Puede ver algunos de ellos en la documentación de referencia del paquete `android.provider` . Con algunas restricciones, estos proveedores son accesibles a cualquier aplicación de Android.

Examples

Implementando una clase de proveedor de contenido básico

1) Crear una clase de contrato

Una clase de contrato define constantes que ayudan a las aplicaciones a trabajar con los URI de contenido, nombres de columna, acciones de intención y otras características de un proveedor de contenido. Las clases de contrato no se incluyen automáticamente con un proveedor; El desarrollador del proveedor tiene que definirlos y luego ponerlos a disposición de otros desarrolladores.

Un proveedor generalmente tiene una sola autoridad, que sirve como su nombre interno de Android. Para evitar conflictos con otros proveedores, use una autoridad de contenido única. Debido a que esta recomendación también es válida para los nombres de paquetes de Android, puede definir la autoridad de su proveedor como una extensión del nombre del paquete que contiene el proveedor. Por ejemplo, si el nombre de su paquete de Android es `com.example.appname` , debe otorgar a su proveedor la autoridad `com.example.appname.provider` .

```
public class MyContract {
    public static final String CONTENT_AUTHORITY = "com.example.myApp";
```

```
    public static final String PATH_DATATABLE = "dataTable";
    public static final String TABLE_NAME = "dataTable";
}
```

Un URI de contenido es un URI que identifica datos en un proveedor. Los URI de contenido incluyen el nombre simbólico de todo el proveedor (su autoridad) y un nombre que apunta a una tabla o archivo (una ruta). La parte de identificación opcional apunta a una fila individual en una tabla. Cada método de acceso a datos de ContentProvider tiene un URI de contenido como argumento; esto le permite determinar la tabla, fila o archivo a acceder. Definir estos en la clase de contrato.

```
public static final Uri BASE_CONTENT_URI = Uri.parse("content://" + CONTENT_AUTHORITY);
public static final Uri CONTENT_URI =
BASE_CONTENT_URI.buildUpon().appendPath(PATH_DATATABLE).build();

// define all columns of table and common functions required
```

2) Crear la clase de ayuda

Una clase auxiliar administra la creación de bases de datos y la administración de versiones.

```
public class DatabaseHelper extends SQLiteOpenHelper {

    // Increment the version when there is a change in the structure of database
    public static final int DATABASE_VERSION = 1;
    // The name of the database in the filesystem, you can choose this to be anything
    public static final String DATABASE_NAME = "weather.db";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // Called when the database is created for the first time. This is where the
        // creation of tables and the initial population of the tables should happen.
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Called when the database needs to be upgraded. The implementation
        // should use this method to drop tables, add tables, or do anything else it
        // needs to upgrade to the new schema version.
    }
}
```

3) Crear una clase que amplíe la clase ContentProvider

```
public class MyProvider extends ContentProvider {

public DatabaseHelper dbHelper;

public static final UriMatcher matcher = buildUriMatcher();
public static final int DATA_TABLE = 100;
public static final int DATA_TABLE_DATE = 101;
```

Un UriMatcher asigna una autoridad y una ruta a un valor entero. El método `match()` devuelve un valor entero único para un URI (puede ser cualquier número arbitrario, siempre que sea único). Una declaración de cambio elige entre consultar toda la tabla y consultar un solo registro. Nuestro UriMatcher devuelve 100 si el URI es el URI de contenido de la tabla y 101 si el URI apunta a una fila específica dentro de esa tabla. Puede usar el `#` comodín para hacer coincidir con cualquier número y `*` para hacer coincidir con cualquier cadena.

```
public static UriMatcher buildUriMatcher() {
    UriMatcher uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
    uriMatcher.addURI(CONTENT_AUTHORITY, MyContract.PATH_DATATABLE,        DATA_TABLE);
    uriMatcher.addURI(CONTENT_AUTHORITY, MyContract.PATH_DATATABLE + "/#", DATA_TABLE_DATE);
    return uriMatcher;
}
```

IMPORTANTE: el orden de las llamadas `addURI()` importante! El UriMatcher se verá en orden secuencial desde el primer agregado hasta el último. Como los comodines como `#` y `*` son codiciosos, deberá asegurarse de haber ordenado sus URI correctamente. Por ejemplo:

```
uriMatcher.addURI(CONTENT_AUTHORITY, "/example", 1);
uriMatcher.addURI(CONTENT_AUTHORITY, "/*",       2);
```

es el orden correcto, ya que el comparador buscará `/example` primero antes de recurrir a la coincidencia `/*` . Si se invirtieron estas llamadas de método y llamó a `uriMatcher.match("/example")` , ¡UriMatcher dejará de buscar coincidencias una vez que encuentre la ruta `/*` y devolverá el resultado incorrecto!

A continuación, tendrá que anular estas funciones:

onCreate () : Inicialice su proveedor. El sistema Android llama a este método inmediatamente después de crear su proveedor. Observe que su proveedor no se crea hasta que un objeto ContentResolver intente acceder a él.

```
@Override
public boolean onCreate() {
    dbhelper = new DatabaseHelper(getContext());
    return true;
}
```

getType () : devuelve el tipo MIME correspondiente a un URI de contenido

```
@Override
public String getType(Uri uri) {
    final int match = matcher.match(uri);
    switch (match) {
        case DATA_TABLE:
            return ContentResolver.CURSOR_DIR_BASE_TYPE + "/" + MyContract.CONTENT_AUTHORITY +
"/" + MyContract.PATH_DATATABLE;
        case DATA_TABLE_DATE:
            return ContentResolver.ANY_CURSOR_ITEM_TYPE + "/" + MyContract.CONTENT_AUTHORITY +
"/" + MyContract.PATH_DATATABLE;
        default:
            throw new UnsupportedOperationException("Unknown Uri: " + uri);
```

```
    }
}
```

consulta () : recupera datos de tu proveedor. Use los argumentos para seleccionar la tabla a consultar, las filas y columnas para regresar, y el orden de clasificación del resultado. Devuelve los datos como un objeto Cursor.

```
@Override
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
String sortOrder) {
    Cursor retCursor = dbHelper.getReadableDatabase().query(
        MyContract.TABLE_NAME, projection, selection, selectionArgs, null, null, sortOrder);
    retCursor.setNotificationUri(getContext().getContentResolver(), uri);
    return retCursor;
}
```

Inserte una nueva fila en su proveedor. Use los argumentos para seleccionar la tabla de destino y para obtener los valores de columna para usar. Devuelve un URI de contenido para la fila recién insertada.

```
@Override
public Uri insert(Uri uri, ContentValues values)
{
    final SQLiteDatabase db = dbHelper.getWritableDatabase();
    long id = db.insert(MyContract.TABLE_NAME, null, values);
    return ContentUris.withAppendedId(MyContract.CONTENT_URI, ID);
}
```

delete () : borra filas de tu proveedor. Usa los argumentos para seleccionar la tabla y las filas para eliminar. Devuelve el número de filas borradas.

```
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    int rowsDeleted = db.delete(MyContract.TABLE_NAME, selection, selectionArgs);
    getContext().getContentResolver().notifyChange(uri, null);
    return rowsDeleted;
}
```

update () : actualice las filas existentes en su proveedor. Utilice los argumentos para seleccionar la tabla y las filas para actualizar y obtener los nuevos valores de columna. Devuelve el número de filas actualizadas.

```
@Override
public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    int rowsUpdated = db.update(MyContract.TABLE_NAME, values, selection, selectionArgs);
    getContext().getContentResolver().notifyChange(uri, null);
    return rowsUpdated;
}
```

4) Actualizar archivo de manifiesto

```
<provider
        android:authorities="com.example.myApp"
        android:name=".DatabaseProvider"/>
```

Lea Proveedor de contenido en línea: https://riptutorial.com/es/android/topic/3075/proveedor-de-contenido

Capítulo 198: Prueba de interfaz de usuario con espresso

Observaciones

Café exprés

La hoja de trucos Espresso te ayudará a escribir tus pruebas y lo que quieres probar:

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

También siempre un buen lugar de referencia es la documentación oficial:

https://google.github.io/android-testing-support-library/docs/espresso/index.html

Sugerencias avanzadas de video expreso de Google:
https://www.youtube.com/watch?v=isihPOY2vS4

Solución de problemas

- Cuando intente desplazarse, asegúrese de cerrar el teclado primero:

Vigilancia: no usar la versión "Espresso" no hará nada cuando se use fuera de ViewAction. Esto puede no ser obvio si tiene una importación en la versión de ViewAction ya que tienen exactamente el mismo nombre de método.

```
ViewActions.closeSoftKeyboard;
Espresso.closeSoftKeyboard();
```

- Al ejecutar pruebas juntas en una suite en lugar de individualmente, tenga en cuenta que la Actividad de la prueba anterior todavía puede estar ejecutándose. No confíe en que se haya llamado a onDestroy () de la prueba anterior antes de las pruebas actuales enResume (). **Resulta que esto es realmente un error** : http://b.android.com/201513

Examples

Preparar espresso

En el archivo `build.gradle` de su módulo de aplicación de Android agregue las siguientes dependencias:

```
dependencies {
    // Android JUnit Runner
```

```
    androidTestCompile 'com.android.support.test:runner:0.5'
    // JUnit4 Rules
    androidTestCompile 'com.android.support.test:rules:0.5'
    // Espresso core
    androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
    // Espresso-contrib for DatePicker, RecyclerView, Drawer actions, Accessibility checks,
CountingIdlingResource
    androidTestCompile 'com.android.support.test.espresso:espresso-contrib:2.2.2'
    //UI Automator tests
    androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.2.2'
}
```

Especificar el `AndroidJUnitRunner` para la `testInstrumentationRunner` parámetro en el `build.gradle` archivo.

```
android {

  defaultConfig {
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }

}
```

Además, agregue esta dependencia para proporcionar soporte burlón intencional

```
androidTestCompile 'com.android.support.test.espresso:espresso-intents:2.2.2'
```

Y agrega este para soporte de prueba de webview

```
// Espresso-web for WebView support
androidTestCompile 'com.android.support.test.espresso:espresso-web:2.2.2'
```

Crear clase de prueba de espresso

Coloque la siguiente clase java en **src / android Test / java** y ejecútelo.

```
public class UITest {

  @Test public void Simple_Test() {
    onView(withId(R.id.my_view))         // withId(R.id.my_view) is a ViewMatcher
        .perform(click())                // click() is a ViewAction
        .check(matches(isDisplayed()));  // matches(isDisplayed()) is a ViewAssertion
  }

}
```

Abrir Cerrar CajónDisposición

```
public final class DrawerLayoutTest  {

  @Test public void Open_Close_Drawer_Layout() {
    onView(withId(R.id.drawer_layout)).perform(actionOpenDrawer());
    onView(withId(R.id.drawer_layout)).perform(actionCloseDrawer());
```

```
  }

  public static ViewAction actionOpenDrawer() {
    return new ViewAction() {
      @Override public Matcher<View> getConstraints() {
        return isAssignableFrom(DrawerLayout.class);
      }

      @Override public String getDescription() {
        return "open drawer";
      }

      @Override public void perform(UiController uiController, View view) {
        ((DrawerLayout) view).openDrawer(GravityCompat.START);
      }
    };
  }

  public static ViewAction actionCloseDrawer() {
    return new ViewAction() {
      @Override public Matcher<View> getConstraints() {
        return isAssignableFrom(DrawerLayout.class);
      }

      @Override public String getDescription() {
        return "close drawer";
      }

      @Override public void perform(UiController uiController, View view) {
        ((DrawerLayout) view).closeDrawer(GravityCompat.START);
      }
    };
  }

}
```

Prueba de IU simple expreso

Herramientas de prueba de interfaz de usuario

Dos herramientas principales que actualmente se utilizan principalmente para las pruebas de UI son Appium y Espresso.

Apio	Café exprés
prueba de caja negra	prueba de caja blanca / gris
lo que ves es lo que puedes probar	puede cambiar el funcionamiento interno de la aplicación y prepararla para la prueba, por ejemplo, guardar algunos datos en la base de datos o las preferencias compartidas antes de ejecutar la prueba
Se utiliza principalmente para pruebas de integración de extremo a extremo y flujos completos de	Probando la funcionalidad de una pantalla y / o flujo.

Apio	Café exprés
usuarios.	
se puede abstraer para que la prueba escrita se pueda ejecutar en iOS y Android	Solo Android
bien apoyado	bien apoyado
Soporta pruebas en paralelo en múltiples dispositivos con rejilla de selenio.	No fuera de la caja pruebas paralelas, existen complementos como Spoon hasta que sale el verdadero soporte de Google

Cómo agregar espresso al proyecto

```
dependencies {
  // Set this dependency so you can use Android JUnit Runner
  androidTestCompile 'com.android.support.test:runner:0.5'
  // Set this dependency to use JUnit 4 rules
  androidTestCompile 'com.android.support.test:rules:0.5'
  // Set this dependency to build and run Espresso tests
  androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
  // Set this dependency to build and run UI Automator tests
  androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.2.2'
}
```

NOTA Si está utilizando las últimas bibliotecas de soporte, anotaciones, etc., debe excluir las versiones anteriores de espresso para evitar colisiones:

```
    // there is a conflict with the test support library (see
http://stackoverflow.com/questions/29857695)
    // so for now re exclude the support-annotations dependency from here to avoid clashes
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2') {
        exclude group: 'com.android.support', module: 'support-annotations'
        exclude module: 'support-annotations'
        exclude module: 'recyclerview-v7'
        exclude module: 'support-v4'
        exclude module: 'support-v7'
    }
    // exclude a couple of more modules here because of
<http://stackoverflow.com/questions/29216327> and
    // more specifically of <https://code.google.com/p/android-test-kit/issues/detail?id=139>
    // otherwise you'll receive weird crashes on devices and dex exceptions on emulators
    // Espresso-contrib for DatePicker, RecyclerView, Drawer actions, Accessibility checks,
CountingIdlingResource
    androidTestCompile('com.android.support.test.espresso:espresso-contrib:2.2.2') {
        exclude group: 'com.android.support', module: 'support-annotations'
        exclude group: 'com.android.support', module: 'design'
        exclude module: 'support-annotations'
        exclude module: 'recyclerview-v7'
        exclude module: 'support-v4'
        exclude module: 'support-v7'
    }
```

```
    //excluded specific packages due to
https://code.google.com/p/android/issues/detail?id=183454
    androidTestCompile('com.android.support.test.espresso:espresso-intents:2.2.2') {
        exclude group: 'com.android.support', module: 'support-annotations'
        exclude module: 'support-annotations'
        exclude module: 'recyclerview-v7'
        exclude module: 'support-v4'
        exclude module: 'support-v7'
    }

    androidTestCompile('com.android.support.test.espresso:espresso-web:2.2.2') {
        exclude group: 'com.android.support', module: 'support-annotations'
        exclude module: 'support-annotations'
        exclude module: 'recyclerview-v7'
        exclude module: 'support-v4'
        exclude module: 'support-v7'
    }

    androidTestCompile('com.android.support.test:runner:0.5') {
        exclude group: 'com.android.support', module: 'support-annotations'
        exclude module: 'support-annotations'
        exclude module: 'recyclerview-v7'
        exclude module: 'support-v4'
        exclude module: 'support-v7'
    }
    androidTestCompile('com.android.support.test:rules:0.5') {
        exclude group: 'com.android.support', module: 'support-annotations'
        exclude module: 'support-annotations'
        exclude module: 'recyclerview-v7'
        exclude module: 'support-v4'
        exclude module: 'support-v7'
    }
```

Aparte de estas importaciones, es necesario agregar el corredor de pruebas de instrumentación de Android a build.gradle android.defaultConfig:

```
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
```

Configuración de dispositivo

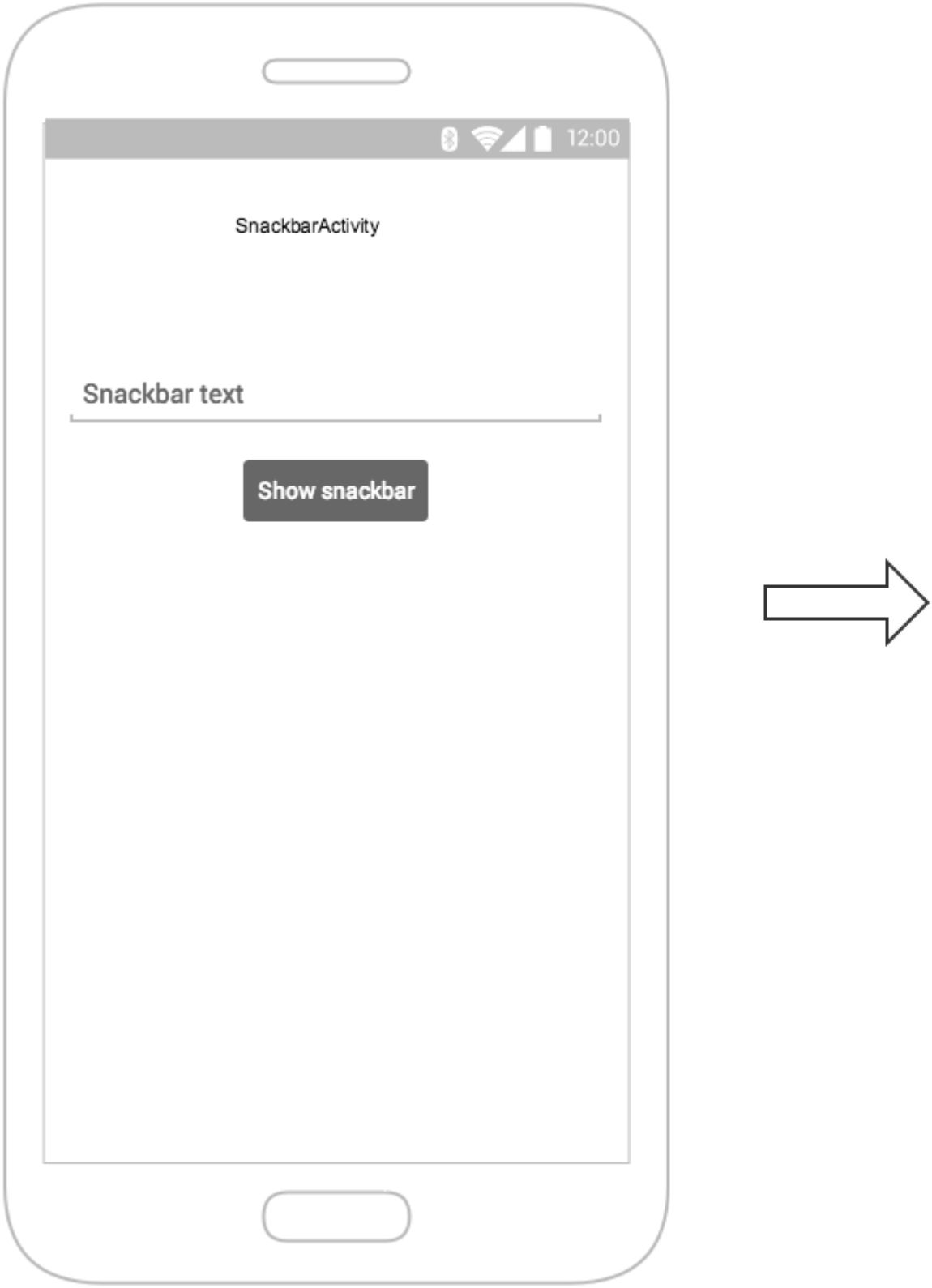
Para pruebas no escamosas, se recomienda establecer las siguientes configuraciones en sus dispositivos:

- Opciones de desarrollador / Deshabilitar animaciones - reduce la descamación de las pruebas
- Opciones de desarrollador / Manténgase despierto: si tiene dispositivos dedicados para pruebas, esto es útil
- Opciones de desarrollador / Tamaños de búfer del registrador: establezca un número más alto si ejecuta conjuntos de pruebas muy grandes en su teléfono
- Accesibilidad / Retardo de toque y retención: largo para evitar problemas con el toque en el espresso

Bastante una configuración del mundo real ha? Bueno, ahora que está fuera del camino, veamos cómo configurar una pequeña prueba.

Escribiendo la prueba

Supongamos que tenemos la siguiente pantalla:



La pantalla contiene:

- campo de entrada de texto - **R.id.textEntry**
- botón que muestra snackbar con texto escrito cuando se hace clic -

R.id.shownSnackbarBtn

- snackbar que debe contener texto escrito por el usuario - **android.support.design.R.id.snackbar_text**

Ahora vamos a crear una clase que probará nuestro flujo:

```
/**
* Testing of the snackbar activity.
**/
@RunWith(AndroidJUnit4.class)
@LargeTest
public class SnackbarActivityTest{
    //espresso rule which tells which activity to start
    @Rule
    public final ActivityTestRule<SnackbarActivity> mActivityRule =
        new ActivityTestRule<>(SnackbarActivity.class, true, false);


    @Override
    public void tearDown() throws Exception {
        super.tearDown();
        //just an example how tear down should cleanup after itself
        mDatabase.clear();
        mSharedPrefs.clear();
    }

    @Override
    public void setUp() throws Exception {
        super.setUp();
        //setting up your application, for example if you need to have a user in shared
        //preferences to stay logged in you can do that for all tests in your setup
        User mUser = new User();
        mUser.setToken("randomToken");
    }

    /**
    *Test methods should always start with "testXYZ" and it is a good idea to
    *name them after the intent what you want to test
    **/
    @Test
    public void testSnackbarIsShown() {
        //start our activity
        mActivityRule.launchActivity(null);
        //check is our text entry displayed and enter some text to it
        String textToType="new snackbar text";
        onView(withId(R.id.textEntry)).check(matches(isDisplayed()));
        onView(withId(R.id.textEntry)).perform(typeText(textToType));
        //click the button to show the snackbar
        onView(withId(R.id.shownSnackbarBtn)).perform(click());
        //assert that a view with snackbar_id with text which we typed and is displayed
        onView(allOf(withId(android.support.design.R.id.snackbar_text),
        withText(textToType))) .check(matches(isDisplayed()));
    }
}
```

Como te habrás dado cuenta, hay 3-4 cosas que podrías notar que vienen a menudo:

onView (withXYZ) <- viewMatchers con ellos puedes encontrar elementos en la pantalla

realizar (clic ()) <- verAcciones, puede ejecutar acciones en elementos que encontró anteriormente

cheque (coincide (isDisplayed ())) <- viewAssertions, cheques que desea hacer en las pantallas que encontró anteriormente

Todos estos y muchos otros se pueden encontrar aquí: https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/index.html

Eso es todo, ahora puede ejecutar la prueba haciendo clic derecho en el nombre de la clase / prueba y seleccionando Ejecutar prueba o con el comando:

```
./gradlew connectedFLAVORNAMEAndroidTest
```

Arriba navegación

```
@Test
public void testUpNavigation() {
    intending(hasComponent(ParentActivity.class.getName())).respondWith(new
Instrumentation.ActivityResult(0, null));

    onView(withContentDescription("Navigate up")).perform(click());

    intended(hasComponent(ParentActivity.class.getName()));
}
```

Tenga en cuenta que esta es una solución alternativa y que chocará con otras Vistas que tienen la misma descripción de contenido.

Realizar una acción en una vista

Es posible realizar `ViewActions` en una vista usando el método de ejecución.
La clase `ViewActions` proporciona métodos de ayuda para las acciones más comunes, como:

```
ViewActions.click()
ViewActions.typeText()
ViewActions.clearText()
```

Por ejemplo, para hacer clic en la vista:

```
onView(...).perform(click());
onView(withId(R.id.button_simple)).perform(click());
```

Puede ejecutar más de una acción con una llamada de ejecución:

```
onView(...).perform(typeText("Hello"), click());
```

Si la vista con la que está trabajando se encuentra dentro de un `ScrollView` (vertical u horizontal), considere las acciones anteriores que requieren que la vista se muestre (como `click()` y

`typeText()`) con `scrollTo()` . Esto asegura que la vista se muestre antes de continuar con la otra acción:

```
onView(...).perform(scrollTo(), click());
```

Encontrar una vista con onView

Con los `ViewMatchers` puede encontrar la vista en la jerarquía de vista actual.

Para encontrar una vista, use el método `onView()` con un comparador de vista que seleccione la vista correcta. Los métodos `onView()` devuelven un objeto de tipo `ViewInteraction` .

Por ejemplo, encontrar una vista por su `R.id` es tan simple como:

```
onView(withId(R.id.my_view))
```

Encontrando una vista con un texto:

```
onView(withText("Hello World"))
```

Cafeteras personalizadas espresso

El espresso por defecto tiene muchos emparejadores que lo ayudan a encontrar vistas que necesita para hacer algunas comprobaciones o interacciones con ellas.

Los más importantes se pueden encontrar en la siguiente hoja de trucos:

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Algunos ejemplos de matchers son:

- withId (R.id.ID_of_object_you_are_looking_for);
- withText ("Algún texto que esperas que tenga el objeto");
- isDisplayed () <- verifique si la vista está visible
- doesNotExist () <- comprueba que la vista no existe

Todos estos son muy útiles para el uso diario, pero si tiene vistas más complejas, escribir sus emparejadores personalizados puede hacer que las pruebas sean más legibles y pueda reutilizarse en diferentes lugares.

Hay 2 tipos de emparejadores más comunes que puede extender: **TypeSafeMatcher BoundedMatcher**

La implementación de TypeSafeMatcher requiere que verifique la instancia de la vista contra la que está afirmando, si es del tipo correcto que coincida con algunas de sus propiedades con el valor que proporcionó a un comparador.

Por ejemplo, el tipo de coincidencia segura que valida una vista de imagen tiene dibujable

correcto:

```
public class DrawableMatcher extends TypeSafeMatcher<View> {

    private @DrawableRes final int expectedId;
    String resourceName;

    public DrawableMatcher(@DrawableRes int expectedId) {
        super(View.class);
        this.expectedId = expectedId;
    }

    @Override
    protected boolean matchesSafely(View target) {
        //Type check we need to do in TypeSafeMatcher
        if (!(target instanceof ImageView)) {
            return false;
        }
        //We fetch the image view from the focused view
        ImageView imageView = (ImageView) target;
        if (expectedId < 0) {
            return imageView.getDrawable() == null;
        }
        //We get the drawable from the resources that we are going to compare with image view
source
        Resources resources = target.getContext().getResources();
        Drawable expectedDrawable = resources.getDrawable(expectedId);
        resourceName = resources.getResourceEntryName(expectedId);

        if (expectedDrawable == null) {
            return false;
        }
        //comparing the bitmaps should give results of the matcher if they are equal
        Bitmap bitmap = ((BitmapDrawable) imageView.getDrawable()).getBitmap();
        Bitmap otherBitmap = ((BitmapDrawable) expectedDrawable).getBitmap();
        return bitmap.sameAs(otherBitmap);
    }


    @Override
    public void describeTo(Description description) {
        description.appendText("with drawable from resource id: ");
        description.appendValue(expectedId);
        if (resourceName != null) {
            description.appendText("[");
            description.appendText(resourceName);
            description.appendText("]");
        }
    }
}
```

El uso del emparejador podría ser envuelto así:

```
  public static Matcher<View> withDrawable(final int resourceId) {
    return new DrawableMatcher(resourceId);
}

 onView(withDrawable(R.drawable.someDrawable)).check(matches(isDisplayed()));
```

Los emparejadores limitados son similares, simplemente no tiene que hacer la verificación de tipo pero, como eso se hace automágicamente para usted:

```
 /**
 * Matches a {@link TextInputFormView}'s input hint with the given resource ID
 *
 * @param stringId
 * @return
 */
public static Matcher<View> withTextInputHint(@StringRes final int stringId) {
    return new BoundedMatcher<View, TextInputFormView>(TextInputFormView.class) {
        private String mResourceName = null;

        @Override
        public void describeTo(final Description description) {
            //fill these out properly so your logging and error reporting is more clear
            description.appendText("with TextInputFormView that has hint ");
            description.appendValue(stringId);
            if (null != mResourceName) {
                description.appendText("[");
                description.appendText(mResourceName);
                description.appendText("]");
            }
        }

        @Override
        public boolean matchesSafely(final TextInputFormView view) {
            if (null == mResourceName) {
                try {
                    mResourceName = view.getResources().getResourceEntryName(stringId);
                } catch (Resources.NotFoundException e) {
                    throw new IllegalStateException("could not find string with ID " +
stringId, e);
                }
            }
            return view.getResources().getString(stringId).equals(view.getHint());
        }
    };
}
```

Más sobre los matchers se puede leer en:

http://hamcrest.org/

https://developer.android.com/reference/android/support/test/espresso/matcher/ViewMatchers.html

Espresso general

Espresso de configuración:

```
androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
androidTestCompile 'com.android.support.test:runner:0.5'
```

ViewMatchers : una colección de objetos que implementan `Matcher<? super View>` interfaz. Puede pasar uno o más de estos al método `onView` para ubicar una vista dentro de la jerarquía de vista

actual.

ViewActions : una colección de `ViewActions` que se puede pasar al método `ViewInteraction.perform()` (por ejemplo, `click()`).

ViewAssertions : una colección de `ViewAssertions` que se puede pasar el método `ViewInteraction.check()` . La mayoría de las veces, utilizará la aserción de coincidencias, que utiliza un igualador de vista para afirmar el estado de la vista seleccionada actualmente.

Espresso cheat sheet de google

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion);
```

View Matchers

USER PROPERTIES

```
withId(...)
withText(...)
withTagKey(...)
withTagValue(...)
hasContentDescription(...)
withContentDescription(...)
withHint(...)
withSpinnerText(...)
hasLinks()
hasEllipsizedText()
hasMultilineTest()
```

HIERARCHY

```
withParent(Matcher)
withChild(Matcher)
hasDescendant(Matcher)
isDescendantOfA(Matcher)
hasSibling(Matcher)
isRoot()
```

INPUT

```
supportsInputMethods(...)
hasIMEAction(...)
```

UI PROPERTIES

```
isDisplayed()
isCompletelyDisplayed()
isEnabled()
hasFocus()
isClickable()
isChecked()
isNotChecked()
withEffectiveVisibility(...)
isSelected()
```

CLASS

```
isAssignableFrom(...)
withClassName(...)
```

ROOT MATCHERS

```
isFocusable()
isTouchable()
isDialog()
withDecorView()
isPlatformPopup()
```

OBJECT MATCHER

```
allOf(Matchers)
anyOf(Matchers)
is(...)
```

SEE ALSO

Preference matchers

https://riptutorial.com/es/android/topic/3485/prueba-de-interfaz-de-usuario-con-espresso

Capítulo 199: Pruebas unitarias en Android con JUnit.

Observaciones

- Vogella: Pruebas unitarias con JUnit
- Anotaciones de Junit: java2novice.com
- Clase de afirmación : junit.org
- JUnit Api: tutorialspoint.com
- Anroid probando posts de Medium.com

Examples

Creando pruebas unitarias locales

Coloque sus clases de prueba aquí: `/src/test/<pkg_name>/`

Ejemplo de clase de prueba

```
public class ExampleUnitTest {
    @Test
    public void addition_isCorrect() throws Exception {
        int a=4, b=5, c;
        c = a + b;
        assertEquals(9, c); // This test passes
        assertEquals(10, c); //Test fails
    }
}
```

Descompostura

```
public class ExampleUnitTest {
  ...
}
```

La clase de prueba, puede crear varias clases de prueba y colocarlas dentro del paquete de prueba.

```
@Test
public void addition_isCorrect() {
   ...
}
```

El método de prueba, varios métodos de prueba se pueden crear dentro de una clase de prueba.

Observe la anotación `@Test` .

> La anotación de prueba le dice a JUnit que el método de anulación público al que se adjunta se puede ejecutar como un caso de prueba.

Hay varias otras anotaciones útiles como `@Before` , `@After` etc. Esta página sería un buen lugar para comenzar.

```
assertEquals(9, c); // This test passes
assertEquals(10, c); //Test fails
```

Estos métodos son miembros de la clase `Assert` . Algunos otros métodos útiles son `assertFalse()` , `assertNotNull()` , `assertTrue` etc. Aquí hay una explicación detallada.

Información de anotación para JUnit Test:

Prueba @: La anotación de prueba le dice a JUnit que el método de anulación público al que está adjunto se puede ejecutar como un caso de prueba. Para ejecutar el método, JUnit primero construye una nueva instancia de la clase y luego invoca el método anotado.

@Antes de: Al escribir pruebas, es común encontrar que varias pruebas necesitan que se creen objetos similares antes de que puedan ejecutarse. La anotación de un método void público con `@Before` hace que ese método se ejecute antes que el método Test.

@ Después **:** si asigna recursos externos en un método Antes, debe liberarlos después de que se ejecute la prueba. La anotación de un método de vacío público con `@After` hace que ese método se ejecute después del método de prueba. Se garantiza que todos los métodos `@After` se ejecutarán incluso si un método Antes o Prueba arroja una excepción

---

Consejo: crea rápidamente clases de prueba en Android Studio

- Coloque el cursor en el nombre de la clase para la que desea crear una clase de prueba.
- Presione Alt + Enter (Windows).
- Seleccione Crear prueba, pulse Retorno.
- Seleccione los métodos para los que desea crear métodos de prueba, haga clic en Aceptar.
- Seleccione el directorio donde desea crear la clase de prueba.
- Ya terminaste, esto lo que obtienes es tu primera prueba.

Sugerencia: Ejecutar pruebas fácilmente en Android Studio.

- Haga clic derecho para probar el paquete.
- Seleccione Run 'Tests en ...
- Todas las pruebas en el paquete se ejecutarán a la vez.

Moviendo la lógica de negocios fuera de los componentes de Android

Gran parte del valor de las pruebas unitarias de JVM locales proviene de la forma en que diseña su aplicación. Debe diseñarlo de tal manera que pueda desacoplar la lógica de su negocio de sus componentes de Android. Este es un ejemplo de tal manera de usar el patrón Model-View-Presenter . Practiquemos esto implementando una pantalla de registro básica que solo requiere un nombre de usuario y contraseña. Nuestra aplicación de Android es responsable de validar que el nombre de usuario que el usuario proporciona no está en blanco y que la contraseña tiene al menos ocho caracteres y contiene al menos un dígito. Si el nombre de usuario / contraseña es válido, realizamos nuestra llamada a la API de registro, de lo contrario, mostramos un mensaje de error.

Ejemplo donde la lógica de negocios está altamente acoplada con el componente Android.

```
public class LoginActivity extends Activity{
    ...
    private void onSubmitButtonClicked(){
        String username = findViewById(R.id.username).getText().toString();
        String password = findViewById(R.id.password).getText().toString();
        boolean isUsernameValid = username != null && username.trim().length() != 0;
        boolean isPasswordValid = password != null && password.trim().length() >= 8 &&
password.matches(".*\\d+.*");
        if(isUsernameValid && isPasswordValid){
            performSignUpApiCall(username, password);
        } else {
            displayInvalidCredentialsErrorMessage();
        }
    }
}
```

Ejemplo donde se desacopla la lógica de negocios del componente Android.

Aquí definimos en una sola clase, LoginContract, que albergará las diversas interacciones entre nuestras diversas clases.

```
public interface LoginContract {
    public interface View {
        performSignUpApiCall(String username, String password);
        displayInvalidCredentialsErrorMessage();
    }
    public interface Presenter {
        void validateUserCredentials(String username, String password);
    }
}
```

Nuestra actividad de inicio de sesión es, en su mayor parte, la misma, excepto que hemos eliminado la responsabilidad de tener que saber cómo validar el formulario de registro de un usuario (nuestra lógica empresarial). LoginActivity ahora se basará en nuestro nuevo LoginPresenter para realizar la validación.

```
public class LoginActivity extends Activity implements LoginContract.View{
    private LoginContract.Presenter presenter;
```

```
    protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            presenter = new LoginPresenter(this);
            ....
        }
        ...

        private void onSubmitButtonClicked(){
            String username = findViewById(R.id.username).getText().toString();
            String password = findViewById(R.id.password).getText().toString();
            presenter.validateUserCredentials(username, password);
    }
    ...
}
```

Ahora su lógica de negocios residirá en su nueva clase LoginPresenter.

```
public class LoginPresenter implements LoginContract.Presenter{
    private LoginContract.View view;

    public LoginPresenter(LoginContract.View view){
        this.view = view;
    }

    public void validateUserCredentials(String username, String password){
        boolean isUsernameValid = username != null && username.trim().length() != 0;
        boolean isPasswordValid = password != null && password.trim().length() >= 8 &&
password.matches(".*\\d+.*");
        if(isUsernameValid && isPasswordValid){
            view.performSignUpApiCall(username, password);
        } else {
            view.displayInvalidCredentialsErrorMessage();
        }
    }
}
```

Y ahora podemos crear pruebas locales de unidades JVM contra su nueva clase LoginPresenter.

```
public class LoginPresenterTest {

    @Mock
    LoginContract.View view;

    private LoginPresenter presenter;

    @Before
    public void setUp() throws Exception {
        MockitoAnnotations.initMocks(this);
        presenter = new LoginPresenter(view);
    }

    @Test
    public void test_validateUserCredentials_userDidNotEnterUsername_displayErrorMessage()
throws Exception {
        String username = "";
        String password = "kingslayer1";
        presenter.validateUserCredentials(username, password);
        Mockito.verify(view). displayInvalidCredentialsErrorMessage();
```

```
    }

    @Test
    public void
test_validateUserCredentials_userEnteredFourLettersAndOneDigitPassword_displayErrorMessage()
throws Exception {
        String username = "Jaime Lanninster";
        String password = "king1";
        presenter.validateUserCredentials(username, password);
        Mockito.verify(view). displayInvalidCredentialsErrorMessage();
    }

    @Test
    public void
test_validateUserCredentials_userEnteredNineLettersButNoDigitsPassword_displayErrorMessage()
throws Exception {
        String username = "Jaime Lanninster";
        String password = "kingslayer";
        presenter.validateUserCredentials(username, password);
        Mockito.verify(view). displayInvalidCredentialsErrorMessage();
    }

    @Test
    public void
test_validateUserCredentials_userEnteredNineLettersButOneDigitPassword_performApiCallToSignUpUser()
throws Exception {
        String username = "Jaime Lanninster";
        String password = "kingslayer1";
        presenter.validateUserCredentials(username, password);
        Mockito.verify(view).performSignUpApiCall(username, password);
    }
}
```

Como se puede ver, cuando se extrajeron nuestra lógica de negocio fuera de la LoginActivity y lo colocó en el LoginPresenter POJO . Ahora podemos crear pruebas locales de unidades JVM contra nuestra lógica empresarial.

Se debe tener en cuenta que nuestro cambio en la arquitectura tiene varias otras implicaciones, ya que estamos cerca de adherirnos a cada clase que tiene una responsabilidad única, clases adicionales, etc. Estos son solo efectos secundarios de la forma en que elijo hacerlo. desacoplamiento a través del estilo MVP. MVP es solo una forma de hacer esto, pero hay otras alternativas que tal vez quiera ver, como MVVM . Solo tienes que elegir el mejor sistema que funcione para ti.

Empezando con JUnit

Preparar

Para iniciar la prueba de unidad de su proyecto de Android usando JUnit, debe agregar la dependencia de JUnit a su proyecto y debe crear un conjunto de fuente de prueba que contendrá el código fuente de las pruebas de unidad. Los proyectos creados con Android Studio a menudo ya incluyen la dependencia de JUnit y el conjunto de fuentes de prueba

Agregue la siguiente línea a su módulo `build.gradle` en el `Closure` dependencias:

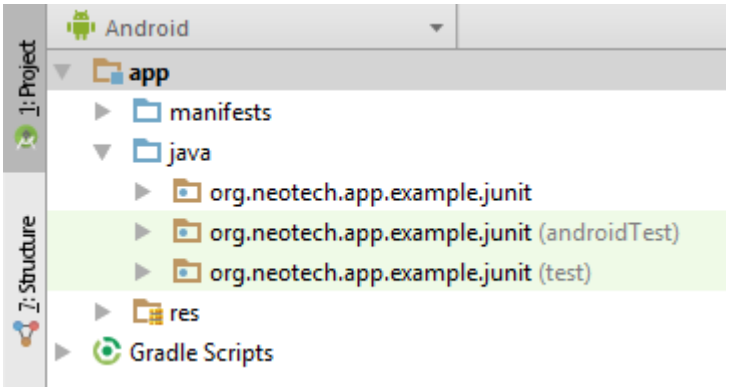
```
testCompile 'junit:junit:4.12'
```

Las clases de prueba de JUnit están ubicadas en un conjunto de fuente especial denominado `test` . Si este conjunto de fuentes no existe, debe crear una nueva carpeta usted mismo. La estructura de carpetas de un proyecto predeterminado de Android Studio (basado en Gradle) se ve así:

```
<project-root-folder>
    /app (module root folder)
        /build
        /libs
        /src
            /main (source code)
            /test (unit test source code)
            /androidTest (instrumentation test source code)
        build.gradle (module gradle file)
    /build
    /gradle
    build.gradle (project gradle file)
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    settings.gradle (gradle settings)
```

Si su proyecto no tiene la carpeta `/app/src/test` , debe crearlo usted mismo. Dentro de la carpeta de `test` también necesita una carpeta `java` (créela si no existe). La carpeta java en el conjunto de fuentes de `test` debe contener la misma estructura de paquetes que su conjunto de fuentes `main` .

Si configura correctamente la estructura de su proyecto (en la vista de Android en Android Studio) debería tener este aspecto:



Nota: No es necesario que tenga el `androidTest` fuentes de `androidTest` , este conjunto de fuentes se encuentra a menudo en proyectos creados por Android Studio y se incluye aquí como referencia.

Escribiendo una prueba

1. **Crear una nueva clase dentro del conjunto de fuentes de `test` .**
   Haga clic con el botón derecho en el conjunto de fuentes de prueba en la vista del proyecto, elija `New` > `Java class` .

El patrón de nomenclatura más utilizado es usar el nombre de la clase que se va a probar con la `Test` agregada. Así que `StringUtilities` convierte en `StringUtilitiesTest` .

2. **Añadir la anotación `@RunWith`**
   La anotación `@RunWith` es necesaria para que JUnit ejecute las pruebas que vamos a definir en nuestra clase de prueba. El corredor de JUnit predeterminado (para JUnit 4) es el `BlockJUnit4ClassRunner` pero en lugar de usar esta ejecución directamente, es más conveniente usar el alias `JUnit4` que es una abreviatura para el corredor de JUnit predeterminado.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

}
```

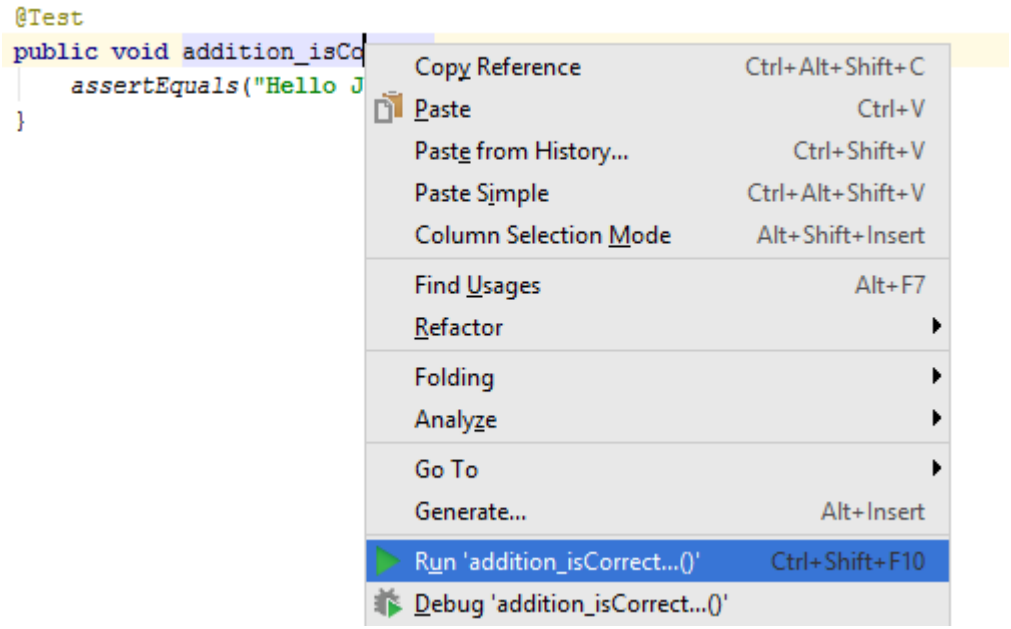
3. **Crear una prueba**
   Una prueba de unidad es esencialmente un método que, en la mayoría de los casos, no debería fallar si se ejecuta. En otras palabras, no debe lanzar una excepción. Dentro de un método de prueba, casi siempre encontrará afirmaciones que verifican si se cumplen las condiciones específicas. Si una aserción falla, lanza una excepción que hace que el método / prueba falle. Un método de prueba siempre se anota con la anotación `@Test` . Sin esta anotación, JUnit no ejecutará automáticamente la prueba.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

    @Test
    public void addition_isCorrect() throws Exception {
        assertEquals("Hello JUnit", "Hello" + " " + "JUnit");
    }
}
```

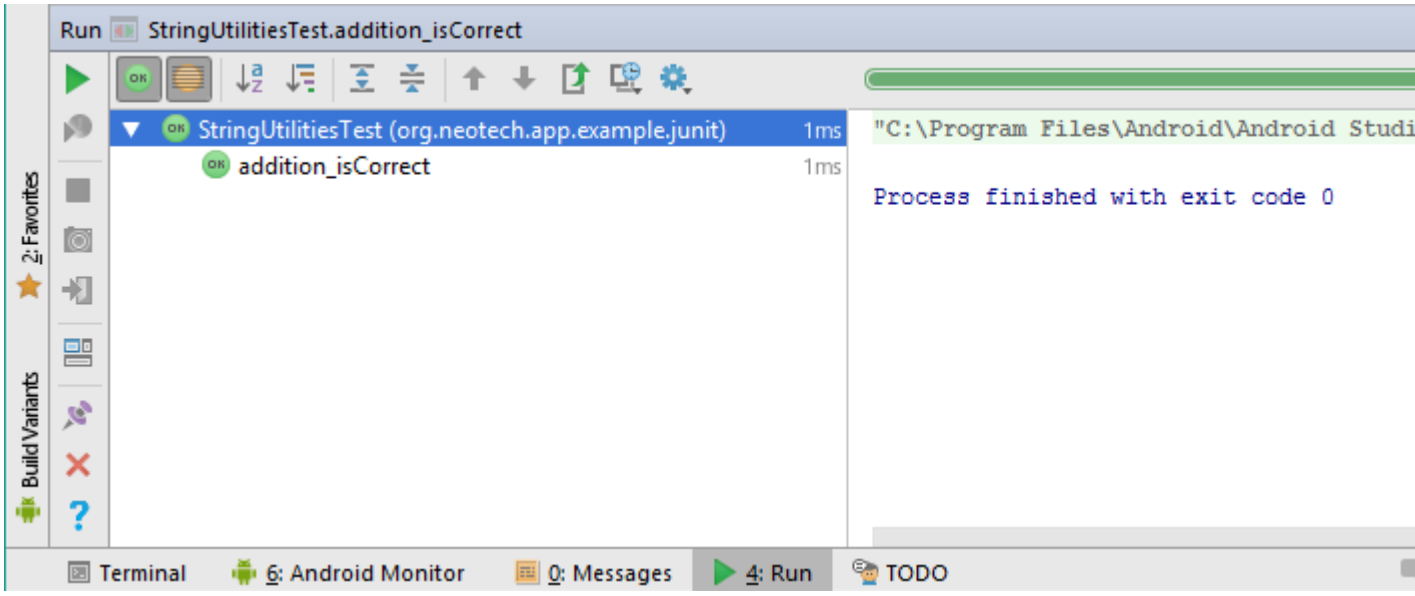
   Nota: a diferencia del método estándar de Java, los nombres de los métodos de prueba de la unidad de convención de nomenclatura suelen contener guiones bajos.

Haciendo una prueba

1. **Método**
   Para ejecutar un solo método de prueba, puede hacer clic con el botón derecho en el método y hacer clic en `Run 'addition_isCorrect()'` o usar el método abreviado de teclado `ctrl+shift+f10` .



Si todo está configurado correctamente, JUnit comienza a ejecutar el método y debería ver la siguiente interfaz dentro de Android Studio:



2. **Clase**
   También puede ejecutar todas las pruebas definidas en una sola clase haciendo clic con el botón derecho en la clase en la vista del proyecto y haciendo clic en `Run 'StringUtilitiesTest '` o use el método abreviado de teclado `ctrl+shift+f10` si ha seleccionado la clase en la vista del proyecto.

3. **Paquete (todo)**
   Si no desea ejecutar todas las pruebas definidas en el proyecto o en un paquete, puede simplemente hacer clic derecho en el paquete y hacer clic en `Run ...` al igual que ejecutaría todas las pruebas definidas en una sola clase.

Excepciones

JUnit también se puede usar para probar si un método lanza una excepción específica para una

entrada determinada.

En este ejemplo, probaremos si el siguiente método realmente produce una excepción si el formato booleano (entrada) no se reconoce / se desconoce:

```
public static boolean parseBoolean(@NonNull String raw) throws IllegalArgumentException{
    raw = raw.toLowerCase().trim();
    switch (raw) {
        case "t": case "yes": case "1": case "true":
            return true;
        case "f": case "no": case "0": case "false":
            return false;
        default:
            throw new IllegalArgumentException("Unknown boolean format: " + raw);
    }
}
```

Al agregar el parámetro `expected` a la anotación `@Test` , se puede definir qué excepción se espera que se genere. La prueba de la unidad fallará si esta excepción no se produce, y tendrá éxito si la excepción es efectivamente lanzada:

```
@Test(expected = IllegalArgumentException.class)
public void parseBoolean_parsesInvalidFormat_throwsException(){
    StringUtilities.parseBoolean("Hello JUnit");
}
```

Esto funciona bien, sin embargo, lo limita a un solo caso de prueba dentro del método. A veces es posible que desee probar varios casos dentro de un solo método. Una técnica que se usa con frecuencia para superar esta limitación es usar `try-catch` bloques `try-catch` y el método `Assert.fail()` :

```
@Test
public void parseBoolean_parsesInvalidFormats_throwsException(){
    try {
        StringUtilities.parseBoolean("Hello!");
        fail("Expected IllegalArgumentException");
    } catch(IllegalArgumentException e){
    }

    try {
        StringUtilities.parseBoolean("JUnit!");
        fail("Expected IllegalArgumentException");
    } catch(IllegalArgumentException e){
    }
}
```

Nota: Algunas personas consideran que es una mala práctica probar más de un solo caso dentro de una prueba unitaria.

Importación estática

JUnit define varios métodos `assertEquals` , al menos uno para cada tipo primitivo y uno para Objetos está disponible. Por defecto, estos métodos no están disponibles directamente para

llamar y deben llamarse así: `Assert.assertEquals` . Pero debido a que estos métodos se usan con mucha frecuencia, la gente casi siempre usa una importación estática para que el método se pueda usar directamente como si fuera parte de la clase en sí.

Para agregar una importación estática para el método `assertEquals` , use la siguiente declaración de importación:

```
import static org.junit.Assert.assertEquals;
```

También puede importar de forma estática todos los métodos de assert, incluyendo `assertArrayEquals` , `assertNotNull` y `assertFalse` etc., utilizando la siguiente importación estática:

```
import static org.junit.Assert.*;
```

Sin importación estática:

```
@Test
public void addition_isCorrect(){
    Assert.assertEquals(4 , 2 + 2);
}
```

Con importación estática:

```
@Test
public void addition_isCorrect(){
    assertEquals(4 , 2 + 2);
}
```

Lea Pruebas unitarias en Android con JUnit. en línea:
https://riptutorial.com/es/android/topic/3205/pruebas-unitarias-en-android-con-junit-

Capítulo 200: Publicar el archivo .aar en Apache Archiva con Gradle

Examples

Ejemplo de implementación simple

```
apply plugin: 'com.android.library'
apply plugin: 'maven'
apply plugin: 'maven-publish'
android {
compileSdkVersion 21
buildToolsVersion "21.1.2"


repositories {
    mavenCentral()
}

defaultConfig {
    minSdkVersion 9
    targetSdkVersion 21
    versionCode 1
    versionName "1.0"
}

buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}


dependencies {
   compile fileTree(include: ['*.jar'], dir: 'libs')
   provided 'com.android.support:support-v4:21.0.3'
   provided 'com.android.support:appcompat-v7:21.0.3'
}

task sourceJar(type: Jar) {
   classifier "source"
}

publishing {
   publications {

       repositories.maven {
           url 'myurl/repositories/myrepo'
           credentials {
               username "user"
               password "password"
           }
       }
```

```
        maven(MavenPublication) {
            artifacts {
                groupId 'com.mycompany'
                artifactId 'mylibrary'
                version '1.0'
                artifact 'build/outputs/aar/app-release.aar'
            }
        }
    }

}
```

Lea Publicar el archivo .aar en Apache Archiva con Gradle en línea:
https://riptutorial.com/es/android/topic/6453/publicar-el-archivo--aar-en-apache-archiva-con-gradle

Capítulo 201: Publicar en Play Store

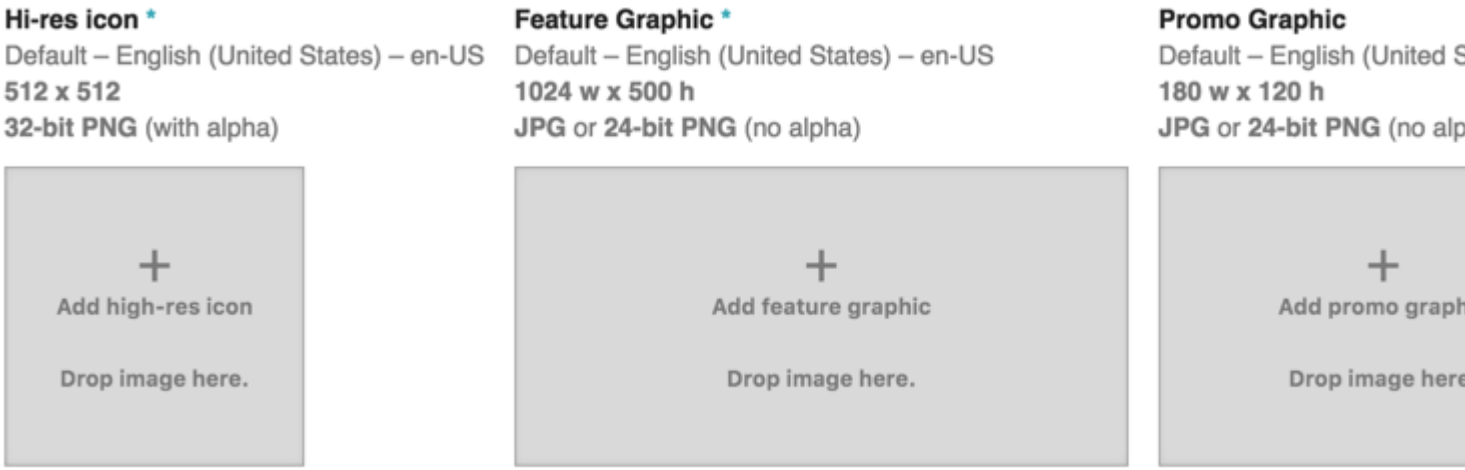
Examples

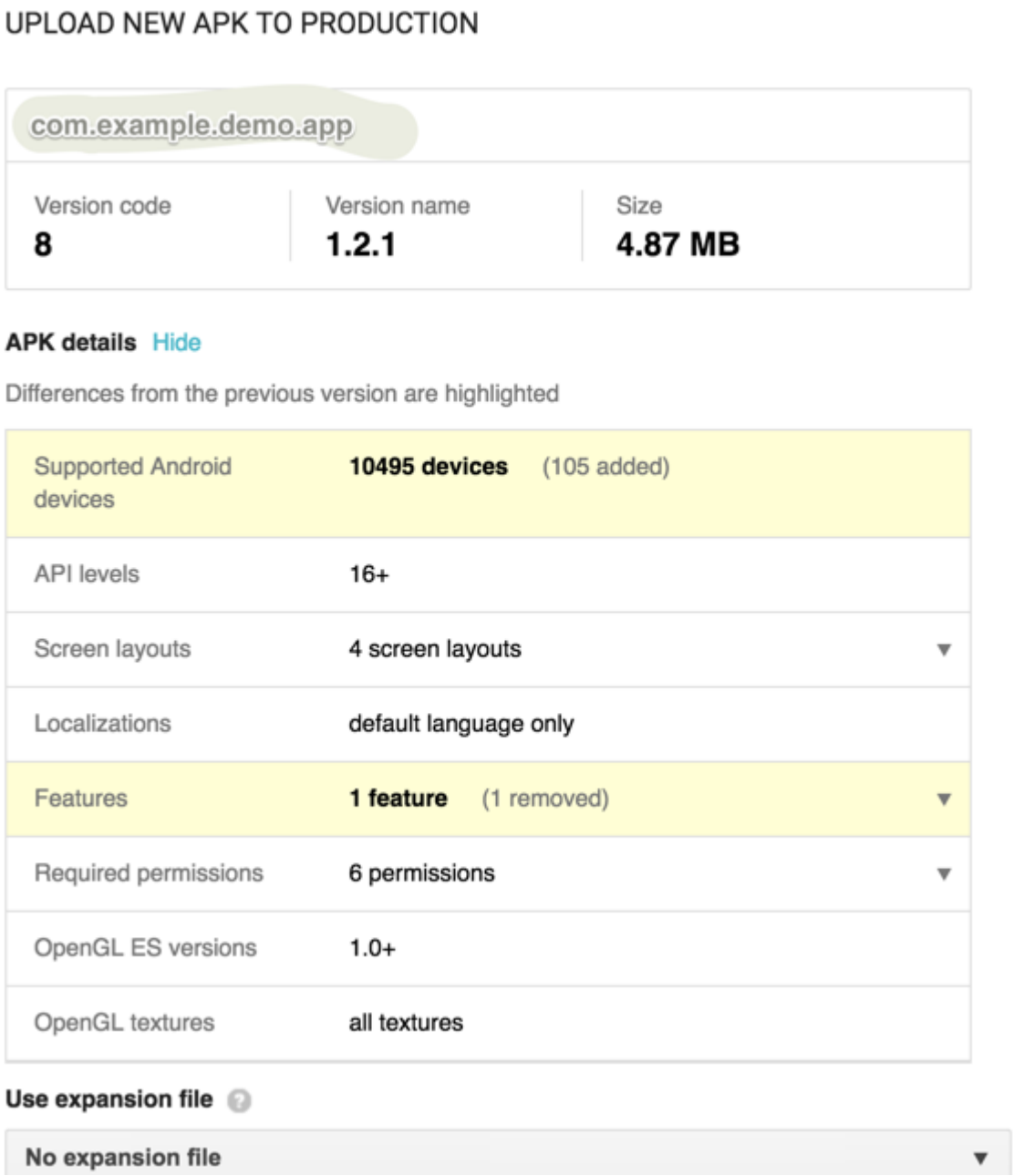
Guía de envío de aplicaciones mínimas

Requisitos:

- Una cuenta de desarrollador
- Un apk ya creado y firmado con una clave de no depuración
- Una aplicación gratuita que no tiene facturación en la aplicación
- No Firebase Cloud Messaging o Game Services

1. Dirígete a https://play.google.com/apps/publish/
   1a) Crea tu cuenta de desarrollador si no tienes una
2. Haga clic en el botón `Create new Application`
3. Haga clic en Enviar APK
4. Rellene todos los campos obligatorios del formulario, incluidos algunos recursos que se mostrarán en Play Store (vea la imagen a continuación)
5. Cuando esté satisfecho `Publish app` botón `Publish app`





Ver más sobre cómo iniciar sesión en Configurar ajustes de firma

Lea Publicar en Play Store en línea: https://riptutorial.com/es/android/topic/5369/publicar-en-play-store

Capítulo 202: Publicar una biblioteca en Repositorios Maven

Examples

Publicar archivo .aar a Maven

Para publicar en un repositorio en formato Maven, se puede utilizar el complemento "maven-publish" para gradle.

El complemento debe agregarse al archivo `build.gradle` en el módulo de biblioteca.

```
apply plugin: 'maven-publish'
```

También debe definir la publicación y sus atributos de identidad en el archivo `build.gradle` . Estos atributos de identidad se mostrarán en el archivo pom generado y en el futuro, para importar esta publicación, los usará. También debe definir qué artefactos desea publicar, por ejemplo, solo quiero publicar el archivo .aar generado después de construir la biblioteca. .

```
publishing {
    publications {
        myPulication(MavenPublication) {
            groupId 'com.example.project'
            version '1.0.2'
            artifactId 'myProject'
            artifact("$buildDir/outputs/aar/myProject.aar")
        }
    }
}
```

También necesitarás definir tu url de repositorio.

```
publishing{
    repositories {
        maven {
            url "http://www.myrepository.com"
        }
    }
}
```

Aquí está la biblioteca `build.gradle` archivo `build.gradle`

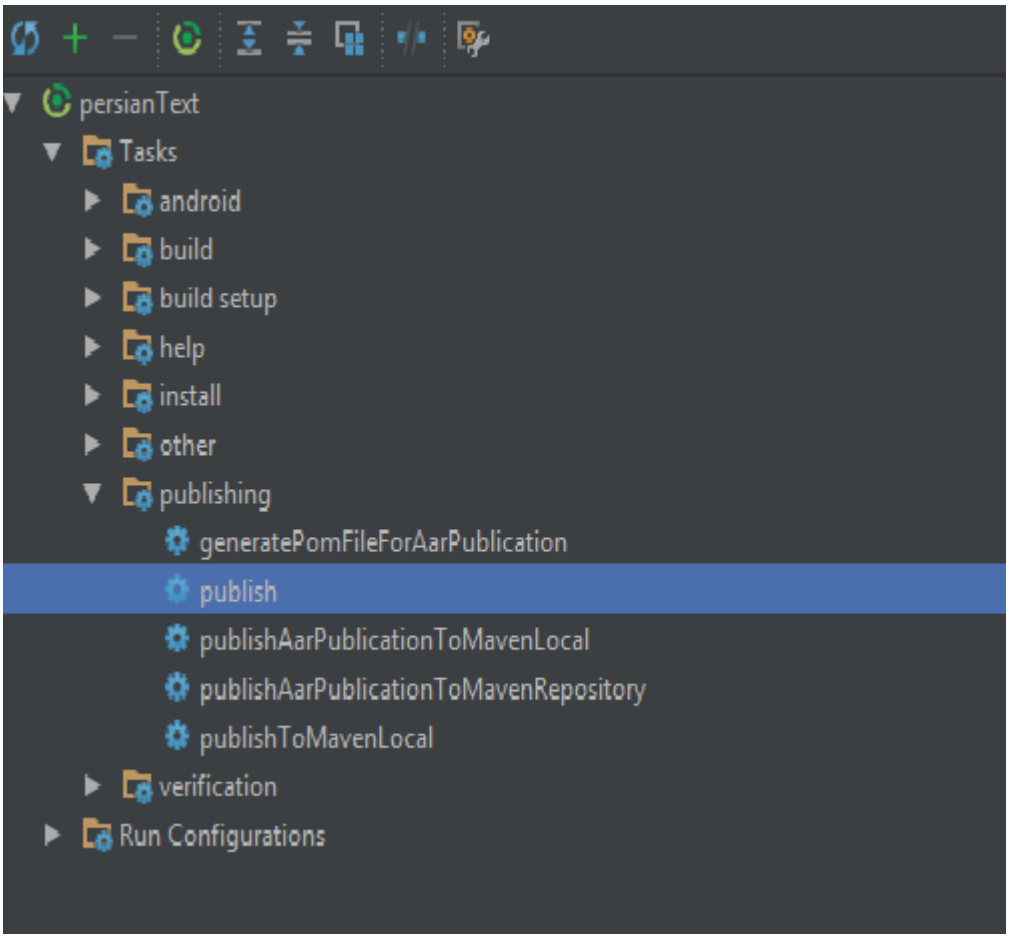
```
apply plugin: 'com.android.library'
apply plugin: 'maven-publish'

buildscript {
   ...
}
android {
```

```
    ...
}
publishing {
    publications {
        myPulication(MavenPublication) {
            groupId 'com.example.project'
            version '1.0.2'
            artifactId 'myProject'
            artifact("$buildDir/outputs/aar/myProject.aar")
        }
    }
    repositories {
        maven {
            url "http://www.myrepository.com"
        }
    }
}
```

Para la publicación puedes ejecutar el comando de la consola de gradle.

```
publicación de gradle
```

o puede ejecutar desde el panel de tareas de Gradle



Lea Publicar una biblioteca en Repositorios Maven en línea:
https://riptutorial.com/es/android/topic/9359/publicar-una-biblioteca-en-repositorios-maven

Capítulo 203: Receptor de radiodifusión

Introducción

BroadcastReceiver (receptor) es un componente de Android que le permite registrarse para eventos del sistema o de la aplicación. Una vez que ocurre este evento, el tiempo de ejecución de Android notifica a todos los receptores registrados para un evento.

por ejemplo, una transmisión que anuncia que la pantalla se apagó, que la batería está baja o que se capturó una imagen.

Las aplicaciones también pueden iniciar transmisiones, por ejemplo, para que otras aplicaciones sepan que algunos datos se han descargado en el dispositivo y están disponibles para su uso.

Examples

Introducción al receptor de radiodifusión

Un receptor de difusión es un componente de Android que le permite registrarse para eventos del sistema o de la aplicación.

Un receptor se puede registrar a través del archivo `AndroidManifest.xml` o dinámicamente a través del método `Context.registerReceiver()` .

```
public class MyReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      //Your implementation goes here.
   }
}
```

Aquí he tomado un ejemplo de `ACTION_BOOT_COMPLETED` que es `ACTION_BOOT_COMPLETED` por el sistema una vez que Android ha completado el proceso de arranque.

Puede registrar un receptor en un archivo de manifiesto como este:

```
<application
   android:icon="@drawable/ic_launcher"
   android:label="@string/app_name"
   android:theme="@style/AppTheme" >
   <receiver android:name="MyReceiver">
      <intent-filter>
         <action android:name="android.intent.action.BOOT_COMPLETED">
         </action>
      </intent-filter>
   </receiver>
</application>
```

Ahora se `onReceive()` dispositivo, se `onReceive()` método `onReceive()` y luego podrá hacer su

trabajo (por ejemplo, iniciar un servicio, iniciar una alarma).

Fundamentos de BroadcastReceiver

Los BroadcastReceivers se utilizan para recibir los Intentos de transmisión enviados por el sistema operativo Android, otras aplicaciones o dentro de la misma aplicación.

Cada intento se crea con un *filtro de intento* , que requiere una *acción de* cadena. Se puede configurar información adicional en la Intención.

Del mismo modo, BroadcastReceivers se registra para recibir Intents con un Intent Filter particular. Se pueden registrar programáticamente:

```
mContext.registerReceiver(new BroadcastReceiver() {
    @Override
   public void onReceive(Context context, Intent intent) {
      //Your implementation goes here.
   }
}, new IntentFilter("Some Action"));
```

o en el archivo `AndroidManifest.xml` :

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="Some Action"/>
    </intent-filter>
</receiver>
```

Para recibir la Intención, configure la Acción como algo documentado por el sistema operativo Android, por otra aplicación o API, o dentro de su propia aplicación, usando `sendBroadcast` :

```
mContext.sendBroadcast(new Intent("Some Action"));
```

Además, la intención puede contener información, como cadenas, primitivas y *parcelables* , que se pueden ver en `onReceive` .

Usando LocalBroadcastManager

LocalBroadcastManager se utiliza para enviar Intentos de difusión dentro de una aplicación, sin exponerlos a oyentes no deseados.

Usar *LocalBroadcastManager* es más eficiente y seguro que usar `context.sendBroadcast()` directamente, ya que no necesita preocuparse por las transmisiones falsificadas por otras Aplicaciones, que pueden representar un peligro para la seguridad.

Aquí hay un ejemplo simple de enviar y recibir transmisiones locales:

```
BroadcastReceiver receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
```

```
        if (intent.getAction().equals("Some Action")) {
            //Do something
        }
    }
});

LocalBroadcastManager manager = LocalBroadcastManager.getInstance(mContext);
manager.registerReceiver(receiver, new IntentFilter("Some Action"));

// onReceive() will be called as a result of this call:
manager.sendBroadcast(new Intent("Some Action"));//See also sendBroadcastSync

//Remember to unregister the receiver when you are done with it:
manager.unregisterReceiver(receiver);
```

Receptor Bluetooth Broadcast

agrega permiso en tu archivo manifiesto

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

En tu Fragmento (o Actividad)

- Añade el método del receptor

```
private BroadcastReceiver mBluetoothStatusChangedReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        final Bundle extras = intent.getExtras();
        final int bluetoothState = extras.getInt(Constants.BUNDLE_BLUETOOTH_STATE);
        switch(bluetoothState) {
            case BluetoothAdapter.STATE_OFF:
                // Bluetooth OFF
                break;
            case BluetoothAdapter.STATE_TURNING_OFF:
                // Turning OFF
                break;
            case BluetoothAdapter.STATE_ON:
                // Bluetooth ON
                break;
            case BluetoothAdapter.STATE_TURNING_ON:
                // Turning ON
                break;
    }
};
```

Registrar transmisión

- Llame a este método en onResume ()

```
private void registerBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    manager.registerReceiver(mBluetoothStatusChangedReceiver, new
IntentFilter(Constants.BROADCAST_BLUETOOTH_STATE));
```

```
}
```

Anular el registro de transmisión

- Llame a este método en onPause ()

```
private void unregisterBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    // Beacon□□□
    manager.unregisterReceiver(mBluetoothStatusChangedReceiver);
}
```

Habilitar y deshabilitar un receptor de difusión programáticamente

Para habilitar o deshabilitar un `BroadcastReceiver` , necesitamos obtener una referencia al `PackageManager` y necesitamos un objeto `ComponentName` que contenga la clase del receptor que deseamos habilitar / deshabilitar:

```
ComponentName componentName = new ComponentName(context, MyBroadcastReceiver.class);
PackageManager packageManager = context.getPackageManager();
```

Ahora podemos llamar al siguiente método para habilitar `BroadcastReceiver` :

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP);
```

O podemos usar `COMPONENT_ENABLED_STATE_DISABLED` para desactivar el receptor:

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
    PackageManager.DONT_KILL_APP);
```

BroadcastReceiver para manejar eventos BOOT_COMPLETED

El siguiente ejemplo muestra cómo crear un `BroadcastReceiver` que puede recibir eventos `BOOT_COMPLETED` . De esta manera, puede iniciar un `Service` o iniciar una `Activity` tan pronto como se encendió el dispositivo.

Además, puede usar eventos `BOOT_COMPLETED` para restaurar sus alarmas, ya que se destruyen cuando se apaga el dispositivo.

NOTA: El usuario debe haber iniciado la aplicación al menos una vez antes de poder recibir la acción `BOOT_COMPLETED` .

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.test.example" >
    ...
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    ...

    <application>
        ...

        <receiver android:name="com.test.example.MyCustomBroadcastReceiver">
        <intent-filter>
                <!-- REGISTER TO RECEIVE BOOT_COMPLETED EVENTS -->
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

MyCustomBroadcastReceiver.java

```
public class MyCustomBroadcastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if(action != null) {
            if (action.equals(Intent.ACTION_BOOT_COMPLETED) ) {
                // TO-DO: Code to handle BOOT COMPLETED EVENT
                // TO-DO: I can start an service.. display a notification... start an activity
            }
        }
    }
}
```

Ejemplo de un LocalBroadcastManager

Un BroadcastReceiver es básicamente un mecanismo para transmitir intenciones a través del sistema operativo para realizar acciones específicas. Una definición clásica siendo

> "Un receptor de difusión es un componente de Android que le permite registrarse para eventos del sistema o de la aplicación".

LocalBroadcastManager es una forma de enviar o recibir transmisiones dentro de un proceso de solicitud. Este mecanismo tiene muchas ventajas.

1. ya que los datos permanecen dentro del proceso de la aplicación, los datos no se pueden filtrar.
2. Las transmisiones locales se resuelven más rápido, ya que la resolución de una transmisión normal ocurre en el tiempo de ejecución en todo el sistema operativo.

Un ejemplo simple de un LocalBroastManager es:

SenderActivity

```
  Intent intent = new Intent("anEvent");
  intent.putExtra("key", "This is an event");
  LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
```

ReceiverActivity

1. Registrar un receptor

```
LocalBroadcastManager.getInstance(this).registerReceiver(aLBReceiver,
              new IntentFilter("anEvent"));
```

2. Un objeto concreto para realizar una acción cuando se llama al receptor.

```
private BroadcastReceiver aLBReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        // perform action here.
    }
};
```

3. anular el registro cuando la vista ya no es visible.

```
@Override
protected void onPause() {
    // Unregister since the activity is about to be closed.
    LocalBroadcastManager.getInstance(this).unregisterReceiver(aLBReceiver);
    super.onDestroy();
}
```

Comunicar dos actividades a través del receptor Broadcast personalizado.

Puede comunicar dos actividades para que la Actividad A pueda ser notificada de un evento que ocurra en la Actividad B.

Actividad A

```
final String eventName = "your.package.goes.here.EVENT";

@Override
protected void onCreate(Bundle savedInstanceState) {
    registerEventReceiver();
    super.onCreate(savedInstanceState);
}

@Override
protected void onDestroy() {
    unregisterEventReceiver(eventReceiver);
    super.onDestroy();
}

private void registerEventReceiver() {
    IntentFilter eventFilter = new IntentFilter();
    eventFilter.addAction(eventName);
    registerReceiver(eventReceiver, eventFilter);
```

```
}

private BroadcastReceiver eventReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
         //This code will be executed when the broadcast in activity B is launched
    }
};
```

Actividad B

```
final String eventName = "your.package.goes.here.EVENT";

private void launchEvent() {
    Intent eventIntent = new Intent(eventName);
    this.sendBroadcast(eventIntent);
}
```

Por supuesto, puede agregar más información a la transmisión, agregando extras a la Intención que se pasa entre las actividades. No añadido para mantener el ejemplo lo más simple posible.

Transmisión pegajosa

Si estamos usando el método sendStickyBroadcast (intento), el intento correspondiente es fijo, lo que significa que el intento que está enviando permanece después de que se completa la transmisión. Un StickyBroadcast como su nombre indica es un mecanismo para leer los datos de una transmisión, una vez que se completa la transmisión. Esto se puede usar en un escenario en el que es posible que desee verificar, por ejemplo, en `Activity's onCreate()` una `Activity's onCreate()` el valor de una clave en la intención antes de que se lanzara esa Actividad.

```
Intent intent = new Intent("com.org.action");
intent.putExtra("anIntegerKey", 0);
sendStickyBroadcast(intent);
```

Usando transmisiones ordenadas

Las transmisiones ordenadas se utilizan cuando necesita especificar una prioridad para los oyentes de transmisión.

En este ejemplo, `firstReceiver` recibirá una transmisión siempre antes que un `secondReceiver` :

```
final int highPriority = 2;
final int lowPriority = 1;
final String action = "action";

// intent filter for first receiver with high priority
final IntentFilter firstFilter = new IntentFilter(action);
first Filter.setPriority(highPriority);
final BroadcastReceiver firstReceiver = new MyReceiver();

// intent filter for second receiver with low priority
final IntentFilter secondFilter = new IntentFilter(action);
secondFilter.setPriority(lowPriority);
```

```
final BroadcastReceiver secondReceiver = new MyReceiver();

// register our receivers
context.registerReceiver(firstReceiver, firstFilter);
context.registerReceiver(secondReceiver, secondFilter);

// send ordered broadcast
context.sendOrderedBroadcast(new Intent(action), null);
```

Además, el receptor de difusión puede abortar la emisión ordenada:

```
@Override
public void onReceive(final Context context, final Intent intent) {
    abortBroadcast();
}
```

en este caso, todos los receptores con prioridad más baja no recibirán un mensaje de difusión.

Android detuvo el estado

A partir de Android 3.1, todas las aplicaciones, después de la instalación, se colocan en un estado detenido. Mientras se encuentre en estado detenido, la aplicación no se ejecutará por ningún motivo, excepto mediante el lanzamiento manual de una actividad o una intención **explícita** que aborde una actividad, servicio o difusión.

Al escribir la aplicación del sistema que instala los APK directamente, tenga en cuenta que la aplicación recién instalada no recibirá ninguna transmisión hasta que se mueva a un estado no detenido.

Una forma fácil de activar una aplicación es enviar una transmisión explícita a esta aplicación. Como la mayoría de las aplicaciones implementan `INSTALL_REFERRER` , podemos usarlo como un punto de `INSTALL_REFERRER` .

Escanee el manifiesto de la aplicación instalada y envíe una transmisión explícita a cada receptor:

```
Intent intent = new Intent();
intent.addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES);
intent.setComponent(new ComponentName(packageName, fullClassName));
sendBroadcast(intent);
```

Lea Receptor de radiodifusión en línea: https://riptutorial.com/es/android/topic/1460/receptor-de-radiodifusion

Capítulo 204: Recolectores de fecha y hora

Examples

Material DatePicker

agregue las dependencias a continuación al archivo `build.gradle` en la sección de dependencia. (esta es una biblioteca no oficial para el selector de fechas)

```
  compile 'com.wdullaer:materialdatetimepicker:2.3.0'
```

Ahora tenemos que abrir `DatePicker` en el evento de clic de botón.

Así que crea un botón en el archivo xml como abajo.

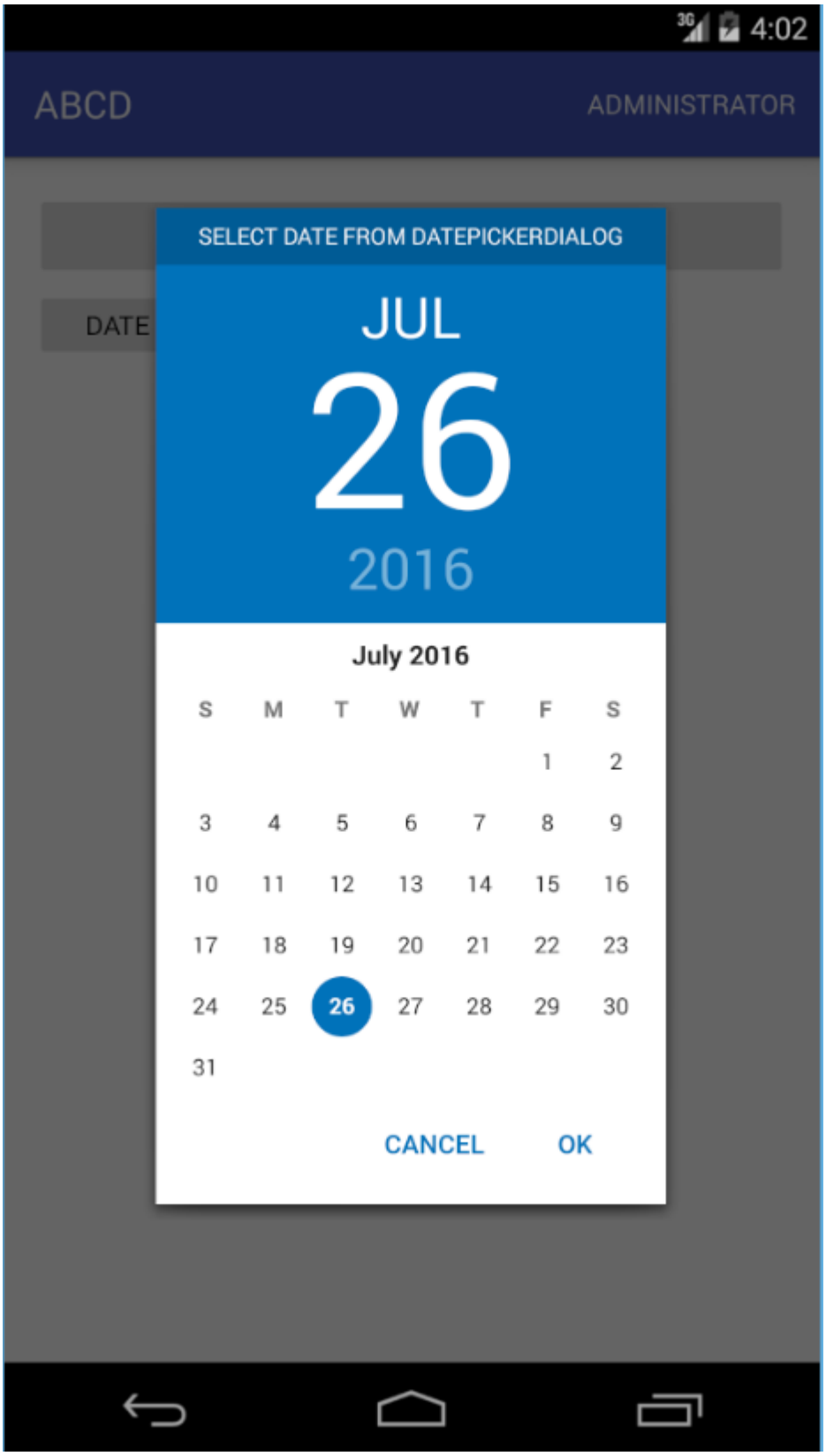
```
     <Button
            android:id="@+id/dialog_bt_date"
            android:layout_below="@+id/resetButton"
            android:layout_width="wrap_content"
            android:layout_height="40dp"
            android:textColor="#FF000000"
            android:gravity="center"
            android:text="DATE"/>
```

y en MainActivity usar de esta manera.

```
public class MainActivity extends AppCompatActivity implements
DatePickerDialog.OnDateSetListener{

    Button button;
    Calendar calendar ;
    DatePickerDialog datePickerDialog ;
    int Year, Month, Day ;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        calendar = Calendar.getInstance();

        Year = calendar.get(Calendar.YEAR) ;
        Month = calendar.get(Calendar.MONTH);
        Day = calendar.get(Calendar.DAY_OF_MONTH);


        Button dialog_bt_date = (Button)findViewById(R.id.dialog_bt_date);
        dialog_bt_date.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                datePickerDialog = DatePickerDialog.newInstance(MainActivity.this, Year,
```

```
Month, Day);

                datePickerDialog.setThemeDark(false);

                datePickerDialog.showYearPickerFirst(false);

                datePickerDialog.setAccentColor(Color.parseColor("#0072BA"));

                datePickerDialog.setTitle("Select Date From DatePickerDialog");

                datePickerDialog.show(getFragmentManager(), "DatePickerDialog");

            }
        });
    }

    @Override
    public void onDateSet(DatePickerDialog view, int Year, int Month, int Day) {

        String date = "Selected Date : " + Day + "-" + Month + "-" + Year;

        Toast.makeText(MainActivity.this, date, Toast.LENGTH_LONG).show();
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        getMenuInflater().inflate(R.menu.abc_main_menu, menu);
        return true;
    }

}
```

Salida:



Cuadro de diálogo Selector de fecha

Es un cuadro de diálogo que solicita al usuario que seleccione la fecha con `DatePicker` . El diálogo requiere contexto, año inicial, mes y día para mostrar el diálogo con la fecha de inicio. Cuando el usuario selecciona la fecha en que `DatePickerDialog.OnDateSetListener` llamadas a través de `DatePickerDialog.OnDateSetListener` .

```
public void showDatePicker(Context context,int initialYear, int initialMonth, int initialDay)
{
       DatePickerDialog datePickerDialog = new DatePickerDialog(context,
                new DatePickerDialog.OnDateSetListener() {
                    @Override
                    public void onDateSet(DatePicker datepicker,int year ,int month, int day)
{
                    //this condition is necessary to work properly on all android versions
                    if(view.isShown()){
                            //You now have the selected year, month and day
                    }

}
                }, initialYear, initialMonth , initialDay);

        //Call show() to simply show the dialog
        datePickerDialog.show();

    }
```

Tenga en cuenta que el mes es un inicio que comienza desde 0 para enero hasta 11 para diciembre

Lea Recolectores de fecha y hora en línea:
https://riptutorial.com/es/android/topic/2836/recolectores-de-fecha-y-hora

Capítulo 205: Reconocimiento de actividad

Introducción

El reconocimiento de actividad es la detección de la actividad física de un usuario para realizar ciertas acciones en el dispositivo, como ganar puntos cuando se detecta una unidad, desactivar el wifi cuando el teléfono está quieto o poner el volumen del timbre al máximo cuando el usuario está para caminar.

Examples

Actividad de Google PlayReconocimientoAPI

Este es solo un ejemplo simple de cómo utilizar ActivityRecognitionApi de GooglePlay Service. Aunque esta es una gran biblioteca, no funciona en dispositivos que no tienen los servicios de Google Play instalados.

Docs para la API ActivityRecognition

Manifiesto

```
<!-- This is needed to use Activity Recognition! -->
<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <receiver android:name=".ActivityReceiver" />
</application>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements
GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener {

    private GoogleApiClient apiClient;
    private LocalBroadcastManager localBroadcastManager;
    private BroadcastReceiver localActivityReceiver;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        apiClient = new GoogleApiClient.Builder(this)
                .addApi(ActivityRecognition.API)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .build();

        //This just gets the activity intent from the ActivityReceiver class
        localBroadcastManager = LocalBroadcastManager.getInstance(this);
        localActivityReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                ActivityRecognitionResult recognitionResult =
ActivityRecognitionResult.extractResult(intent);
                TextView textView = (TextView) findViewById(R.id.activityText);

                //This is just to get the activity name. Use at your own risk.

textView.setText(DetectedActivity.zzkf(recognitionResult.getMostProbableActivity().getType()));

            }
        };
    }

    @Override
    protected void onResume() {
        super.onResume();

        //Register local broadcast receiver
        localBroadcastManager.registerReceiver(localActivityReceiver, new
IntentFilter("activity"));

        //Connect google api client
        apiClient.connect();
    }

    @Override
    protected void onPause() {
        super.onPause();

        //Unregister for activity recognition
        ActivityRecognition.ActivityRecognitionApi.removeActivityUpdates(apiClient,
PendingIntent.getBroadcast(this, 0, new Intent(this, ActivityReceiver.class),
PendingIntent.FLAG_UPDATE_CURRENT));

        //Disconnects api client
        apiClient.disconnect();

        //Unregister local receiver
        localBroadcastManager.unregisterReceiver(localActivityReceiver);
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        //Only register for activity recognition if google api client has connected
        ActivityRecognition.ActivityRecognitionApi.requestActivityUpdates(apiClient, 0,
PendingIntent.getBroadcast(this, 0, new Intent(this, ActivityReceiver.class),
```

```
PendingIntent.FLAG_UPDATE_CURRENT));
    }

    @Override
    public void onConnectionSuspended(int i) {
    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

    }
}
```

ActividadReceptor

```
public class ActivityReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

LocalBroadcastManager.getInstance(context).sendBroadcast(intent.setAction("activity"));
    }
}
```

Reconocimiento de la actividad PathSense

El reconocimiento de actividad PathSense es otra buena biblioteca para dispositivos que no cuentan con Google Play Services, ya que han creado su propio modelo de reconocimiento de actividad, pero requieren que los desarrolladores se registren en http://developer.pathsense.com para obtener una clave de API y una ID de cliente .

Manifiesto

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <receiver android:name=".ActivityReceiver" />

    <!-- You need to acquire these from their website (http://developer.pathsense.com) -->
    <meta-data
        android:name="com.pathsense.android.sdk.CLIENT_ID"
        android:value="YOUR_CLIENT_ID" />
    <meta-data
```

```
        android:name="com.pathsense.android.sdk.API_KEY"
        android:value="YOUR_API_KEY" />
</application>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    private PathsenseLocationProviderApi pathsenseLocationProviderApi;
    private LocalBroadcastManager localBroadcastManager;
    private BroadcastReceiver localActivityReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        pathsenseLocationProviderApi = PathsenseLocationProviderApi.getInstance(this);

        //This just gets the activity intent from the ActivityReceiver class
        localBroadcastManager = LocalBroadcastManager.getInstance(this);
        localActivityReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                //The detectedActivities object is passed as a serializable
                PathsenseDetectedActivities detectedActivities = (PathsenseDetectedActivities)
intent.getSerializableExtra("ps");
                TextView textView = (TextView) findViewById(R.id.activityText);

textView.setText(detectedActivities.getMostProbableActivity().getDetectedActivity().name());
            }
        };
    }

    @Override
    protected void onResume() {
        super.onResume();

        //Register local broadcast receiver
        localBroadcastManager.registerReceiver(localActivityReceiver, new
IntentFilter("activity"));

        //This gives an update everytime it receives one, even if it was the same as the last
update
        pathsenseLocationProviderApi.requestActivityUpdates(ActivityReceiver.class);

//        This gives updates only when it changes (ON_FOOT -> IN_VEHICLE for example)
//        pathsenseLocationProviderApi.requestActivityChanges(ActivityReceiver.class);
    }

    @Override
    protected void onPause() {
        super.onPause();

        pathsenseLocationProviderApi.removeActivityUpdates();

//        pathsenseLocationProviderApi.removeActivityChanges();

        //Unregister local receiver
        localBroadcastManager.unregisterReceiver(localActivityReceiver);
    }
```

```
}
```

ActivityReceiver.java

```
// You don't have to use their broadcastreceiver, but it's best to do so, and just pass the
result
// as needed to another class.
public class ActivityReceiver extends PathsenseActivityRecognitionReceiver {

    @Override
    protected void onDetectedActivities(Context context, PathsenseDetectedActivities
pathsenseDetectedActivities) {
        Intent intent = new Intent("activity").putExtra("ps", pathsenseDetectedActivities);
        LocalBroadcastManager.getInstance(context).sendBroadcast(intent);
    }
}
```

Lea Reconocimiento de actividad en línea:
https://riptutorial.com/es/android/topic/9831/reconocimiento-de-actividad

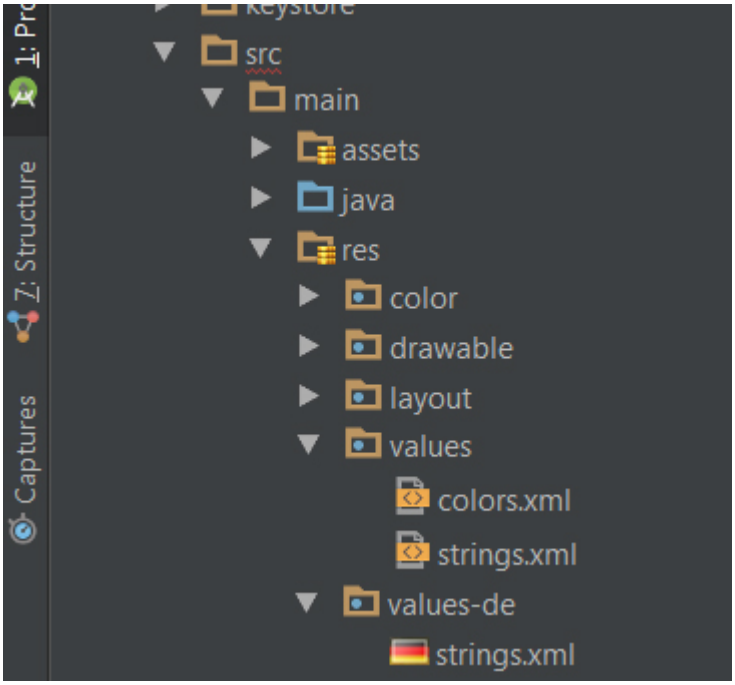
Capítulo 206: Recursos

Examples

Traducir una cadena

Las cadenas se pueden internacionalizar definiendo un archivo strings.xml diferente para cada idioma que admita.

Usted agrega un nuevo idioma al crear un nuevo directorio de valores con el código de idioma ISO como un sufijo. Por ejemplo, cuando se agrega un conjunto alemán, su estructura puede tener el siguiente aspecto:



Cuando el sistema busca la cadena solicitada, primero verifica el xml específico del idioma, si no se encuentra, se devuelve el valor del archivo strings.xml predeterminado. La clave sigue siendo la misma para cada idioma y solo cambia el valor.

Contenidos de ejemplo:

/res/valores/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello World!</string>
</resources>
```

/res/valores-fr/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resources>
    <string name="hello_world">Bonjour tout le monde !!!</string>
</resources>
```

Definir cuerdas

Normalmente, las cadenas se almacenan en el archivo de recursos `strings.xml` . Se definen utilizando un elemento XML `<string>` .

El propósito de strings.xml es permitir la internacionalización. Puede definir un strings.xml para cada código iso de idioma. Por lo tanto, cuando el sistema busca la cadena 'app_name', primero verifica el archivo xml correspondiente al idioma actual y, si no se encuentra, busca la entrada en el archivo strings.xml predeterminado. Esto significa que puede elegir solo localizar algunas de sus cadenas mientras que otras no.

/res/valores/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Hello World App</string>
    <string name="hello_world">Hello World!</string>
</resources>
```

Una vez que se define una cadena en un archivo de recursos XML, puede ser utilizada por otras partes de la aplicación.

Los archivos de proyecto XML de una aplicación pueden usar un elemento `<string>` refiriéndose a `@string/string_name` . Por ejemplo, el archivo de manifiesto de una aplicación (/manifests/AndroidManifest.xml) incluye la siguiente línea de forma predeterminada en Android Studio:

```
android:label="@string/app_name"
```

Esto le dice a Android que busque un recurso `<string>` llamado "app_name" para usarlo como el nombre de la aplicación cuando se instala o se muestra en un iniciador.

Otra vez que usaría un recurso `<string>` de un archivo XML en Android estaría en un archivo de diseño. Por ejemplo, lo siguiente representa un TextView que muestra la cadena `hello_world` que definimos anteriormente:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world"/>
```

También puede acceder a `<string>` recursos `<string>` desde la parte java de su aplicación. Para recuperar nuestra misma cadena `hello_world` desde arriba dentro de una clase de actividad, use:

```
String helloWorld = getString(R.string.hello_world);
```

Definir matriz de cadena

Para definir una matriz de cadenas, escriba en un archivo de recursos.

res / values / filename.xml

```
<string-array name="string_array_name">
    <item>text_string</item>
    <item>@string/string_id</item>
</string-array>
```

por ejemplo

res / values / arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="string_array_example">
        <item>@string/app_name</item>
        <item>@string/hello_world</item>
    </string-array>
</resources>
```

y usarlo desde java como

```
String[] strings = getResources().getStringArray(R.array.string_array_example;
Log.i("TAG",Arrays.toString(strings)));
```

Salida

```
I/TAG: [HelloWorld, Hello World!]
```

Definir dimensiones

Las dimensiones normalmente se almacenan en un archivo de recursos con el nombre `dimens.xml` . Se definen utilizando un elemento `<dimen>` .

res / values / dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="small_padding">5dp</dimen>
    <dimen name="medium_padding">10dp</dimen>
    <dimen name="large_padding">20dp</dimen>

    <dimen name="small_font">14sp</dimen>
    <dimen name="medium_font">16sp</dimen>
    <dimen name="large_font">20sp</dimen>
</resources>
```

Puedes usar diferentes unidades:

- **sp:** Píxeles independientes de la escala. Para fuentes.
- **dp:** Pixeles independientes de densidad. Para todo lo demás.
- **pt:** puntos
- **px:** píxeles
- **mm:** milimetros
- **im:** pulgadas

Ahora se puede hacer referencia a las dimensiones en XML con la sintaxis `@dimen/name_of_the_dimension` .

Por ejemplo:

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="@dimen/large_padding">
</RelativeLayout>
```

Definir enteros

Los enteros normalmente se almacenan en un archivo de recursos llamado `integers.xml` , pero el nombre del archivo puede elegirse arbitrariamente. Cada entero se define utilizando un elemento `<integer>` , como se muestra en el siguiente archivo:

res / values / integers.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer name="max">100</integer>
</resources>
```

Ahora se puede hacer referencia a `@integer/name_of_the_integer` en XML con la sintaxis `@integer/name_of_the_integer` , como se muestra en el siguiente ejemplo:

```
<ProgressBar
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:max="@integer/max"/>
```

Definir matriz de enteros

Para definir una matriz de enteros, escriba en un archivo de recursos.

res / values / filename.xml

```
<integer-array name="integer_array_name">
    <item>integer_value</item>
    <item>@integer/integer_id</item>
```

```
</integer-array>
```

por ejemplo

res / values / arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer-array name="fibo">
        <item>@integer/zero</item>
        <item>@integer/one</item>
        <item>@integer/one</item>
        <item>@integer/two</item>
        <item>@integer/three</item>
        <item>@integer/five</item>
    </integer-array>
</resources>
```

y usarlo desde java como

```
int[] values = getResources().getIntArray(R.array.fibo);
Log.i("TAG",Arrays.toString(values)));
```

Salida

```
I/TAG: [0, 1, 1, 2, 3, 5]
```

Definir colores

Los colores generalmente se almacenan en un archivo de recursos llamado `colors.xml` en la carpeta `/res/values/` .

Están definidos por elementos `<color>` :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>

    <color name="blackOverlay">#66000000</color>
</resources>
```

Los colores se representan mediante valores de color hexadecimales para cada canal de color `(0 - FF)` en uno de los formatos:

- `#RGB`
- `#ARGB`
- `#RRGGBB`
- `#AARRGGBB`

Leyenda

- A - canal alfa - el valor 0 es completamente transparente, el valor FF es opaco
- R - canal rojo
- G - canal verde
- B - canal azul

Los colores definidos se pueden usar en XML con la siguiente sintaxis `@color/name_of_the_color`

Por ejemplo:

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/blackOverlay">
```

Usando colores en codigo

Estos ejemplos asumen que `this` es una referencia de actividad. También se puede utilizar una referencia de contexto en su lugar.

1.6

```
int color = ContextCompat.getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

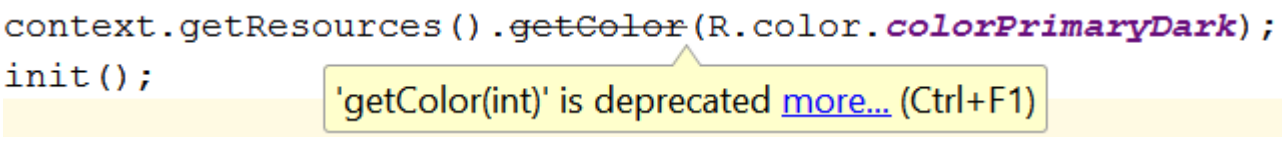
6.0

```
int color = this.getResources().getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

En la declaración `colorPrimary` , `colorPrimary` , `colorPrimaryDark` y `colorAccent` se utilizan para definir los colores de diseño del material que se utilizarán para definir el tema personalizado de Android en `styles.xml` . Se agregan automáticamente cuando se crea un nuevo proyecto con Android Studio.

Obteniendo recursos sin advertencias "obsoletas"

Usando la API de Android 23 o superior, muy a menudo tal situación se puede ver:



Esta situación es causada por el cambio estructural de la API de Android con respecto a obtener los recursos.
Ahora la función:

```
public int getColor(@ColorRes int id, @Nullable Theme theme) throws NotFoundException
```

debería ser usado. Pero la biblioteca `android.support.v4` tiene otra solución.

Agregue la siguiente dependencia al archivo `build.gradle` :

```
com.android.support:support-v4:24.0.0
```

Entonces todos los métodos de la biblioteca de soporte están disponibles:

```
ContextCompat.getColor(context, R.color.colorPrimaryDark);
ContextCompat.getDrawable(context, R.drawable.btn_check);
ContextCompat.getColorStateList(context, R.color.colorPrimary);
DrawableCompat.setTint(drawable);
ContextCompat.getColor(context,R.color.colorPrimaryDark));
```

Además, se pueden utilizar más métodos de la biblioteca de soporte:

```
ViewCompat.setElevation(textView, 1F);
ViewCompat.animate(textView);
TextViewCompat.setTextAppearance(textView, R.style.AppThemeTextStyle);
...
```

Defina un recurso de menú y utilícelo dentro de Actividad / Fragmento

Definir un menú en **res / menu.**

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/first_item_id"
        android:orderInCategory="100"
        android:title="@string/first_item_string"
        android:icon="@drawable/first_item_icon"
        app:showAsAction="ifRoom"/>

    <item
        android:id="@+id/second_item_id"
        android:orderInCategory="110"
        android:title="@string/second_item_string"
        android:icon="@drawable/second_item_icon"
        app:showAsAction="ifRoom"/>

</menu>
```

Para más opciones de configuración consulte: Recurso de menú.

`Activity` interior:

```
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    ///Override defining menu resource
    inflater.inflate(R.menu.menu_resource_id, menu);
    super.onCreateOptionsMenu(menu, inflater);
}
```

```
@Override
public void onPrepareOptionsMenu(Menu menu) {
    //Override for preparing items (setting visibility, change text, change icon...)
    super.onPrepareOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    //Override it for handling items
    int menuItemId = item.getItemId();
    switch (menuItemId) {
        case: R.id.first_item_id
             return true; //return true, if is handled
    }
    return super.onOptionsItemSelected(item);
}
```

Para invocar los métodos anteriores durante la visualización de la vista, llame a `getActivity().invalidateOptionsMenu();`

Dentro del **`Fragment`** se necesita una llamada adicional:

```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    setHasOptionsMenu(true);
    super.onCreateView(inflater, container, savedInstanceState);
}
```

Formato de cadena en cadenas.xml

La definición de cadenas en el archivo strings.xml también permite el formato de cadenas. La única advertencia es que la Cadena deberá tratarse en un código como el que se muestra a continuación, en lugar de simplemente adjuntarlo a un diseño.

```
<string name="welcome_trainer">Hello Pokémon Trainer, %1$s! You have caught %2$d
Pokémon.</string>
```

```
String welcomePokemonTrainerText = getString(R.string.welcome_trainer, tranerName,
pokemonCount);
```

En el ejemplo anterior,
% 1 $ s
'%' se separa de los caracteres normales,
'1' denota el primer parámetro,
'$' se usa como separador entre el número de parámetro y el tipo,
's' indica el tipo de cadena ('d' se usa para un entero)

Tenga en cuenta que `getString()` es un método de `Context` o `Resources` , es decir, puede usarlo directamente dentro de una instancia de `Activity` , o bien puede usar `getActivity().getString()` o `getContext().getString()` respectivamente.

Definir una lista de estados de color.

Las listas de estados de color se pueden usar como colores, pero cambiarán dependiendo del estado de la vista para la que se usan.

Para definir uno, cree un archivo de recursos en `res/color/foo.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="#888888" android:state_enabled="false"/>
    <item android:color="@color/lightGray" android:state_selected="false"/>
    <item android:color="@android:color/white" />
</selector>
```

Los elementos se evalúan en el orden en que se definen y se utiliza el primer elemento cuyos estados especificados coinciden con el estado actual de la vista. Por lo tanto, es una buena práctica especificar un catch-all al final, sin ningún selector de estado especificado.

Cada elemento puede usar un literal de color o hacer referencia a un color definido en otro lugar.

Definir plurales de cadena

Para diferenciar entre cadenas en plural y en singular, puede definir un plural en su archivo *strings.xml* y enumerar las diferentes cantidades, como se muestra en el siguiente ejemplo:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals name="hello_people">
        <item quantity="one">Hello to %d person</item>
        <item quantity="other">Hello to %d people</item>
    </plurals>
</resources>
```

Se puede acceder a esta definición desde el código Java utilizando el método `getQuantityString()` de la clase de `Resources` , como se muestra en el siguiente ejemplo:

```
getResources().getQuantityString(R.plurals.hello_people, 3, 3);
```

Aquí, el primer parámetro `R.plurals.hello_people` es el nombre del recurso. El segundo parámetro (`3` en este ejemplo) se usa para elegir la cadena de `quantity` correcta. El tercer parámetro (también `3` en este ejemplo) es el argumento de formato que se usará para sustituir el especificador de formato `%d` .

Los valores de cantidad posibles (listados en orden alfabético) son:

```
few
many
one
other
two
zero
```

Es importante tener en cuenta que no todas las configuraciones regionales admiten cada denominación de `quantity` . Por ejemplo, el idioma chino no tiene un concepto de `one` elemento. El inglés no tiene un elemento `zero` , ya que es gramaticalmente el mismo que `other` . Las instancias de `quantity` no admitidas serán marcadas por el IDE como advertencias de pelusa, pero no causarán errores de complicación si se usan.

Importar matriz de objetos definidos en recursos.

Hay casos en los que es necesario crear y definir objetos personalizados en los recursos de la aplicación. Dichos objetos pueden estar compuestos de tipos simples de `Java` , por ejemplo, `Integer` , `Float` , `String` .

Este es el ejemplo de cómo importar un objeto definido en los recursos de la aplicación. La `Category` objeto consta de 3 propiedades de categoría:

- CARNÉ DE IDENTIDAD
- Color
- Nombre

Este `POJO` tiene su equivalente en el archivo `categories.xml` , donde cada matriz tiene las mismas propiedades definidas para cada categoría.

1. **Crea un modelo para tu objeto:**

```
public class Category {
    private Type id;
    private @ColorRes int color;
    private @StringRes String name;

    public Category getId() {
        return id;
    }

    public void setId(Category id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getColor() {
        return color;
    }

    public void setColor(int color) {
        this.color = color;
    }

    public enum Type{
        REGISTRATION,
```

```
        TO_ACCEPT,
        TO_COMPLETE,
        TO_VERIFY,
        CLOSED
    }
}
```

2. **Crea el archivo en la carpeta `res/values` :**

```
categories.xml
```

3. **Componga cada modelo que consta de recursos:**

```
<array name="no_action">
    <item>0</item>
    <item>@android:color/transparent</item>
    <item>@string/statusRegistration</item>
</array>
<array name="to_accept">
    <item>1</item>
    <item>@color/light_gray</item>
    <item>@string/acceptance</item>
</array>
<array name="opened">
    <item>2</item>
    <item>@color/material_green_500</item>
    <item>@string/open</item>
</array>
<array name="to_verify">
    <item>3</item>
    <item>@color/material_gray_800</item>
    <item>@string/verification</item>
</array>
<array name="to_close">
    <item>4</item>
    <item>@android:color/black</item>
    <item>@string/closed</item>
</array>
```

4. **Defina una matriz en el archivo de recursos:**

```
<array name="categories">
    <item>@array/no_action</item>
    <item>@array/to_accept</item>
    <item>@array/opened</item>
    <item>@array/to_verify</item>
    <item>@array/to_close</item>
</array>
```

5. **Crea una función para importarlos:**

```
@NonNull
public List<Category> getCategories(@NonNull Context context) {
final int DEFAULT_VALUE = 0;
final int ID_INDEX = 0;
final int COLOR_INDEX = 1;
final int LABEL_INDEX = 2;
```

```
if (context == null) {
    return Collections.emptyList();
}
// Get the array of objects from the `tasks_categories` array
TypedArray statuses = context.getResources().obtainTypedArray(R.array.categories);
if (statuses == null) {
    return Collections.emptyList();
}
List<Category> categoryList = new ArrayList<>();
for (int i = 0; i < statuses.length(); i++) {
    int statusId = statuses.getResourceId(i, DEFAULT_VALUE);
    // Get the properties of one object
    TypedArray rawStatus = context.getResources().obtainTypedArray(statusId);

    Category category = new Category();

    int id = rawStatus.getInteger(ID_INDEX, DEFAULT_VALUE);
    Category.Type categoryId;
    //The ID's should maintain the order with `Category.Type`
    switch (id) {
        case 0:
            categoryId = Category.Type.REGISTRATION;
            break;
        case 1:
            categoryId = Category.Type.TO_ACCEPT;
            break;
        case 2:
            categoryId = Category.Type.TO_COMPLETE;
            break;
        case 3:
            categoryId = Category.Type.TO_VERIFY;
            break;
        case 4:
            categoryId  = Category.Type.CLOSED;
            break;
        default:
            categoryId = Category.Type.REGISTRATION;
            break;
    }
    category.setId(categoryId);

    category.setColor(rawStatus.getResourceId(COLOR_INDEX, DEFAULT_VALUE));

    int labelId = rawStatus.getResourceId(LABEL_INDEX, DEFAULT_VALUE);
    category.setName(getString(context.getResources(), labelId));

    categoryList.add(taskCategory);
}
return taskCategoryList;
}
```

9 parches

9 Los parches son imágenes **estirables** en las que las áreas que se pueden estirar están definidas por marcadores negros en un borde transparente.

Hay un gran tutorial aquí .
A pesar de ser tan viejo, sigue siendo tan valioso y nos ayudó a muchos a comprender el

engranaje de 9 parches.

Desafortunadamente, recientemente esa página ha sido colocada por un tiempo (actualmente está arriba nuevamente).

Por lo tanto, la necesidad de tener una copia física de esa página para desarrolladores de Android en nuestros servidores confiables.

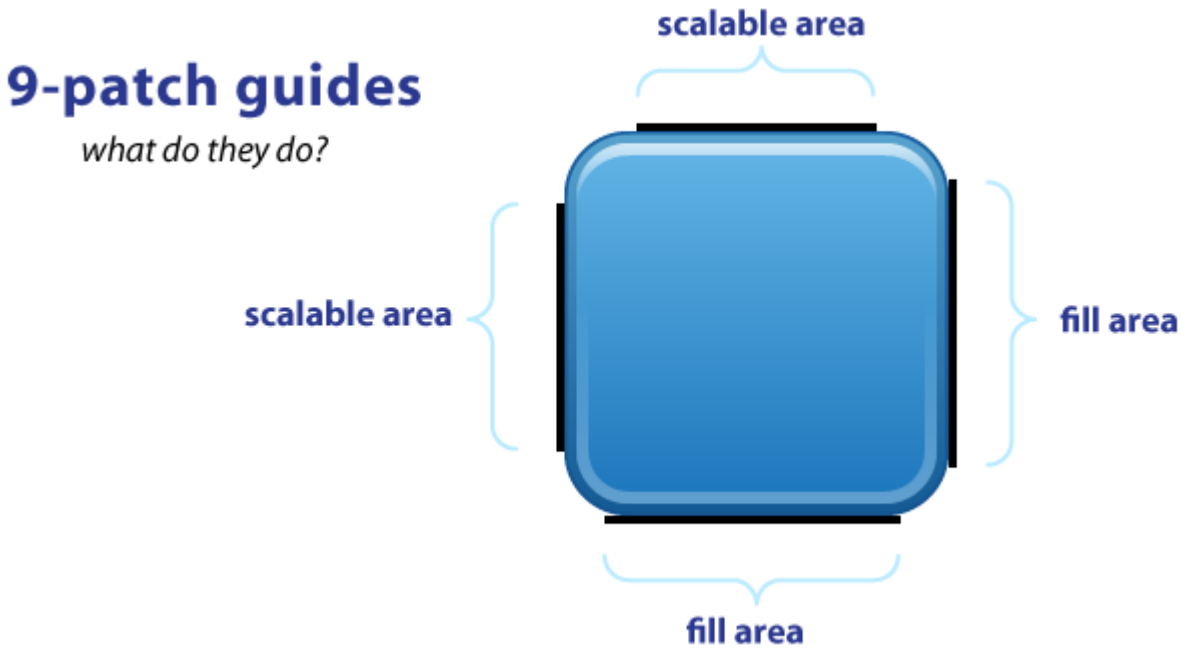
Aquí está.

GUÍA SENCILLA DE 9-PATCH PARA LA IU DE ANDROID 18 de mayo de 2011

Mientras trabajaba en mi primera aplicación de Android, encontré que 9 parches (también conocido como 9.png) son confusos y están mal documentados. Después de un rato, finalmente entendí cómo funciona y decidí juntar algo para ayudar a otros a resolverlo.

Básicamente, el parche 9 utiliza la transparencia png para hacer una forma avanzada de 9 cortes o escala9. Las guías son líneas negras rectas de 1 píxel dibujadas en el borde de la imagen que definen la escala y el relleno de la imagen. Al nombrar el nombre de su archivo de imagen.9.png, Android reconocerá el formato 9.png y utilizará las guías negras para escalar y completar sus mapas de bits.

Aquí hay un mapa guía básico:



Como puedes ver, tienes guías a cada lado de tu imagen. Las guías SUPERIOR e IZQUIERDA son para escalar su imagen (es decir, 9 cortes), mientras que las guías DERECHA y ABAJO definen el área de relleno.

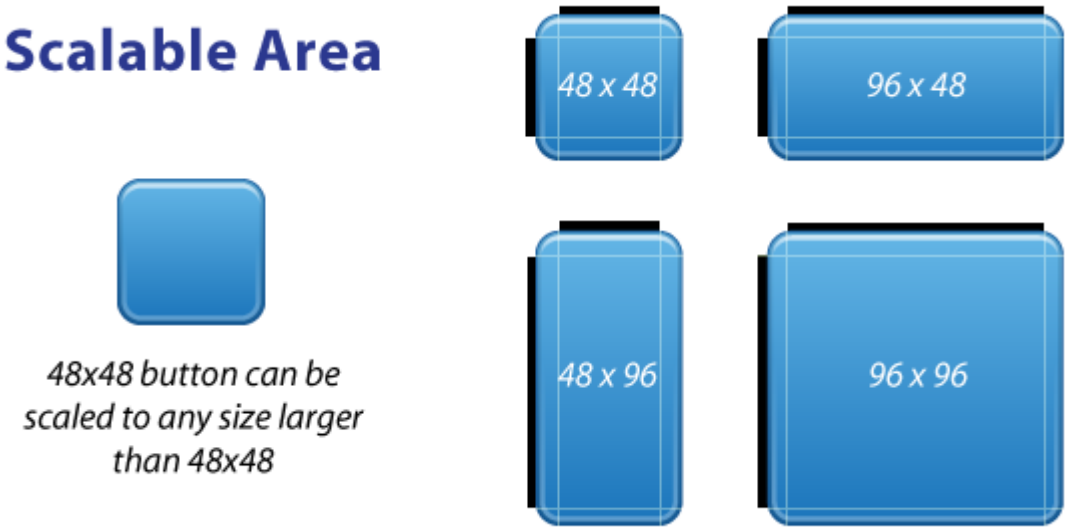
Las líneas de guía negras se cortan / eliminan de su imagen, no se mostrarán en la aplicación.

Las guías solo deben tener un píxel de ancho, por lo que si desea un botón 48 × 48, su png será en realidad 50 × 50. Cualquier cosa más gruesa que un píxel seguirá siendo parte de su imagen. (Mis ejemplos tienen guías de 4 píxeles de ancho para una mejor visibilidad. En realidad, deberían ser solo de 1 píxel).

Sus guías deben ser de color negro sólido (# 000000). Incluso una ligera diferencia en el color (# 000001) o alfa causará que falle y se estire normalmente. Este fallo tampoco será obvio *, ¡falla silenciosamente! Sí. De Verdad. Ahora tu sabes

También debe tener en cuenta que el área restante del contorno de un píxel debe ser completamente transparente. Esto incluye las cuatro esquinas de la imagen, que siempre deben ser claras. Esto puede ser un problema más grande de lo que te das cuenta. Por ejemplo, si escala una imagen en Photoshop, agregará píxeles con antialias que pueden incluir píxeles casi invisibles que también harán que falle *. Si debe escalar en Photoshop, use la configuración de Vecino más cercano en el menú desplegable Remuestrear imagen (en la parte inferior del menú emergente Tamaño de imagen) para mantener los bordes afilados en sus guías.

* (actualizado el 1/2012) Esto es en realidad un "arreglo" en el último kit de desarrollo. Anteriormente, se manifestaría como si todas tus otras imágenes y recursos se rompieran repentinamente, no la imagen de 9 parches realmente rota.

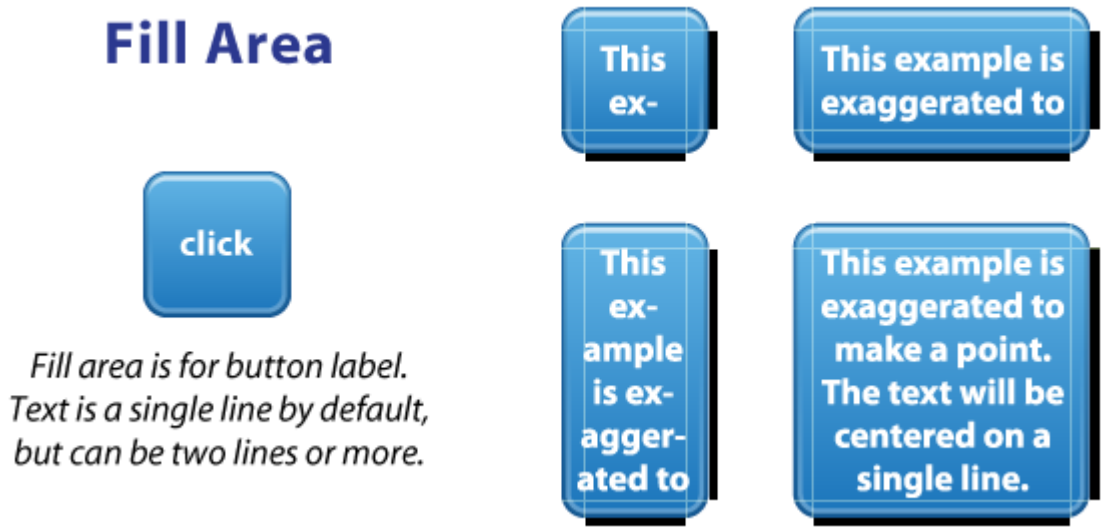


Las guías SUPERIOR e IZQUIERDA se utilizan para definir la parte escalable de su imagen: IZQUIERDA para la altura de escalado, TOP para el ancho de escala. Usando una imagen de botón como ejemplo, esto significa que el botón puede estirarse horizontal y verticalmente dentro de la parte negra y todo lo demás, como las esquinas, seguirá siendo del mismo tamaño. Le permite tener botones que pueden escalarse a cualquier tamaño y mantener una apariencia uniforme.

Es importante tener en cuenta que las imágenes de 9 parches no se reducen, solo aumentan. Así que es mejor empezar lo más pequeño posible.

Además, puede omitir partes en el centro de la línea de escala. Así, por ejemplo, si tiene un botón con un borde brillante y afilado en el medio, puede dejar algunos píxeles en el centro de la guía

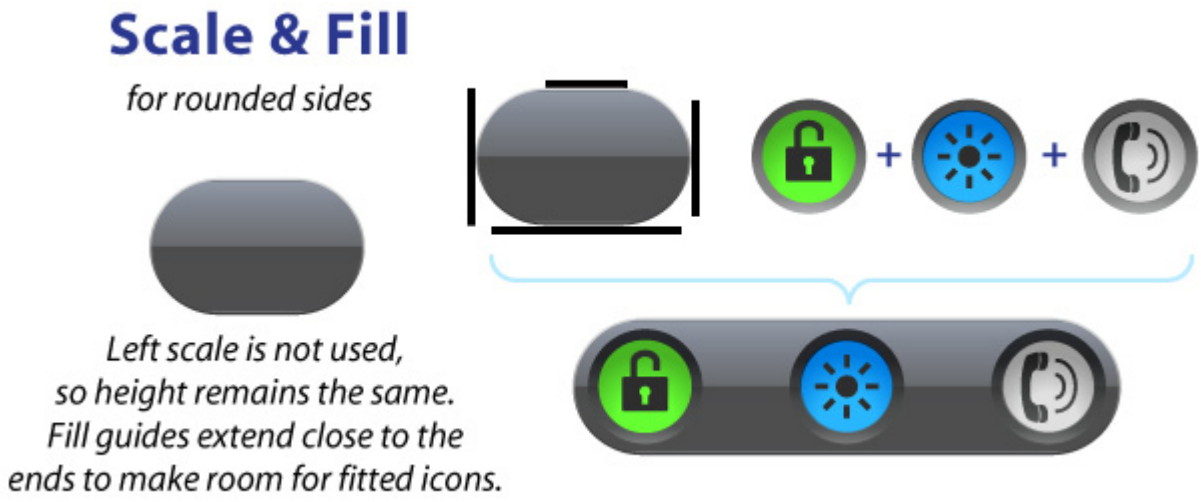
IZQUIERDA. El eje horizontal central de su imagen no se escalará, solo las partes arriba y abajo, por lo que su brillo nítido no se suavizará.



Las guías de área de relleno son opcionales y proporcionan una manera de definir el área para cosas como su etiqueta de texto. El relleno determina la cantidad de espacio que hay dentro de la imagen para colocar texto, un icono u otras cosas. El parche 9 no es solo para botones, también funciona para imágenes de fondo.

El ejemplo anterior de botón y etiqueta es exagerado simplemente para explicar la idea de relleno: la etiqueta no es completamente precisa. Para ser honesto, no he experimentado cómo Android hace etiquetas multilínea ya que una etiqueta de botón suele ser una sola fila de texto.

Finalmente, aquí hay una buena demostración de cómo pueden variar las guías de escala y relleno, como un LinearLayout con una imagen de fondo y lados completamente redondeados:



Con este ejemplo, la guía IZQUIERDA no se usa, pero aún se requiere que tengamos una guía. La imagen de fondo no se escala verticalmente; solo escala horizontalmente (basado en la guía TOP). Al mirar las guías de relleno, las guías DERECHA e INFERIOR se extienden más allá de

donde se encuentran los bordes curvos de la imagen. Esto me permite colocar mis botones redondos cerca de los bordes del fondo para un aspecto ajustado y ajustado.

Eso es todo. 9 parches es súper fácil, una vez que lo consigues. No es una forma perfecta de hacer escala, pero las guías de escala de área de relleno y multilínea ofrecen más flexibilidad que las tradicionales de 9 cortes y escala9. Pruébalo y lo resolverás rápidamente.

Nivel de transparencia de color (alfa)

Valores de opacidad del hex.

```
 ------------------------------
|   Alpha(%)  |   Hex Value    |
 ------------------------------
|    100%     |      FF        |
|     95%     |      F2        |
|     90%     |      E6        |
|     85%     |      D9        |
|     80%     |      CC        |
|     75%     |      BF        |
|     70%     |      B3        |
|     65%     |      A6        |
|     60%     |      99        |
|     55%     |      8C        |
|     50%     |      80        |
|     45%     |      73        |
|     40%     |      66        |
|     35%     |      59        |
|     30%     |      4D        |
|     25%     |      40        |
|     20%     |      33        |
|     15%     |      26        |
|     10%     |      1A        |
|      5%     |      0D        |
|      0%     |      00        |
 ------------------------------
```

Si desea configurar el 45% al color rojo.

```
<color name="red_with_alpha_45">#73FF0000</color>
```

valor hexadecimal para rojo - # FF0000

Puede agregar 73 para una opacidad del 45% en el prefijo - # 73FF0000

Trabajando con el archivo strings.xml

Un recurso de cadena proporciona cadenas de texto para su aplicación con un estilo y formato de texto opcionales. Hay tres tipos de recursos que pueden proporcionar a su aplicación cadenas:

Cuerda

```
XML resource that provides a single string.
```

Sintaxis:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="string_name">text_string</string>
</resources>
```

Y para usar esta cadena en el diseño:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/string_name" />
```

Array de cuerdas

```
XML resource that provides an array of strings.
```

Sintaxis:

```
    <resources>
<string-array name="planets_array">
    <item>Mercury</item>
    <item>Venus</item>
    <item>Earth</item>
    <item>Mars</item>
</string-array>
```

Uso

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

Cantidad de cadenas (plurales)

```
XML resource that carries different strings for pluralization.
```

Sintaxis:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals
        name="plural_name">
        <item
            quantity=["zero" | "one" | "two" | "few" | "many" | "other"]
            >text_string</item>
    </plurals>
</resources>
```

Uso:

```
int count = getNumberOfsongsAvailable();
Resources res = getResources();
String songsFound = res.getQuantityString(R.plurals.plural_name, count, count);
```

Lea Recursos en línea: https://riptutorial.com/es/android/topic/108/recursos

Capítulo 207: RecyclerView

Introducción

RecyclerView es una versión más avanzada de la vista de lista con un rendimiento mejorado y características adicionales.

Parámetros

Parámetro	Detalle
Adaptador	Una subclase de RecyclerView.Adapter responsable de proporcionar vistas que representan elementos en un conjunto de datos
Posición	La posición de un elemento de datos dentro de un adaptador
Índice	El índice de una vista secundaria adjunta como se usa en una llamada para getChildAt (int). Contraste con la posición
Unión	El proceso de preparación de una vista secundaria para mostrar los datos correspondientes a una posición dentro del adaptador
Reciclar (ver)	Una vista utilizada anteriormente para mostrar datos para una posición de adaptador específica se puede colocar en una memoria caché para luego reutilizarla y mostrar el mismo tipo de datos más tarde. Esto puede mejorar drásticamente el rendimiento al omitir la construcción inicial o la inflación.
Chatarra (ver)	Una vista secundaria que ha entrado en un estado separado temporalmente durante el diseño. Las vistas de chatarra se pueden reutilizar sin separarse completamente de RecyclerView principal, ya sea sin modificar si el adaptador no requiere un nuevo encuadernado o si el adaptador considera que la vista está sucia
Sucio (ver)	Una vista secundaria que debe ser recuperada por el adaptador antes de mostrarse

Observaciones

`RecyclerView` es una vista flexible para proporcionar una ventana limitada a un conjunto de datos de gran tamaño.

Antes de usar `RecyclerView` , debe agregar la dependencia de la biblioteca de soporte en el archivo `build.gradle` :

```
dependencies {
```

```
    // Match the version of your support library dependency
    compile 'com.android.support:recyclerview-v7:25.3.1'
}
```

Puede encontrar el número de versión más reciente de recyclerview en el sitio oficial.

Otros temas relacionados:

Hay otros temas que describen los componentes de `RecyclerView` :

- RecyclerView LayoutManagers
- RecicladorVer artículoDecoraciones
- RecyclerView onClickListeners

Documentacion oficial

http://developer.android.com/reference/android/support/v7/widget/RecyclerView.html

Versiones anteriores:

```
//it requires compileSdkVersion 25
compile 'com.android.support:recyclerview-v7:25.2.0'
compile 'com.android.support:recyclerview-v7:25.1.0'
compile 'com.android.support:recyclerview-v7:25.0.0'

//it requires compileSdkVersion 24
compile 'com.android.support:recyclerview-v7:24.2.1'
compile 'com.android.support:recyclerview-v7:24.2.0'
compile 'com.android.support:recyclerview-v7:24.1.1'
compile 'com.android.support:recyclerview-v7:24.1.0'

//it requires compileSdkVersion 23
compile 'com.android.support:recyclerview-v7:23.4.0'
compile 'com.android.support:recyclerview-v7:23.3.0'
compile 'com.android.support:recyclerview-v7:23.2.1'
compile 'com.android.support:recyclerview-v7:23.2.0'
compile 'com.android.support:recyclerview-v7:23.1.1'
compile 'com.android.support:recyclerview-v7:23.1.0'
compile 'com.android.support:recyclerview-v7:23.0.1'
compile 'com.android.support:recyclerview-v7:23.0.0'

//it requires compileSdkVersion 22
compile 'com.android.support:recyclerview-v7:22.2.1'
compile 'com.android.support:recyclerview-v7:22.2.0'
compile 'com.android.support:recyclerview-v7:22.1.1'
compile 'com.android.support:recyclerview-v7:22.1.0'
compile 'com.android.support:recyclerview-v7:22.0.0'

//it requires compileSdkVersion 21
compile 'com.android.support:recyclerview-v7:21.0.3'
compile 'com.android.support:recyclerview-v7:21.0.2'
compile 'com.android.support:recyclerview-v7:21.0.0'
```

Examples

Añadiendo un RecyclerView

Agregue la dependencia como se describe en la sección Comentario, luego agregue `RecyclerView` a su diseño:

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

Una vez que haya agregado un widget `RecyclerView` a su diseño, obtenga un controlador para el objeto, conéctelo a un administrador de diseño y adjunte un adaptador para que se muestren los datos:

```
mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);

// set a layout manager (LinearLayoutManager in this example)

mLayoutManager = new LinearLayoutManager(getApplicationContext());
mRecyclerView.setLayoutManager(mLayoutManager);

// specify an adapter
mAdapter = new MyAdapter(myDataset);
mRecyclerView.setAdapter(mAdapter);
```

O simplemente configure el administrador de diseño desde xml agregando estas líneas:

```
xmlns:app="http://schemas.android.com/apk/res-auto"
app:layoutManager="android.support.v7.widget.LinearLayoutManager"
```

Si sabe que los cambios en el contenido de `RecyclerView` no cambiarán el tamaño del diseño de `RecyclerView` , use el siguiente código para mejorar el rendimiento del componente. Si RecyclerView tiene un tamaño fijo, sabe que RecyclerView no cambiará de tamaño debido a sus hijos, por lo que no llama en absoluto al diseño de la solicitud. Simplemente maneja el cambio en sí. Si se invalida lo que sea el padre, el coordinador, el diseño o lo que sea. (Puedes usar este método incluso antes de configurar `LayoutManager` y el `Adapter`):

```
mRecyclerView.setHasFixedSize(true);
```

`RecyclerView` proporciona estos gestores de diseño integrados para usar. Por lo tanto, puede crear una lista, una cuadrícula y una cuadrícula escalonada utilizando `RecyclerView` :

1. LinearLayoutManager muestra los elementos en una lista de desplazamiento vertical u horizontal.
2. GridLayoutManager muestra los elementos en una cuadrícula.
3. StaggeredGridLayoutManager muestra los elementos en una cuadrícula escalonada.

Carga más suave de artículos

Si los elementos en su `RecyclerView` cargan datos de la red (comúnmente imágenes) o realizan otro procesamiento, eso puede tomar una cantidad significativa de tiempo y puede terminar con elementos en pantalla pero sin carga completa. Para evitar esto, puede ampliar el `LinearLayoutManager` existente para precargar varios elementos antes de que se `LinearLayoutManager` en pantalla:

```
package com.example;

import android.content.Context;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.OrientationHelper;
import android.support.v7.widget.RecyclerView;

/**
 * A LinearLayoutManager that preloads items off-screen.
 * <p>
 * Preloading is useful in situations where items might take some time to load
 * fully, commonly because they have maps, images or other items that require
 * network requests to complete before they can be displayed.
 * <p>
 * By default, this layout will load a single additional page's worth of items,
 * a page being a pixel measure equivalent to the on-screen size of the
 * recycler view.  This can be altered using the relevant constructor, or
 * through the {@link #setPages(int)} method.
 */
public class PreLoadingLinearLayoutManager extends LinearLayoutManager {
  private int mPages = 1;
  private OrientationHelper mOrientationHelper;

  public PreLoadingLinearLayoutManager(final Context context) {
    super(context);
  }

  public PreLoadingLinearLayoutManager(final Context context, final int pages) {
    super(context);
    this.mPages = pages;
  }

  public PreLoadingLinearLayoutManager(final Context context, final int orientation, final
boolean reverseLayout) {
    super(context, orientation, reverseLayout);
  }

  @Override
  public void setOrientation(final int orientation) {
    super.setOrientation(orientation);
    mOrientationHelper = null;
  }

  /**
   * Set the number of pages of layout that will be preloaded off-screen,
   * a page being a pixel measure equivalent to the on-screen size of the
   * recycler view.
   * @param pages the number of pages; can be {@code 0} to disable preloading
   */
  public void setPages(final int pages) {
```

```
    this.mPages = pages;
  }

  @Override
  protected int getExtraLayoutSpace(final RecyclerView.State state) {
    if (mOrientationHelper == null) {
      mOrientationHelper = OrientationHelper.createOrientationHelper(this, getOrientation());
    }
    return mOrientationHelper.getTotalSpace() * mPages;
  }
}
```

Arrastrar y soltar y deslizar con RecyclerView

Puede implementar las funciones de deslizar para descartar y arrastrar y soltar con RecyclerView sin utilizar bibliotecas de terceros.
Solo use la clase `ItemTouchHelper` incluida en la biblioteca de soporte de RecyclerView.

Cree una instancia de `ItemTouchHelper` con la devolución de llamada `SimpleCallback` y dependiendo de la funcionalidad que admita, debe anular `onMove(RecyclerView, ViewHolder, ViewHolder)` y / o `onSwiped(ViewHolder, int)` y finalmente adjuntar a su `RecyclerView` .

```
ItemTouchHelper.SimpleCallback simpleItemTouchCallback = new ItemTouchHelper.SimpleCallback(0,
ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {

    @Override
    public void onSwiped(RecyclerView.ViewHolder viewHolder, int swipeDir) {
        // remove item from adapter
    }

    @Override
    public boolean onMove(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder,
RecyclerView.ViewHolder target) {
            final int fromPos = viewHolder.getAdapterPosition();
            final int toPos = target.getAdapterPosition();
            // move item in `fromPos` to `toPos` in adapter.
            return true;// true if moved, false otherwise
        }

};

ItemTouchHelper itemTouchHelper = new ItemTouchHelper(simpleItemTouchCallback);
itemTouchHelper.attachToRecyclerView(recyclerView);
```

Vale la pena mencionar que el constructor `SimpleCallback` aplica la misma estrategia de deslizamiento a todos los elementos en `RecyclerView` . En cualquier caso, es posible actualizar la dirección de deslizamiento predeterminada para elementos específicos simplemente anulando el método `getSwipeDirs(RecyclerView, ViewHolder)` .

Supongamos, por ejemplo, que nuestro `RecyclerView` incluye un `HeaderViewHolder` y que, obviamente, no queremos aplicarle el deslizamiento. Bastará con anular `getSwipeDirs` siguiente manera:

```
@Override
```

```
public int getSwipeDirs(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder) {
    if (viewHolder instanceof HeaderViewHolder) {
        // no swipe for header
        return 0;
    }
    // default swipe for all other items
    return super.getSwipeDirs(recyclerView, viewHolder);
}
```

Añadir encabezado / pie de página a un RecyclerView

Este es un código de adaptador de muestra.

```
public class SampleAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

private static final int FOOTER_VIEW = 1;

// Define a view holder for Footer view

public class FooterViewHolder extends ViewHolder {
    public FooterViewHolder(View itemView) {
        super(itemView);
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Do whatever you want on clicking the item
            }
        });
    }
}

// Now define the viewholder for Normal list item
public class NormalViewHolder extends ViewHolder {
    public NormalViewHolder(View itemView) {
        super(itemView);

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Do whatever you want on clicking the normal items
            }
        });
    }
}

// And now in onCreateViewHolder you have to pass the correct view
// while populating the list item.

@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

    View v;

    if (viewType == FOOTER_VIEW) {
        v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item_footer,
parent, false);

        FooterViewHolder vh = new FooterViewHolder(v);
```

```
        return vh;
    }

    v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item_normal, parent,
false);

    NormalViewHolder vh = new NormalViewHolder(v);

    return vh;
}

// Now bind the viewholders in onBindViewHolder
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {

    try {
        if (holder instanceof NormalViewHolder) {
            NormalViewHolder vh = (NormalViewHolder) holder;

            vh.bindView(position);
        } else if (holder instanceof FooterViewHolder) {
            FooterViewHolder vh = (FooterViewHolder) holder;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

// Now the critical part. You have return the exact item count of your list
// I've only one footer. So I returned data.size() + 1
// If you've multiple headers and footers, you've to return total count
// like, headers.size() + data.size() + footers.size()

@Override
public int getItemCount() {
    if (data == null) {
        return 0;
    }

    if (data.size() == 0) {
        //Return 1 here to show nothing
        return 1;
    }

    // Add extra view to show the footer view
    return data.size() + 1;
}

// Now define getItemViewType of your own.

@Override
public int getItemViewType(int position) {
    if (position == data.size()) {
        // This is where we'll add footer.
        return FOOTER_VIEW;
    }

    return super.getItemViewType(position);
}

// So you're done with adding a footer and its action on onClick.
```

```
// Now set the default ViewHolder for NormalViewHolder

public class ViewHolder extends RecyclerView.ViewHolder {
    // Define elements of a row here
    public ViewHolder(View itemView) {
        super(itemView);
        // Find view by ID and initialize here
    }

    public void bindView(int position) {
        // bindView() method to implement actions
    }
}
}
```

Aquí hay una buena lectura sobre la implementación de `RecyclerView` con encabezado y pie de página.

Metodo alternativo:

Si bien la respuesta anterior funcionará, también puede utilizar este enfoque utilizando una vista de reciclador utilizando un `NestedScrollView` . Puede agregar un diseño para el encabezado utilizando el siguiente enfoque:

```
<android.support.v4.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

     <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <include
            layout="@layout/drawer_view_header"
            android:id="@+id/navigation_header"/>

        <android.support.v7.widget.RecyclerView
            android:layout_below="@id/navigation_header"
            android:id="@+id/followers_list"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>

    </RelativeLayout>
</android.support.v4.widget.NestedScrollView>
```

O también puede usar un `LinearLayout` con alineación vertical en su `NestedScrollView` .

Nota: Esto solo funcionará con `RecyclerView` encima de **23.2.0**

```
compile 'com.android.support:recyclerview-v7:23.2.0'
```

Usando varios ViewHolders con ItemViewType

A veces, un RecyclerView necesitará usar varios tipos de vistas para mostrarse en la lista que se muestra en la interfaz de usuario, y cada vista necesita un diseño XML diferente para inflarse.

Para este problema, puede usar diferentes ViewHolders en un solo Adaptador, usando un método especial en RecyclerView - `getItemViewType(int position)` .

A continuación se muestra un ejemplo del uso de dos ViewHolders:

1. Un ViewHolder para mostrar las entradas de la lista
2. Un ViewHolder para mostrar múltiples vistas de encabezado

```
@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View itemView = LayoutInflater.from(context).inflate(viewType, parent, false);
    return ViewHolder.create(itemView, viewType);
}

@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
    final Item model = this.items.get(position);
    ((ViewHolder) holder).bind(model);
}

@Override
public int getItemViewType(int position) {
    return inSearchState ? R.layout.item_header : R.layout.item_entry;
}

abstract class ViewHolder {
    abstract void bind(Item model);

    public static ViewHolder create(View v, int viewType) {
        return viewType == R.layout.item_header ? new HeaderViewHolder(v) :new
EntryViewHolder(v);
    }
}

static class EntryViewHolder extends ViewHolder {
    private View v;

    public EntryViewHolder(View v) {
        this.v = v;
    }

    @Override public void bind(Item model) {
        // Bind item data to entry view.
    }
}

static class HeaderViewHolder extends ViewHolder {
    private View v;

    public HeaderViewHolder(View v) {
        this.v = v;
    }

    @Override public void bind(Item model) {
        // Bind item data to header view.
    }
}
```

Filtrar elementos dentro de RecyclerView con un SearchView

Agregar método de `filter` en `RecyclerView.Adapter` :

```
public void filter(String text) {
        if(text.isEmpty()){
            items.clear();
            items.addAll(itemsCopy);
        } else{
            ArrayList<PhoneBookItem> result = new ArrayList<>();
            text = text.toLowerCase();
            for(PhoneBookItem item: itemsCopy){
                //match by name or phone
                if(item.name.toLowerCase().contains(text) ||
item.phone.toLowerCase().contains(text)){
                    result.add(item);
                }
            }
            items.clear();
            items.addAll(result);
        }
        notifyDataSetChanged();
    }
```

`itemsCopy` se inicializa en el constructor del adaptador como `itemsCopy.addAll(items)` .

Si lo hace, sólo llame `filter` de `OnQueryTextListener` de `SearchView` :

```
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String query) {
        adapter.filter(query);
        return true;
    }

    @Override
    public boolean onQueryTextChange(String newText) {
        adapter.filter(newText);
        return true;
    }
});
```

Menú emergente con recyclerView

ponga este código dentro de su ViewHolder

nota: en este código estoy usando `btnExpand` click-event, para todo el evento de clic de `recyclerview` puede configurar el oyente en el objeto itemView.

```
public class MyViewHolder extends RecyclerView.ViewHolder{
        CardView cv;
        TextView recordName, visibleFile, date, time;
        Button btnIn, btnExpand;

        public MyViewHolder(final View itemView) {
```

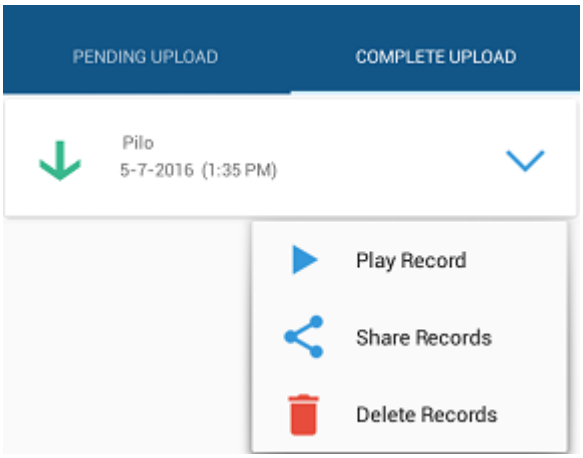
```
            super(itemView);

            cv = (CardView)itemView.findViewById(R.id.cardview);
            recordName = (TextView)itemView.findViewById(R.id.tv_record);
            visibleFile = (TextView)itemView.findViewById(R.id.visible_file);
            date = (TextView)itemView.findViewById(R.id.date);
            time = (TextView)itemView.findViewById(R.id.time);
            btnIn = (Button)itemView.findViewById(R.id.btn_in_out);

            btnExpand = (Button) itemView.findViewById(R.id.btn_expand);

            btnExpand.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    PopupMenu popup = new PopupMenu(btnExpand.getContext(), itemView);

                    popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
                        @Override
                        public boolean onMenuItemClick(MenuItem item) {
                            switch (item.getItemId()) {
                                case R.id.action_delete:
                                    moveFile(recordName.getText().toString(),
getAdapterPosition());
                                    return true;
                                case R.id.action_play:
                                    String valueOfPath = recordName.getText().toString();
                                    Intent intent = new Intent();
                                    intent.setAction(android.content.Intent.ACTION_VIEW);
                                    File file = new File(valueOfPath);
                                    intent.setDataAndType(Uri.fromFile(file), "audio/*");
                                    context.startActivity(intent);
                                    return true;
                                case R.id.action_share:
                                    String valueOfPath = recordName.getText().toString();
                                    File filee = new File(valueOfPath);
                                    try {
                                        Intent sendIntent = new Intent();
                                        sendIntent.setAction(Intent.ACTION_SEND);
                                        sendIntent.setType("audio/*");
                                        sendIntent.putExtra(Intent.EXTRA_STREAM,
Uri.fromFile(filee));
                                        context.startActivity(sendIntent);
                                    } catch (NoSuchMethodError | IllegalArgumentException |
NullPointerException e) {
                                        e.printStackTrace();
                                    } catch (Exception e) {
                                        e.printStackTrace();
                                    }
                                    return true;
                                default:
                                    return false;
                            }
                        }
                    });
                    // here you can inflate your menu
                    popup.inflate(R.menu.my_menu_item);
                    popup.setGravity(Gravity.RIGHT);

                    // if you want icon with menu items then write this try-catch block.
                    try {
                        Field mFieldPopup=popup.getClass().getDeclaredField("mPopup");
```

```
                        mFieldPopup.setAccessible(true);
                        MenuPopupHelper mPopup = (MenuPopupHelper) mFieldPopup.get(popup);
                        mPopup.setForceShowIcon(true);
                    } catch (Exception e) {

                    }
                    popup.show();
                }
            });

        }
    }
```

forma alternativa de mostrar iconos en el menú

```
try {
    Field[] fields = popup.getClass().getDeclaredFields();
    for (Field field : fields) {
        if ("mPopup".equals(field.getName())) {
            field.setAccessible(true);
            Object menuPopupHelper = field.get(popup);
            Class<?> classPopupHelper = Class.forName(menuPopupHelper
                .getClass().getName());
            Method setForceIcons = classPopupHelper.getMethod(
                "setForceShowIcon", boolean.class);
            setForceIcons.invoke(menuPopupHelper, true);
            break;
        }
    }
} catch (Exception e) {

}
```

Aquí está la salida:



Animar el cambio de datos

`RecyclerView` realizará una animación relevante si se utiliza alguno de los métodos de "notificar", excepto `notifyDataSetChanged` ; esto incluye `notifyItemChanged` , `notifyItemInserted` , `notifyItemMoved` , `notifyItemRemoved` , etc.

El adaptador debe extender esta clase en lugar de `RecyclerView.Adapter` .

```
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;

import java.util.List;

public abstract class AnimatedRecyclerAdapter<T, VH extends RecyclerView.ViewHolder>
        extends RecyclerView.Adapter<VH> {
    protected List<T> models;

    protected AnimatedRecyclerAdapter(@NonNull List<T> models) {
        this.models = models;
    }

    //Set new models.
    public void setModels(@NonNull final List<T> models) {
        applyAndAnimateRemovals(models);
        applyAndAnimateAdditions(models);
        applyAndAnimateMovedItems(models);
    }

    //Remove an item at position and notify changes.
    private T removeItem(int position) {
        final T model = models.remove(position);
        notifyItemRemoved(position);
        return model;
    }

    //Add an item at position and notify changes.
    private void addItem(int position, T model) {
        models.add(position, model);
        notifyItemInserted(position);
    }

    //Move an item at fromPosition to toPosition and notify changes.
    private void moveItem(int fromPosition, int toPosition) {
        final T model = models.remove(fromPosition);
        models.add(toPosition, model);
        notifyItemMoved(fromPosition, toPosition);
    }

    //Remove items that no longer exist in the new models.
    private void applyAndAnimateRemovals(@NonNull final List<T> newTs) {
        for (int i = models.size() - 1; i >= 0; i--) {
            final T model = models.get(i);
            if (!newTs.contains(model)) {
                removeItem(i);
            }
        }
    }

    //Add items that do not exist in the old models.
    private void applyAndAnimateAdditions(@NonNull final List<T> newTs) {
        for (int i = 0, count = newTs.size(); i < count; i++) {
            final T model = newTs.get(i);
            if (!models.contains(model)) {
                addItem(i, model);
            }
        }
    }

    //Move items that have changed their position.
```

```
    private void applyAndAnimateMovedItems(@NonNull final List<T> newTs) {
        for (int toPosition = newTs.size() - 1; toPosition >= 0; toPosition--) {
            final T model = newTs.get(toPosition);
            final int fromPosition = models.indexOf(model);
            if (fromPosition >= 0 && fromPosition != toPosition) {
                moveItem(fromPosition, toPosition);
            }
        }
    }
}
```

NO debe usar la misma `List` para `setModels` y `List` en el adaptador.

Usted declara los `models` como variables globales. `DataModel` es solo una clase ficticia.

```
private List<DataModel> models;
private YourAdapter adapter;
```

Inicialice los `models` antes de pasarlos al adaptador. `YourAdapter` es la implementación de `AnimatedRecyclerAdapter` .

```
models = new ArrayList<>();
//Add models
models.add(new DataModel());
//Do NOT pass the models directly. Otherwise, when you modify global models,
//you will also modify models in adapter.
//adapter = new YourAdapter(models); <- This is wrong.
adapter = new YourAdapter(new ArrayList(models));
```

Llama a esto después de que hayas actualizado tus `models` globales.

```
adapter.setModels(new ArrayList(models));
```

Si no anula `equals` , toda la comparación se compara por referencia.

Ejemplo usando SortedList

Android introdujo la clase `SortedList` poco después de que se introdujo `RecyclerView` . Esta clase maneja todas las llamadas de método de "notificación" al `RecyclerView.Adapter` para asegurar una animación adecuada, e incluso permite agrupar múltiples cambios, por lo que las animaciones no tiemblan.

```
import android.support.v7.util.SortedList;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.util.SortedListAdapterCallback;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import java.util.List;

public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {
```

```
    private SortedList<DataModel> mSortedList;

    class ViewHolder extends RecyclerView.ViewHolder {

        TextView text;
        CheckBox checkBox;

        ViewHolder(View itemView){
            super(itemView);

            //Initiate your code here...

        }

        void setDataModel(DataModel model) {
            //Update your UI with the data model passed here...
            text.setText(modle.getText());
            checkBox.setChecked(model.isChecked());
        }
    }

    public MyAdapter() {
        mSortedList = new SortedList<>(DataModel.class, new
SortedListAdapterCallback<DataModel>(this) {
            @Override
            public int compare(DataModel o1, DataModel o2) {
                //This gets called to find the ordering between objects in the array.
                if (o1.someValue() < o2.someValue()) {
                    return -1;
                } else if (o1.someValue() > o2.someValue()) {
                    return 1;
                } else {
                    return 0;
                }
            }

            @Override
            public boolean areContentsTheSame(DataModel oldItem, DataModel newItem) {
                //This is to see of the content of this object has changed. These items are
only considered equal if areItemsTheSame() returned true.

                //If this returns false, onBindViewHolder() is called with the holder
containing the item, and the item's position.
                return oldItem.getText().equals(newItem.getText()) && oldItem.isChecked() ==
newItem.isChecked();
            }

            @Override
            public boolean areItemsTheSame(DataModel item1, DataModel item2) {
                //Checks to see if these two items are the same. If not, it is added to the
list, otherwise, check if content has changed.
                return item1.equals(item2);
            }
        });
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = //Initiate your item view here.
        return new ViewHolder(itemView);
```

```
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        //Just update the holder with the object in the sorted list from the given position
        DataModel model = mSortedList.get(position);
        if (model != null) {
            holder.setDataModel(model);
        }
    }

    @Override
    public int getItemCount() {
        return mSortedList.size();
    }

    public void resetList(List<DataModel> models) {
        //If you are performing multiple changes, use the batching methods to ensure proper
animation.
        mSortedList.beginBatchedUpdates();
        mSortedList.clear();
        mSortedList.addAll(models);
        mSortedList.endBatchedUpdates();
    }

    //The following methods each modify the data set and automatically handles calling the
appropriate 'notify' method on the adapter.
    public void addModel(DataModel model) {
        mSortedList.add(model);
    }

    public void addModels(List<DataModel> models) {
        mSortedList.addAll(models);
    }

    public void clear() {
        mSortedList.clear();
    }

    public void removeModel(DataModel model) {
        mSortedList.remove(model);
    }

    public void removeModelAt(int i) {
        mSortedList.removeItemAt(i);
    }
}
```

RecyclerView con DataBinding

Aquí hay una clase de ViewHolder genérica que puede usar con cualquier diseño de DataBinding. Aquí se crea una instancia de una clase particular de `ViewDataBinding` utilizando el objeto de `View` inflado y la clase de utilidad `DataBindingUtil` .

```
import android.databinding.DataBindingUtil;
import android.support.v7.widget.RecyclerView;
import android.view.View;
```

```
public class BindingViewHolder<T> extends RecyclerView.ViewHolder{

    private final T binding;

    public BindingViewHolder(View itemView) {
        super(itemView);
        binding = (T)DataBindingUtil.bind(itemView);
    }

    public T getBinding() {
        return binding;
    }
}
```

Después de crear esta clase, puede usar `<layout>` en su archivo de diseño para habilitar el enlace de datos para ese diseño como este:

`file name: my_item.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>
        <variable
            name="item"
            type="ItemModel" />
    </data>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:text="@{item.itemLabel}" />
    </LinearLayout>
</layout>
```

y aquí está su modelo de datos de muestra:

```
public class ItemModel {
    public String itemLabel;
}
```

De forma predeterminada, la biblioteca de enlace de datos de Android genera una clase `ViewDataBinding` basada en el nombre del archivo de diseño, convirtiéndola en el caso de Pascal y con el sufijo "Enlace". Para este ejemplo, sería `MyItemBinding` para el archivo de diseño `my_item.xml` . Esa clase Binding también tendría un método de establecimiento para establecer el objeto definido como datos en el archivo de diseño (`ItemModel` para este ejemplo).

Ahora que tenemos todas las piezas podemos implementar nuestro adaptador de esta manera:

```
class MyAdapter extends RecyclerView.Adapter<BindingViewHolder<MyItemBinding>>{
    ArrayList<ItemModel> items = new ArrayList<>();
```

```
    public MyAdapter(ArrayList<ItemModel> items) {
        this.items = items;
    }

    @Override public BindingViewHolder<MyItemBinding> onCreateViewHolder(ViewGroup parent, int
viewType) {
        return new
BindingViewHolder<>(LayoutInflater.from(parent.getContext()).inflate(R.layout.my_item, parent,
false));
    }

    @Override public void onBindViewHolder(BindingViewHolder<ItemModel> holder, int position)
{
        holder.getBinding().setItemModel(items.get(position));
        holder.getBinding().executePendingBindings();
    }

    @Override public int getItemCount() {
        return items.size();
    }
}
```

Desplazamiento sin fin en Recycleview.

Aquí he compartido un fragmento de código para implementar el desplazamiento sin fin en la vista de reciclaje.

Paso 1: Primero haga un método abstracto en el adaptador de reciclaje como se muestra a continuación.

```
public abstract class ViewAllCategoryAdapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    public abstract void load();
}
```

Paso 2: ahora anule el método onBindViewHolder y `getItemCount()` de la clase ViewAllCategoryAdapter y llame al método `Load()` como se muestra a continuación.

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, final int position) {
    if ((position >= getItemCount() - 1)) {
        load();
    }
}

@Override
public int getItemCount() {
    return YOURLIST.size();
}
```

Paso 3: Ahora, toda la lógica de back-end está completa, ahora es el momento de ejecutar esta lógica. Es simple, puede anular el método de carga donde crea el objeto de su adaptador. Este método se llama automáticamente mientras el usuario llega al final de la lista.

```
adapter = new ViewAllCategoryAdapter(CONTEXT, YOURLIST) {
      @Override
      public void load() {

          /* do your stuff here */
          /* This method is automatically call while user reach at end of your list. */
      }
  };
  recycleCategory.setAdapter(adapter);
```

Ahora el método `load()` llama automáticamente mientras el usuario se desplaza al final de la lista.

La mejor de las suertes

Mostrar vista predeterminada hasta que los elementos se carguen o cuando los datos no estén disponibles

Captura de pantalla



Clase de adaptador

```
private class MyAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

final int EMPTY_VIEW = 77777;
List<CustomData> datalist = new ArrayList<>();
```

```
MyAdapter() {
    super();
}

@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

    LayoutInflater layoutInflater = LayoutInflater.from(parent.getContext());

    if (viewType == EMPTY_VIEW) {
        return new EmptyView(layoutInflater.inflate(R.layout.nothing_yet, parent, false));
    } else {
        return new ItemView(layoutInflater.inflate(R.layout.my_item, parent, false));
    }
}

@SuppressLint("SetTextI18n")
@Override
public void onBindViewHolder(final RecyclerView.ViewHolder holder, int position) {
    if (getItemViewType(position) == EMPTY_VIEW) {
        EmptyView emptyView = (EmptyView) holder;
        emptyView.primaryText.setText("No data yet");
        emptyView.secondaryText.setText("You're doing good !");
        emptyView.primaryText.setCompoundDrawablesWithIntrinsicBounds(null, new
IconicsDrawable(getActivity()).icon(FontAwesome.Icon.faw_ticket).sizeDp(48).color(Color.DKGRAY),
null, null);

    } else {
        ItemView itemView = (ItemView) holder;
        // Bind data to itemView
    }
}

@Override
public int getItemCount() {
    return datalist.size() > 0 ? datalist.size() : 1;
}

@Override
public int getItemViewType(int position) {
    if  datalist.size() == 0) {
        return EMPTY_VIEW;
    }
    return super.getItemViewType(position);
}

}
```

nothing_yet.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:orientation="vertical"
    android:paddingBottom="100dp"
    android:paddingTop="100dp">
```

```
    <TextView
        android:id="@+id/nothingPrimary"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:drawableTint="@android:color/secondary_text_light"
        android:drawableTop="@drawable/ic_folder_open_black_24dp"
        android:enabled="false"
        android:fontFamily="sans-serif-light"
        android:text="No Item's Yet"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="@android:color/secondary_text_light"
        android:textSize="40sp"
        tools:targetApi="m" />

    <TextView
        android:id="@+id/nothingSecondary"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:enabled="false"
        android:fontFamily="sans-serif-condensed"
        android:text="You're doing good !"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:textColor="@android:color/tertiary_text_light" />
</LinearLayout>
```

Estoy usando FontAwesome con Iconics Library para las imágenes. Agrega esto a tu archivo de nivel de aplicación build.gradle.

```
compile 'com.mikepenz:fontawesome-typeface:4.6.0.3@aar'
compile 'com.mikepenz:iconics-core:2.8.1@aar'
```

Agregue líneas divisorias a los artículos RecyclerView

Solo agrega estas líneas a la inicialización.

```
RecyclerView mRecyclerView = (RecyclerView) view.findViewById(recyclerView);
mRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
mRecyclerView.addItemDecoration(new DividerItemDecoration(getActivity(),
DividerItemDecoration.VERTICAL));
```

Agregue un `adapter` y llame a `.notifyDataSetChanged();` como siempre !
Esta no es una característica incorporada de Recyclerview pero se agrega en las bibliotecas de soporte. Así que no olvides incluir esto en tu archivo build.gradle a nivel de aplicación

```
compile "com.android.support:appcompat-v7:25.3.1"
compile "com.android.support:recyclerview-v7:25.3.1"
```

Se pueden agregar múltiples ItemDecorations a un solo RecyclerView.

Cambio de color del divisor :

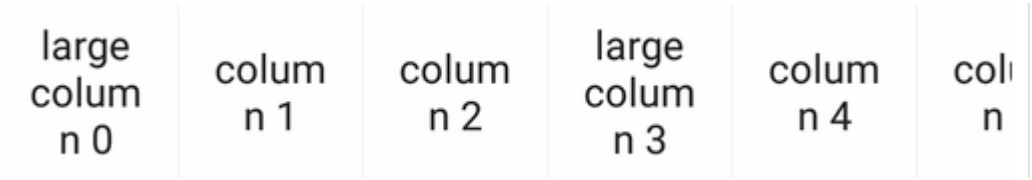
Es bastante fácil establecer un color para un elementoDecoración.

1. el paso es: crear un archivo `divider.xml` que se encuentra en la carpeta `drawable`

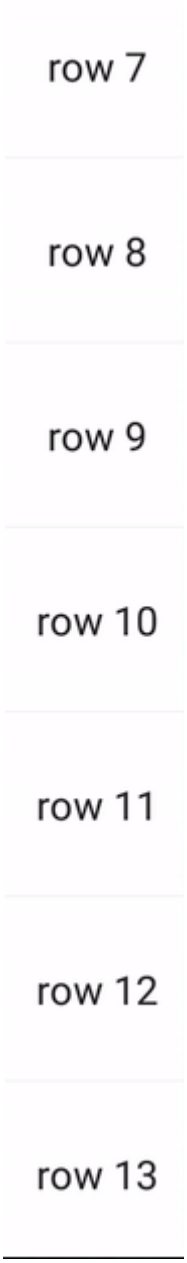
```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="line">
    <size
        android:width="1px"
        android:height="1px"/>
    <solid android:color="@color/divider_color"/>
</shape>
```

2. paso es: configuración dibujable

```
    // Get drawable object
    Drawable mDivider = ContextCompat.getDrawable(m_jContext, R.drawable.divider);
    // Create a DividerItemDecoration whose orientation is Horizontal
    DividerItemDecoration hItemDecoration = new DividerItemDecoration(m_jContext,
            DividerItemDecoration.HORIZONTAL);
    // Set the drawable on it
    hItemDecoration.setDrawable(mDivider);
```




```
    // Create a DividerItemDecoration whose orientation is vertical
    DividerItemDecoration vItemDecoration = new DividerItemDecoration(m_jContext,
            DividerItemDecoration.VERTICAL);
    // Set the drawable on it
    vItemDecoration.setDrawable(mDivider);
```



Lea RecyclerView en línea: https://riptutorial.com/es/android/topic/169/recyclerview

Capítulo 208: RecyclerView Decoraciones

Sintaxis

- RecyclerView addItemDecoration (RecyclerView.ItemDecoration decoration)
- RecyclerView addItemDecoration (RecyclerView.ItemDecoration decoration, int index)

Parámetros

Parámetro	Detalles
decoración	la decoración del artículo para agregar al `RecyclerView`
índice	El índice en la lista de decoraciones para este `RecyclerView` . Este es el orden en el que se `getItemOffset` y `onDraw` . Las llamadas posteriores pueden sobregirar las anteriores.

Observaciones

Las decoraciones son estáticas.

Dado que las decoraciones solo se dibujan, no es posible agregarles escuchas de clics u otra funcionalidad de UI.

Decoraciones multiples

La adición de múltiples decoraciones a un `RecyclerView` funcionará en algunos casos, pero actualmente no existe una API pública que tenga en cuenta otras decoraciones posibles al medir o dibujar. Puede obtener los límites de vista o los límites decorados de vista, donde los límites decorados son la suma de todas las compensaciones de decoración aplicadas.

Otros temas relacionados:

RecyclerView
RecyclerView onClickListeners

Oficial javadoc

https://developer.android.com/reference/android/support/v7/widget/RecyclerView.ItemDecoration.html

Examples

Dibujando un separador

Esto dibujará una línea en la parte inferior de cada vista, pero la última en actuar como un separador entre los elementos.

```
public class SeparatorDecoration extends RecyclerView.ItemDecoration {

    private final Paint mPaint;
    private final int mAlpha;

    public SeparatorDecoration(@ColorInt int color, float width) {
        mPaint = new Paint();
        mPaint.setColor(color);
        mPaint.setStrokeWidth(width);
        mAlpha = mPaint.getAlpha();
    }

    @Override
    public void getItemOffsets(Rect outRect, View view, RecyclerView parent,
RecyclerView.State state) {
        final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams)
view.getLayoutParams();

        // we retrieve the position in the list
        final int position = params.getViewAdapterPosition();

        // add space for the separator to the bottom of every view but the last one
        if (position < state.getItemCount()) {
            outRect.set(0, 0, 0, (int) mPaint.getStrokeWidth()); // left, top, right, bottom
        } else {
            outRect.setEmpty(); // 0, 0, 0, 0
        }
    }

    @Override
    public void onDraw(Canvas c, RecyclerView parent, RecyclerView.State state) {
        // a line will draw half its size to top and bottom,
        // hence the offset to place it correctly
        final int offset = (int) (mPaint.getStrokeWidth() / 2);

        // this will iterate over every visible view
        for (int i = 0; i < parent.getChildCount(); i++) {
            final View view = parent.getChildAt(i);
            final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams)
view.getLayoutParams();

            // get the position
            final int position = params.getViewAdapterPosition();

            // and finally draw the separator
            if (position < state.getItemCount()) {
                // apply alpha to support animations
                mPaint.setAlpha((int) (view.getAlpha() * mAlpha));

                float positionY = view.getBottom() + offset + view.getTranslationY();
                // do the drawing
```

```
                c.drawLine(view.getLeft() + view.getTranslationX(),
                        positionY,
                        view.getRight() + view.getTranslationX(),
                        positionY,
                        mPaint);
            }
        }
    }
}
```

Márgenes por artículo con ItemDecoration

Puede usar un `RecyclerView.ItemDecoration` para colocar márgenes adicionales alrededor de cada elemento en un RecyclerView. En algunos casos, esto puede limpiar tanto la implementación del adaptador como la vista de elementos XML.

```
public class MyItemDecoration
    extends RecyclerView.ItemDecoration {

    private final int extraMargin;

    @Override
    public void getItemOffsets(Rect outRect, View view,
            RecyclerView parent, RecyclerView.State state) {

        int position = parent.getChildAdapterPosition(view);

        // It's easy to put extra margin on the last item...
        if (position + 1 == parent.getAdapter().getItemCount()) {
            outRect.bottom = extraMargin; // unit is px
        }

        // ...or you could give each item in the RecyclerView different
        // margins based on its position...
        if (position % 2 == 0) {
            outRect.right = extraMargin;
        } else {
            outRect.left = extraMargin;
        }

        // ...or based on some property of the item itself
        MyListItem item = parent.getAdapter().getItem(position);
        if (item.isFirstItemInSection()) {
            outRect.top = extraMargin;
        }
    }

    public MyItemDecoration(Context context) {
        extraMargin = context.getResources()
                .getDimensionPixelOffset(R.dimen.extra_margin);
    }
}
```

Para habilitar la decoración, simplemente agrégala a tu RecyclerView:

```
// in your onCreate()
RecyclerView rv = (RecyclerView) findItemById(R.id.myList);
```

```
rv.addItemDecoration(new MyItemDecoration(context));
```

Añadir divisor a RecyclerView

En primer lugar, debe crear una clase que amplíe `RecyclerView.ItemDecoration` :

```
public class SimpleBlueDivider extends RecyclerView.ItemDecoration {
private Drawable mDivider;

public SimpleBlueDivider(Context context) {
    mDivider = context.getResources().getDrawable(R.drawable.divider_blue);
}

@Override
public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {
    //divider padding give some padding whatever u want or disable
    int left =parent.getPaddingLeft()+80;
    int right = parent.getWidth() - parent.getPaddingRight()-30;

    int childCount = parent.getChildCount();
    for (int i = 0; i < childCount; i++) {
        View child = parent.getChildAt(i);

        RecyclerView.LayoutParams params = (RecyclerView.LayoutParams)
child.getLayoutParams();

        int top = child.getBottom() + params.bottomMargin;
        int bottom = top + mDivider.getIntrinsicHeight();

        mDivider.setBounds(left, top, right, bottom);
        mDivider.draw(c);
    }
}

}
```

Agregue `divider_blue.xml` a su carpeta `divider_blue.xml` :

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle">
<size android:width="1dp" android:height="4dp" />
<solid android:color="#AA123456" />
</shape>
```

Entonces úsalo como:

```
recyclerView.addItemDecoration(new SimpleBlueDivider(context));
```

El resultado será como:


28% 14:20
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)

Esta imagen es solo un ejemplo de cómo funcionan los separadores. Si desea seguir las especificaciones de Diseño de materiales al agregar separadores, eche un vistazo a este enlace: separadores y gracias a @Brenden Kromhout al proporcionar el enlace.

Cómo agregar divisores usando y DividerItemDecoration

`DividerItemDecoration` es un `RecyclerView.ItemDecoration` que se puede usar como divisor entre los elementos.

```
DividerItemDecoration mDividerItemDecoration = new DividerItemDecoration(context,
             mLayoutManager.getOrientation());
recyclerView.addItemDecoration(mDividerItemDecoration);
```

Admite ambas orientaciones usando `DividerItemDecoration.VERTICAL` y `DividerItemDecoration.HORIZONTAL` .

ItemOffsetDecoration para GridLayoutManager en RecycleView

El siguiente ejemplo ayudará a dar el mismo espacio a un elemento en GridLayout.

ItemOffsetDecoration.java

```
public class ItemOffsetDecoration extends RecyclerView.ItemDecoration {

    private int mItemOffset;

    private int spanCount = 2;

    public ItemOffsetDecoration(int itemOffset) {
        mItemOffset = itemOffset;
    }

    public ItemOffsetDecoration(@NonNull Context context, @DimenRes int itemOffsetId) {
        this(context.getResources().getDimensionPixelSize(itemOffsetId));
    }

    @Override
    public void getItemOffsets(Rect outRect, View view, RecyclerView parent,
                               RecyclerView.State state) {
        super.getItemOffsets(outRect, view, parent, state);

        int position = parent.getChildLayoutPosition(view);

        GridLayoutManager manager = (GridLayoutManager) parent.getLayoutManager();

        if (position < manager.getSpanCount())
            outRect.top = mItemOffset;

        if (position % 2 != 0) {
            outRect.right = mItemOffset;
        }

        outRect.left = mItemOffset;
        outRect.bottom = mItemOffset;
    }
```

```
}
```

Puede llamar a ItemDecoration como el código de abajo.

```
recyclerView = (RecyclerView) view.findViewById(R.id.recycler_view);

GridLayoutManager lLayout = new GridLayoutManager(getActivity(), 2);

ItemOffsetDecoration itemDecoration = new ItemOffsetDecoration(mActivity,
R.dimen.item_offset);
recyclerView.addItemDecoration(itemDecoration);

recyclerView.setLayoutManager(lLayout);
```

y ejemplo de desplazamiento de elemento

```
<dimen name="item_offset">5dp</dimen>
```

Lea RecyclerView Decoraciones en línea: https://riptutorial.com/es/android/topic/506/recyclerview-decoraciones

Capítulo 209: RecyclerView onClickListeners

Examples

Nuevo ejemplo

```
public class SampleAdapter extends RecyclerView.Adapter<SampleAdapter.ViewHolder> {

    private String[] mDataSet;
    private OnRVItemClickListener mListener;

    /**
     * Provide a reference to the type of views that you are using (custom ViewHolder)
     */
    public static class ViewHolder extends RecyclerView.ViewHolder {
        private final TextView textView;

        public ViewHolder(View v) {
            super(v);
            // Define click listener for the ViewHolder's View.
            v.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) { // handle click events here
                    Log.d(TAG, "Element " + getPosition() + " clicked.");
                    mListener.onRVItemClicked(getPosition(),v); //set callback
                }
            });
            textView = (TextView) v.findViewById(R.id.textView);
        }

        public TextView getTextView() {
            return textView;
        }
    }

    /**
     * Initialize the dataset of the Adapter.
     *
     * @param dataSet String[] containing the data to populate views to be used by
RecyclerView.
     */
    public SampleAdapter(String[] dataSet) {
        mDataSet = dataSet;
    }

    // Create new views (invoked by the layout manager)
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
        // Create a new view.
        View v = LayoutInflater.from(viewGroup.getContext())
                .inflate(R.layout.text_row_item, viewGroup, false);

        return new ViewHolder(v);
    }

    // Replace the contents of a view (invoked by the layout manager)
    @Override
```

```
    public void onBindViewHolder(ViewHolder viewHolder, final int position) {
        // Get element from your dataset at this position and replace the contents of the view
        // with that element
        viewHolder.getTextView().setText(mDataSet[position]);
    }

    // Return the size of your dataset (invoked by the layout manager)
    @Override
    public int getItemCount() {
        return mDataSet.length;
    }

    public void setOnRVClickListener(OnRVItemClickListener) {
        mListener = OnRVItemClickListener;
    }

    public interface OnRVItemClickListener {
       void onRVItemClicked(int position, View v);
    }
}
```

Ejemplo de Kotlin y RxJava.

Primer ejemplo reimplementado en Kotlin y usando RxJava para una interacción más limpia.

```
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.support.v7.widget.RecyclerView
import rx.subjects.PublishSubject

public class SampleAdapter(private val items: Array<String>) :
RecyclerView.Adapter<SampleAdapter.ViewHolder>() {

    // change to different subjects from rx.subjects to get different behavior
    // BehaviorSubject for example allows to receive last event on subscribe
    // PublishSubject sends events only after subscribing on the other hand which is desirable
for clicks
    public val itemClickStream: PublishSubject<View> = PublishSubject.create()

    override fun getItemCount(): Int {
        return items.size
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder? {
        val v = LayoutInflater.from(parent.getContext()).inflate(R.layout.text_row_item,
parent, false);
        return ViewHolder(view)
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        holder.bind(items[position])
    }

    public inner class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        private val textView: TextView by lazy { view.findViewById(R.id.textView) as TextView
}

        init {
```

```
            view.setOnClickListener { v -> itemClickStream.onNext(v) }
        }

        fun bind(text: String) {
            textView.text = text
        }
    }
}
```

El uso es bastante simple entonces. Es posible suscribirse en un hilo separado utilizando las instalaciones de RxJava.

```
val adapter = SampleAdapter(arrayOf("Hello", "World"))
adapter.itemClickStream.subscribe { v ->
    if (v.id == R.id.textView) {
        // do something
    }
}
```

Ejemplo fácil de OnLongClick y OnClick

En primer lugar, implemente el titular de su vista:

```
implements View.OnClickListener, View.OnLongClickListener
```

Luego, registre a los oyentes de la siguiente manera:

```
itemView.setOnClickListener(this);
itemView.setOnLongClickListener(this);
```

A continuación, anule a los oyentes de la siguiente manera:

```
@Override
public void onClick(View v) {
    oncliclistner.onItemClick(getAdapterPosition(), v);
}

@Override
public boolean onLongClick(View v) {
    oncliclistner.onItemLongClick(getAdapterPosition(), v);
    return true;
}
```

Y finalmente, agregue el siguiente código:

```
public void setOnItemClickListener(onClickListner oncliclistner) {
    SampleAdapter.oncliclistner = oncliclistner;
}

public void setHeader(View v) {
    this.headerView = v;
}
```

```
public interface onClickListner {
    void onItemClick(int position, View v);
    void onItemLongClick(int position, View v);
}
```

Demostración del adaptador

```
package adaptor;

import android.annotation.SuppressLint;
import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.wings.example.recycleview.MainActivity;
import com.wings.example.recycleview.R;

import java.util.ArrayList;

public class SampleAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {
    Context context;
    private ArrayList<String> arrayList;
    private static onClickListner onclicklistner;
    private static final int VIEW_HEADER = 0;
    private static final int VIEW_NORMAL = 1;
    private View headerView;

    public SampleAdapter(Context context) {
        this.context = context;
        arrayList = MainActivity.arrayList;
    }

    public class HeaderViewHolder extends RecyclerView.ViewHolder {
        public HeaderViewHolder(View itemView) {
            super(itemView);
        }
    }

    public class ItemViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener, View.OnLongClickListener {
        TextView txt_pos;
        SampleAdapter sampleAdapter;

        public ItemViewHolder(View itemView, SampleAdapter sampleAdapter) {
            super(itemView);

            itemView.setOnClickListener(this);
            itemView.setOnLongClickListener(this);

            txt_pos = (TextView) itemView.findViewById(R.id.txt_pos);
            this.sampleAdapter = sampleAdapter;

            itemView.setOnClickListener(this);
        }
```

```
        @Override
        public void onClick(View v) {
            oncliclistner.onItemClick(getAdapterPosition(), v);
        }

        @Override
        public boolean onLongClick(View v) {
            oncliclistner.onItemLongClick(getAdapterPosition(), v);
            return true;
        }
    }

    public void setOnItemClickListener(onClickListner oncliclistner) {
        SampleAdapter.oncliclistner = oncliclistner;
    }

    public void setHeader(View v) {
        this.headerView = v;
    }

    public interface onClickListner {
        void onItemClick(int position, View v);
        void onItemLongClick(int position, View v);
    }

    @Override
    public int getItemCount() {
        return arrayList.size()+1;
    }

    @Override
    public int getItemViewType(int position) {
        return position == 0 ? VIEW_HEADER : VIEW_NORMAL;
    }

    @SuppressLint("InflateParams")
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
        if (viewType == VIEW_HEADER) {
            return new HeaderViewHolder(headerView);
        } else {
            View view =
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.custom_recycler_row_sample_item,
viewGroup, false);
            return new ItemViewHolder(view, this);
        }
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
        if (viewHolder.getItemViewType() == VIEW_HEADER) {
            return;
        } else {
            ItemViewHolder itemViewHolder = (ItemViewHolder) viewHolder;
            itemViewHolder.txt_pos.setText(arrayList.get(position-1));
        }
    }
}
```

El código de ejemplo anterior se puede llamar mediante el siguiente código:

```
sampleAdapter.setOnItemClickListener(new SampleAdapter.onClickListner() {
    @Override
    public void onItemClick(int position, View v) {
        position = position+1;//As we are adding header
        Log.e(TAG + "ON ITEM CLICK", position + "");
        Snackbar.make(v, "On item click "+position, Snackbar.LENGTH_LONG).show();
    }

    @Override
    public void onItemLongClick(int position, View v) {
        position = position+1;//As we are adding header
        Log.e(TAG + "ON ITEM LONG CLICK", position + "");
        Snackbar.make(v, "On item longclick  "+position, Snackbar.LENGTH_LONG).show();
    }
});
```

Artículo Click Listeners

Para implementar un elemento, haga clic en escucha y / o en un elemento haga clic en escucha, puede crear una interfaz en su adaptador:

```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.ViewHolder> {

    public interface OnItemClickListener {

        void onItemSeleted(int position, View view, CustomObject object);
    }

    public interface OnItemLongClickListener {

        boolean onItemSelected(int position, View view, CustomObject object);
    }

    public final class ViewHolder extends RecyclerView.ViewHolder {

        public ViewHolder(View itemView) {
            super(itemView);
            final int position = getAdapterPosition();

            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    if(mOnItemClickListener != null) {
                        mOnItemClickListener.onItemSeleted(position, view,
mDataSet.get(position));
                    }
                }
            });

            itemView.setOnLongClickListener(new View.OnLongClickListener() {
                @Override
                public boolean onLongClick(View view) {
                    if(mOnItemLongClickListener != null) {
                        return mOnItemLongClickListener.onItemSelected(position, view,
mDataSet.get(position));
                    }
                }
            });
```

```
        }
    }

    private List<CustomObject> mDataSet;

    private OnItemClickListener mOnItemClickListener;
    private OnItemLongClickListener mOnItemLongClickListener;

    public CustomAdapter(List<CustomObject> dataSet) {
        mDataSet = dataSet;
    }

    @Override
    public CustomAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.view_item_custom, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(CustomAdapter.ViewHolder holder, int position) {
        // Bind views
    }

    @Override
    public int getItemCount() {
        return mDataSet.size();
    }

    public void setOnItemClickListener(OnItemClickListener listener) {
        mOnItemClickListener = listener;
    }

    public void setOnItemLongClickListener(OnItemLongClickListener listener) {
        mOnItemLongClickListener = listener;
    }

}
```

Luego puede configurar sus escuchas de clics después de crear una instancia del adaptador:

```
customAdapter.setOnItemClickListener(new CustomAdapter.OnItemClickListener {
    @Override
    public void onItemSelected(int position, View view, CustomObject object) {
        // Your implementation here
    }
});

customAdapter.setOnItemLongClickListener(new CustomAdapter.OnItemLongClickListener {
    @Override
    public boolean onItemSelected(int position, View view, CustomObject object) {
        // Your implementation here
        return true;
    }
});
```

Otra forma de implementar Item Click Listener

Otra forma de implementar el elemento de escucha de clics es utilizar la interfaz con varios métodos, cuyo número es igual al número de vistas en las que se puede hacer clic, y usar los escuchas de clics anulados, como puede ver a continuación. Este método es más flexible, ya que puede configurar escuchas de clic para diferentes vistas y controlar la lógica de clic por separado para cada una.

```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.CustomHolder> {

    private ArrayList<Object> mObjects;
    private ClickInterface mClickInterface;

    public interface ClickInterface {
        void clickEventOne(Object obj);
        void clickEventTwo(Object obj1, Object obj2);
    }

    public void setClickInterface(ClickInterface clickInterface) {
        mClickInterface = clickInterface;
    }

    public CustomAdapter(){
        mList = new ArrayList<>();
    }

    public void addItems(ArrayList<Object> objects) {
        mObjects.clear();
        mObjects.addAll(objects);
        notifyDataSetChanged();
    }

    @Override
    public CustomHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(parent.getContext())
                               .inflate(R.layout.list_item, parent, false);
        return new CustomHolder(v);
    }

    @Override
    public void onBindViewHolder(CustomHolder holder, int position) {
        //make all even positions not clickable
        holder.firstClickListener.setClickable(position%2==0);
        holder.firstClickListener.setPosition(position);
        holder.secondClickListener.setPosition(position);
    }


    private class FirstClickListener implements View.OnClickListener {
        private int mPosition;
        private boolean mClickable;

        void setPosition(int position) {
            mPosition = position;
        }

        void setClickable(boolean clickable) {
            mPosition = position;
        }

        @Override
```

```
        public void onClick(View v) {
            if(mClickable) {
                mClickInterface.clickEventOne(mObjects.get(mPosition));
            }
        }
    }

    private class SecondClickListener implements View.OnClickListener {
        private int mPosition;

        void setPosition(int position) {
            mPosition = position;
        }

        @Override
        public void onClick(View v) {
            mClickInterface.clickEventTwo(mObjects.get(mPosition), v);
        }
    }

    @Override
    public int getItemCount() {
        return mObjects.size();
    }

    protected class CustomHolder extends RecyclerView.ViewHolder {
        FirstClickListener firstClickListener;
        SecondClickListener secondClickListener;
        View v1, v2;

        public DialogHolder(View itemView) {
            super(itemView);
            v1 = itemView.findViewById(R.id.v1);
            v2 = itemView.findViewById(R.id.v2);
            firstClickListener = new FirstClickListener();
            secondClickListener = new SecondClickListener();

            v1.setOnClickListener(firstClickListener);
            v2.setOnClickListener(secondClickListener);
        }
    }
}
```

Y cuando tiene una instancia de adaptador, puede configurar su agente de escucha de clics que escucha hacer clic en cada una de las vistas:

```
customAdapter.setClickInterface(new CustomAdapter.ClickInterface {
    @Override
    public void clickEventOne(Object obj) {
        // Your implementation here
    }
    @Override
    public void clickEventTwo(Object obj1, Object obj2) {
        // Your implementation here
    }
});
```

RecyclerView Click listener

```
public  class RecyclerTouchListener implements RecyclerView.OnItemTouchListener {

    private GestureDetector gestureDetector;
    private RecyclerTouchListener.ClickListener clickListener;

    public RecyclerTouchListener(Context context, final RecyclerView recyclerView, final
RecyclerTouchListener.ClickListener clickListener) {
        this.clickListener = clickListener;

        gestureDetector = new GestureDetector(context, new
GestureDetector.SimpleOnGestureListener() {
            @Override
            public boolean onSingleTapUp(MotionEvent e) {
                return true;
            }
            @Override
            public void onLongPress(MotionEvent e) {
                View child = recyclerView.findChildViewUnder(e.getX(), e.getY());
                if (child != null && clickListener != null) {
                    clickListener.onLongClick(child, recyclerView.getChildPosition(child));
                }
            }
        });
    }


    @Override
    public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {
        View child = rv.findChildViewUnder(e.getX(), e.getY());
        if (child != null && clickListener != null && gestureDetector.onTouchEvent(e)) {
            clickListener.onClick(child, rv.getChildPosition(child));
        }
        return false;
    }

    @Override
    public void onTouchEvent(RecyclerView rv, MotionEvent e) {

    }

    @Override
    public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {

    }

    public interface ClickListener {
        void onLongClick(View child, int childPosition);

        void onClick(View child, int childPosition);
    }
}
```

En mainActivity

```
RecyclerView recyclerView =(RecyclerView) findViewById(R.id.recyclerview);
 recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getActivity(),recyclerView, new
RecyclerTouchListener.ClickListener() {
      @Override
      public void onLongClick(View child, int childPosition) {
```

```
      }

      @Override
      public void onClick(View child, int childPosition) {


     }
    }));
```

Lea RecyclerView onClickListeners en línea:
https://riptutorial.com/es/android/topic/96/recyclerview-onclicklisteners

Capítulo 210: RecyclerView y LayoutManagers

Examples

GridLayoutManager con recuento dinámico de span

Al crear una recyclerview con un administrador de diseño de gridlayout, debe especificar el recuento de intervalo en el constructor. El recuento de span se refiere al número de columnas. Esto es bastante torpe y no tiene en cuenta los tamaños de pantalla más grandes o la orientación de la pantalla. Un enfoque es crear diseños múltiples para los distintos tamaños de pantalla. Otro enfoque más dinámico se puede ver a continuación.

Primero creamos una clase personalizada RecyclerView de la siguiente manera:

```
public class AutofitRecyclerView extends RecyclerView {
    private GridLayoutManager manager;
    private int columnWidth = -1;

    public AutofitRecyclerView(Context context) {
        super(context);
        init(context, null);
    }

    public AutofitRecyclerView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(context, attrs);
    }

    public AutofitRecyclerView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init(context, attrs);
    }

    private void init(Context context, AttributeSet attrs) {
        if (attrs != null) {
            int[] attrsArray = {
                    android.R.attr.columnWidth
            };
            TypedArray array = context.obtainStyledAttributes(attrs, attrsArray);
            columnWidth = array.getDimensionPixelSize(0, -1);
            array.recycle();
        }

        manager = new GridLayoutManager(getContext(), 1);
        setLayoutManager(manager);
    }

    @Override
    protected void onMeasure(int widthSpec, int heightSpec) {
        super.onMeasure(widthSpec, heightSpec);
        if (columnWidth > 0) {
            int spanCount = Math.max(1, getMeasuredWidth() / columnWidth);
```

```
            manager.setSpanCount(spanCount);
        }
    }
}
```

Esta clase determina cuántas columnas pueden caber en el recyclerview. Para usarlo, deberá colocarlo en su layout.xml de la siguiente manera:

```
<?xml version="1.0" encoding="utf-8"?>
<com.path.to.your.class.autofitRecyclerView.AutofitRecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/auto_fit_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnWidth="200dp"
    android:clipToPadding="false"
    />
```

Observe que usamos el atributo columnWidth. La recyclerview lo necesitará para determinar cuántas columnas cabrán en el espacio disponible.

En su actividad / fragmento, solo obtiene una referencia a recylerview y le asigna un adaptador (y las decoraciones o animaciones de elementos que desee agregar). **NO CONFIGURAR UN GERENTE DE DISEÑO**

```
RecyclerView recyclerView = (RecyclerView) findViewById(R.id.auto_fit_recycler_view);
recyclerView.setAdapter(new MyAdapter());
```

(donde MyAdapter es su clase de adaptador)

Ahora tiene una vista de reciclaje que ajustará el número de usuarios (es decir, columnas) para que se ajuste al tamaño de la pantalla. Como adición final, es posible que desee centrar las columnas en recyclerview (de forma predeterminada, están alineadas con layout_start). Puedes hacerlo modificando un poco la clase AutofitRecyclerView. Comience creando una clase interna en el recyclerview. Esta será una clase que se extiende desde GridLayoutManager. Agregará suficiente relleno a la izquierda y la derecha para centrar las filas:

```
public class AutofitRecyclerView extends RecyclerView {

    // etc see above

    private class CenteredGridLayoutManager extends GridLayoutManager {

        public CenteredGridLayoutManager(Context context, AttributeSet attrs, int
defStyleAttr, int defStyleRes) {
            super(context, attrs, defStyleAttr, defStyleRes);
        }

        public CenteredGridLayoutManager(Context context, int spanCount) {
            super(context, spanCount);
        }

        public CenteredGridLayoutManager(Context context, int spanCount, int orientation,
boolean reverseLayout) {
```

```
            super(context, spanCount, orientation, reverseLayout);
        }

        @Override
        public int getPaddingLeft() {
            final int totalItemWidth = columnWidth * getSpanCount();
            if (totalItemWidth >= AutofitRecyclerView.this.getMeasuredWidth()) {
                return super.getPaddingLeft(); // do nothing
            } else {
                return Math.round((AutofitRecyclerView.this.getMeasuredWidth() / (1f +
getSpanCount())) - (totalItemWidth / (1f + getSpanCount())));
            }
        }

        @Override
        public int getPaddingRight() {
            return getPaddingLeft();
        }
    }
}
```

Luego, cuando configura LayoutManager en AutofitRecyclerView, use CenteredGridLayoutManager de la siguiente manera:

```
private void init(Context context, AttributeSet attrs) {
    if (attrs != null) {
        int[] attrsArray = {
                android.R.attr.columnWidth
        };
        TypedArray array = context.obtainStyledAttributes(attrs, attrsArray);
        columnWidth = array.getDimensionPixelSize(0, -1);
        array.recycle();
    }

    manager = new CenteredGridLayoutManager(getContext(), 1);
    setLayoutManager(manager);
}
```

¡Y eso es! Usted tiene un recuento basado en el centro de gestión de cuadrantes alineado dinámico basado en administrador de recortes.

Fuentes:

- Blog de la isla cuadrada de Chiu-Ki Chan
- Desbordamiento de pila

Agregar vista de encabezado a recyclerview con el administrador de gridlayout

Para agregar un encabezado a una recyclerview con un gridlayout, primero se debe informar al adaptador que la vista del encabezado es la primera posición, en lugar de la celda estándar utilizada para el contenido. A continuación, se debe informar al administrador de diseño que la primera posición debe tener un intervalo igual al * recuento de intervalos de toda la lista. *

Tome una clase regular RecyclerView.Adapter y configúrela de la siguiente manera:

```
public class HeaderAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

    private static final int ITEM_VIEW_TYPE_HEADER = 0;
    private static final int ITEM_VIEW_TYPE_ITEM = 1;

    private List<YourModel> mModelList;

    public HeaderAdapter (List<YourModel> modelList) {
        mModelList = modelList;
    }

    public boolean isHeader(int position) {
        return position == 0;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflater inflater = LayoutInflater.from(parent.getContext());

        if (viewType == ITEM_VIEW_TYPE_HEADER) {
            View headerView = inflater.inflate(R.layout.header, parent, false);
            return new HeaderHolder(headerView);
        }

        View cellView = inflater.inflate(R.layout.gridcell, parent, false);
        return new ModelHolder(cellView);
    }

    @Override
    public int getItemViewType(int position) {
        return isHeader(position) ? ITEM_VIEW_TYPE_HEADER : ITEM_VIEW_TYPE_ITEM;
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder h, int position) {
        if (isHeader(position)) {
            return;
        }

        final YourModel model = mModelList.get(position -1 ); // Subtract 1 for header

        ModelHolder holder = (ModelHolder) h;
        // populate your holder with data from your model as usual
    }

    @Override
    public int getItemCount() {
        return _categories.size() + 1; // add one for the header
    }
}
```

Luego en la actividad / fragmento:

```
final HeaderAdapter adapter = new HeaderAdapter (mModelList);
final GridLayoutManager manager = new GridLayoutManager();
manager.setSpanSizeLookup(new GridLayoutManager.SpanSizeLookup() {
    @Override
    public int getSpanSize(int position) {
        return adapter.isHeader(position) ? manager.getSpanCount() : 1;
    }
```

```
});
mRecyclerView.setLayoutManager(manager);
mRecyclerView.setAdapter(adapter);
```

Se puede utilizar el mismo método para agregar un pie de página además de o en lugar de un encabezado.

Fuente: blog Square Island de Chiu-Ki Chan.

Lista simple con LinearLayoutManager

Este ejemplo agrega una lista de lugares con imagen y nombre usando una `ArrayList` de objetos personalizados de `Place` como conjunto de datos.

Diseño de la actividad

El diseño de la actividad / fragmento o donde se utiliza RecyclerView solo tiene que contener RecyclerView. No hay ScrollView o un diseño específico necesario.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.v7.widget.RecyclerView
        android:id="@+id/my_recycler_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</RelativeLayout>
```

Definir el modelo de datos.

Puede usar cualquier clase o tipo de datos primitivos como modelo, como `int` , `String` , `float[]` o `CustomObject` . RecyclerView se referirá a una `List` de estos objetos / primitivos.

Cuando un elemento de la lista hace referencia a diferentes tipos de datos como texto, números, imágenes (como en este ejemplo con lugares), a menudo es una buena idea usar un objeto personalizado.

```
public class Place {
    // these fields will be shown in a list item
    private Bitmap image;
    private String name;

    // typical constructor
    public Place(Bitmap image, String name) {
        this.image = image;
        this.name = name;
    }
```

```
    // getters
    public Bitmap getImage() {
        return image;
    }
    public String getName() {
        return name;
    }
}
```

Lista de elementos de diseño

Debe especificar un archivo de diseño xml que se utilizará para cada elemento de la lista. En este ejemplo, se utiliza un `ImageView` para la imagen y un `TextView` para el nombre. `LinearLayout` coloca el `ImageView` a la izquierda y el `TextView` a la derecha de la imagen.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:orientation="horizontal"
    android:padding="8dp">

    <ImageView
        android:id="@+id/image"
        android:layout_width="36dp"
        android:layout_height="36dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp" />

    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Crear un adaptador RecyclerView y ViewHolder

A continuación, debe heredar `RecyclerView.Adapter` y `RecyclerView.ViewHolder` . Una estructura de clase habitual sería:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    public class ViewHolder extends RecyclerView.ViewHolder {
        // ...
    }
}
```

Primero, implementamos el `ViewHolder` . Solo hereda el constructor predeterminado y guarda las vistas necesarias en algunos campos:

```
public class ViewHolder extends RecyclerView.ViewHolder {
    private ImageView imageView;
    private TextView nameView;

    public ViewHolder(View itemView) {
        super(itemView);

        imageView = (ImageView) itemView.findViewById(R.id.image);
        nameView = (TextView) itemView.findViewById(R.id.name);
    }
}
```

El constructor del adaptador establece el conjunto de datos utilizado:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    private List<Place> mPlaces;

    public PlaceListAdapter(List<Place> contacts) {
        mPlaces = contacts;
    }

    // ...
}
```

Para usar el diseño de nuestro elemento de lista personalizado, reemplazamos el método `onCreateViewHolder(...)` . En este ejemplo, el archivo de diseño se llama `place_list_item.xml` .

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(
                R.layout.place_list_item,
                parent,
                false
        );
        return new ViewHolder(view);
    }

    // ...
}
```

En `onBindViewHolder(...)` , configuramos el contenido de las vistas. Obtenemos el modelo usado al encontrarlo en la `List` en la posición dada y luego configuramos la imagen y el nombre en las vistas de `ViewHolder` .

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public void onBindViewHolder(PlaceListAdapter.ViewHolder viewHolder, int position) {
```

```
        Place place = mPlaces.get(position);

        viewHolder.nameView.setText(place.getName());
        viewHolder.imageView.setImageBitmap(place.getImage());
    }

    // ...
}
```

También necesitamos implementar `getItemCount()` , que simplemente devuelve el tamaño de la `List` .

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public int getItemCount() {
        return mPlaces.size();
    }

    // ...
}
```

(Generar datos aleatorios)

Para este ejemplo, generaremos algunos lugares al azar.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...

    List<Place> places = randomPlaces(5);

    // ...
}

private List<Place> randomPlaces(int amount) {
    List<Place> places = new ArrayList<>();
    for (int i = 0; i < amount; i++) {
        places.add(new Place(
                BitmapFactory.decodeResource(getResources(), Math.random() > 0.5 ?
                        R.drawable.ic_account_grey600_36dp :
                        R.drawable.ic_android_grey600_36dp
                ),
                "Place #" + (int) (Math.random() * 1000)
        ));
    }
    return places;
}
```

Conecte el RecyclerView con el PlaceListAdapter y el conjunto de datos

Conectar un `RecyclerView` con un adaptador es muy fácil. `LinearLayoutManager` configurar `LinearLayoutManager` como administrador de diseño para lograr el diseño de la lista.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...

    RecyclerView recyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
    recyclerView.setAdapter(new PlaceListAdapter(places));
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
}
```

¡Hecho!

StaggeredGridLayoutManager

1. Crea tu RecyclerView en tu archivo xml de diseño:

```
<android.support.v7.widget.RecyclerView
            android:id="@+id/recycleView"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
```

2. Crea tu clase de modelo para mantener tus datos:

```
     public class PintrestItem {
     String url;
     public PintrestItem(String url,String name){
         this.url=url;
         this.name=name;
     }
     public String getUrl() {
         return url;
     }

    public String getName(){
        return name;
    }
     String name;
 }
```

3. Cree un archivo de diseño para contener los elementos RecyclerView:

```
<ImageView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:adjustViewBounds="true"
    android:scaleType="centerCrop"
    android:id="@+id/imageView"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
```

```
        android:id="@+id/name"
        android:layout_gravity="center"
        android:textColor="@android:color/white"/>
```

4. Cree la clase de adaptador para RecyclerView:

```
 public class PintrestAdapter extends
RecyclerView.Adapter<PintrestAdapter.PintrestViewHolder>{
    private ArrayList<PintrestItem>images;
    Picasso picasso;
    Context context;
    public PintrestAdapter(ArrayList<PintrestItem>images,Context context){
        this.images=images;
        picasso=Picasso.with(context);
        this.context=context;

    }

    @Override
    public PintrestViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view=
LayoutInflater.from(parent.getContext()).inflate(R.layout.pintrest_layout_item,parent,false);

        return new PintrestViewHolder(view);
    }

    @Override
    public void onBindViewHolder(PintrestViewHolder holder, int position) {
          picasso.load(images.get(position).getUrl()).into(holder.imageView);
          holder.tv.setText(images.get(position).getName());
    }

    @Override
    public int getItemCount() {
        return images.size();
    }

    public class PintrestViewHolder extends RecyclerView.ViewHolder{
        ImageView imageView;
        TextView tv;
        public PintrestViewHolder(View itemView) {
            super(itemView);
            imageView=(ImageView)itemView.findViewById(R.id.imageView);
            tv=(TextView)itemView.findViewById(R.id.name);

        }
    }
}
```

5. Cree una instancia de RecyclerView en su actividad o fragmento:

```
RecyclerView recyclerView = (RecyclerView)findViewById(R.id.recyclerView);
//Create the instance of StaggeredGridLayoutManager with 2 rows i.e the span count and
provide the orientation
StaggeredGridLayoutManager layoutManager=new new StaggeredGridLayoutManager(2,
StaggeredGridLayoutManager.VERTICAL);
recyclerView.setLayoutManager(layoutManager);
// Create Dummy Data and Add to your List<PintrestItem>
```

```
List<PintrestItem>items=new ArrayList<PintrestItem>
items.add(new PintrestItem("url of image you want to show","imagename"));
items.add(new PintrestItem("url of image you want to show","imagename"));
items.add(new PintrestItem("url of image you want to show","imagename"));
recyclerView.setAdapter(new PintrestAdapter(items,getContext() );
```

No olvide agregar la dependencia de Picasso en su archivo build.gradle:

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

Lea RecyclerView y LayoutManagers en línea:
https://riptutorial.com/es/android/topic/6772/recyclerview-y-layoutmanagers

Capítulo 211: Registro y uso de Logcat

Sintaxis

- `Log.v(String tag, String msg, Throwable tr)`
- `Log.v(String tag, String msg)`
- `Log.d(String tag, String msg, Throwable tr)`
- `Log.d(String tag, String msg)`
- `Log.i(String tag, String msg, Throwable tr)`
- `Log.i(String tag, String msg)`
- `Log.w(String tag, String msg, Throwable tr)`
- `Log.w(String tag, String msg)`
- `Log.e(String tag, String msg, Throwable tr)`
- `Log.e(String tag, String msg)`

Parámetros

Opción	Descripción
-b (buffer)	Carga un búfer de registro alternativo para ver, como eventos o radio. El búfer principal se utiliza por defecto. Consulte Visualización de búferes de registro alternativos.
-do	Borra (vacía) todo el registro y sale.
-re	Vuelca el registro en la pantalla y sale.
-f (nombre de archivo)	Escribe la salida del mensaje de registro en (nombre de archivo). El valor predeterminado es stdout.
-sol	Imprime el tamaño del búfer de registro especificado y sale.
-n (contar)	Establece el número máximo de registros rotados a (contar). El valor predeterminado es 4. Requiere la opción -r.
-r (kbytes)	Gira el archivo de registro cada (kbytes) de salida. El valor predeterminado es 16. Requiere la opción -f.
-s	Establece la especificación de filtro por defecto en silencio.
-v (formato)	Establece el formato de salida para los mensajes de registro. El valor predeterminado es formato breve.

Observaciones

Definición

Logcat es una herramienta de línea de comandos que descarga un registro de los mensajes del sistema, incluidos los seguimientos de la pila cuando el dispositivo emite un error y los mensajes que ha escrito desde su aplicación con la clase de registro .

Cuándo usar

Si está considerando usar los métodos System.out de Java para imprimir en la consola en lugar de usar uno de los métodos de registro de Android, entonces debe saber que básicamente funcionan de la misma manera. Sin embargo, es mejor evitar el uso de los métodos de Java porque la información adicional y el formato proporcionado por los métodos de registro de Android son más beneficiosos. Además, los métodos de impresión System.out se redirigen al método `Log.i()` .

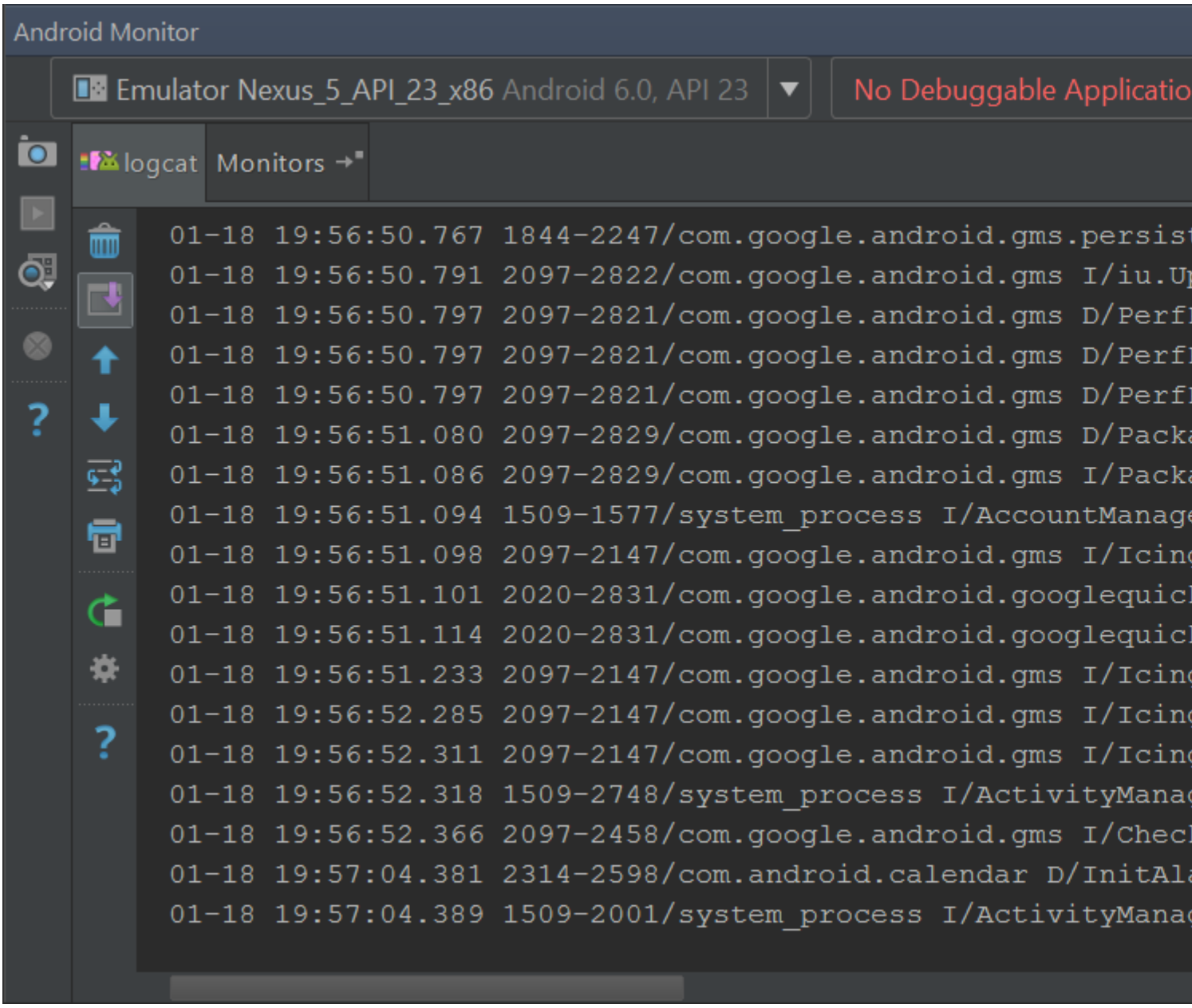
Enlaces útiles

- Documentación oficial del desarrollador de Android para Log y logcat .
- Pregunta de Stackoveflow : Android Log.v (), Log.d (), Log.i (), Log.w (), Log.e () - ¿Cuándo usar cada uno?

Examples

Filtrado de la salida logcat

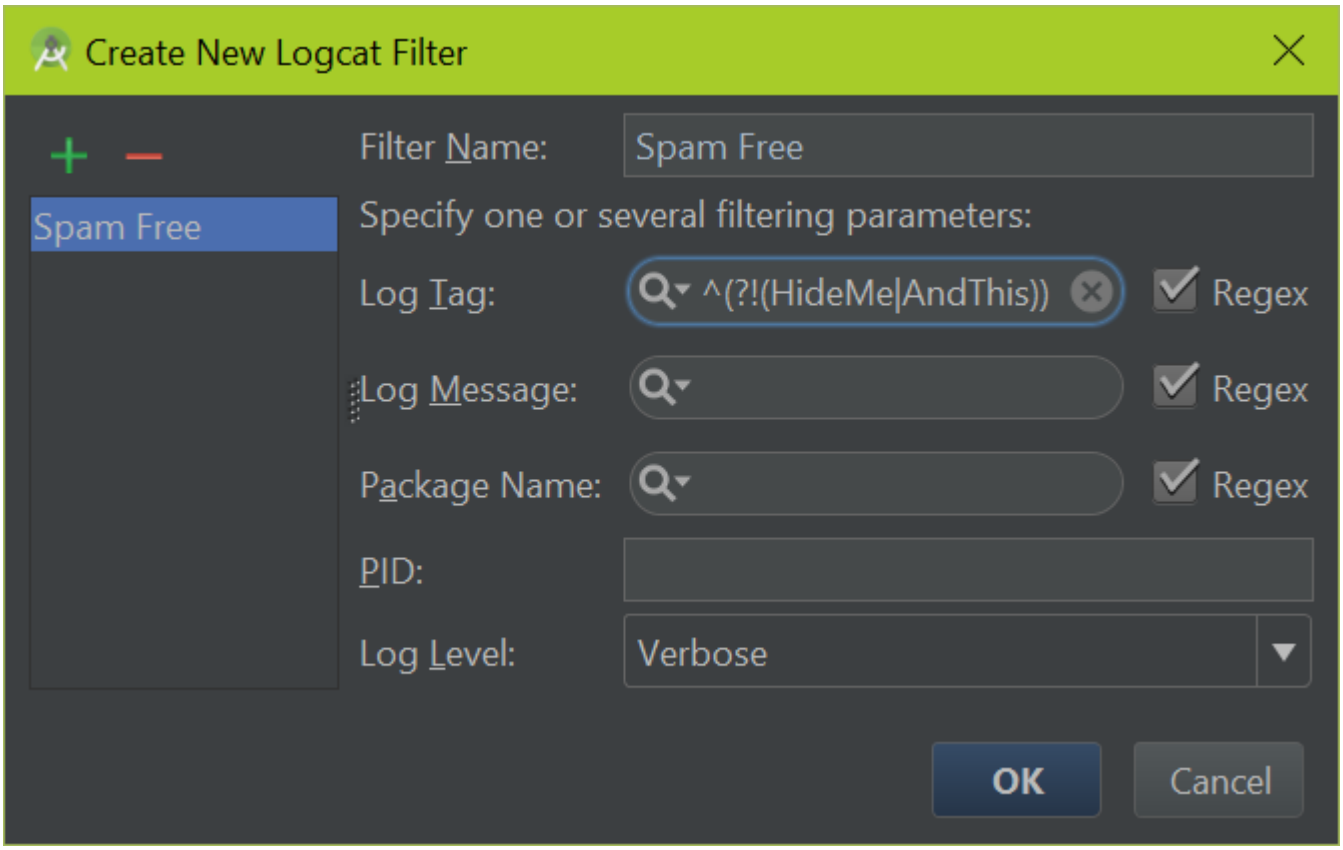
Es útil filtrar la salida logcat porque hay muchos mensajes que no son de interés. Para filtrar la salida, abra el "Monitor de Android" y haga clic en el menú desplegable en la parte superior derecha y seleccione *Editar configuración de filtro*



Ahora puede agregar filtros personalizados para mostrar los mensajes de interés, así como también filtrar las líneas de registro conocidas que pueden ignorarse de forma segura. Para ignorar una parte de la salida, puede definir una expresión regular . Aquí hay un ejemplo de exclusión de etiquetas coincidentes:

```
^(?!(HideMe|AndThis))
```

Esto se puede introducir siguiendo este ejemplo:



Lo anterior es una expresión regular que excluye entradas. Si desea agregar otra etiqueta a la *lista negra* , agréguela después de una tubería `|` personaje. Por ejemplo, si quisiera poner en la lista negra "GC", usaría un filtro como este:

```
^(?!(HideMe|AndThis|GC))
```

Para más documentación y ejemplos, visite Logging y use Logcat

Explotación florestal

Cualquier aplicación de Android de calidad hará un seguimiento de lo que está haciendo a través de los registros de aplicaciones. Estos registros permiten una fácil depuración de la ayuda para que el desarrollador pueda diagnosticar qué está sucediendo con la aplicación. La documentación completa de Android se puede encontrar aquí , pero a continuación se presenta un resumen:

Registro basico

La clase de `Log` es la fuente principal de escritura de registros de desarrollador, especificando una `tag` y un `message` . La etiqueta es lo que puede usar para filtrar los mensajes de registro para identificar qué líneas provienen de su Actividad en particular. Simplemente llama

```
Log.v(String tag, String msg);
```

Y el sistema Android escribirá un mensaje al logcat:

```
07-28 12:00:00.759 24812-24839/my.packagename V/MyAnimator: Some log messages
 └ time stamp             |  app.package┘     |    └ any tag  |
    process & thread ids ┘          log level┘               └ the log message
```

> PROPINA:
> Observe la identificación del proceso y la identificación del hilo. Si son iguales, ¡el registro proviene del hilo principal / UI!

Se puede usar cualquier etiqueta, pero es común usar el nombre de la clase como una etiqueta:

```
public static final String tag = MyAnimator.class.getSimpleName();
```

Niveles de registro

El registrador de Android tiene 6 niveles diferentes, cada uno de los cuales cumple un determinado propósito:

- `ERROR` : `Log.e()`
  - Utilizado para indicar una falla crítica, este es el nivel impreso al lanzar una `Exception` .
- `WARN` : `Log.w()`
  - Se usa para indicar una advertencia, principalmente para fallas recuperables
- `INFO` : `Log.i()`
  - Se utiliza para indicar información de nivel superior sobre el estado de la aplicación
- `DEBUG` : `Log.d()`
  - Se utiliza para registrar información que sería útil saber al depurar la aplicación, pero se interpondría al ejecutar la aplicación.
- `VERBOSE` : `Log.v()`
  - Se utiliza para registrar información que refleja los pequeños detalles sobre el estado de la aplicación
- `ASSERT` : `Log.wtf()`
  - Se utiliza para registrar información sobre una condición que nunca debería ocurrir.
  - *wtf* significa "What a Terrible Failure".

Motivación para la tala

La motivación para el registro es encontrar fácilmente errores, advertencias y otra información echando un vistazo a la cadena de eventos de la aplicación. Por ejemplo, imagine una aplicación que lee líneas de un archivo de texto, pero supone incorrectamente que el archivo nunca estará vacío. La traza de registro (de una aplicación que no registra) se vería así:

```
E/MyApplication: Process: com.example.myapplication, PID: 25788
                         com.example.SomeRandomException: Expected string, got 'null' instead
```

Seguido de un montón de rastros de pila que eventualmente conducirían a la línea ofensiva, donde el paso con un depurador eventualmente conduciría al problema

Sin embargo, la traza de registro de una aplicación con el registro habilitado podría tener este aspecto:

```
V/MyApplication: Looking for file myFile.txt on the SD card
D/MyApplication: Found file myFile.txt at path <path>
V/MyApplication: Opening file myFile.txt
D/MyApplication: Finished reading myFile.txt, found 0 lines
V/MyApplication: Closing file myFile.txt
...
E/MyApplication: Process: com.example.myapplication, PID: 25788
                         com.example.SomeRandomException: Expected string, got 'null' instead
```

Un vistazo rápido a los registros y es obvio que el archivo estaba vacío.

Cosas a tener en cuenta al iniciar sesión:

Si bien el registro es una herramienta poderosa que permite a los desarrolladores de Android obtener una mejor visión del funcionamiento interno de su aplicación, el registro tiene algunos inconvenientes.

Legibilidad del registro:

Es común que las aplicaciones de Android tengan varios registros que se ejecutan de forma síncrona. Como tal, es muy importante que cada registro sea fácil de leer y solo contenga información relevante y necesaria.

Actuación:

El registro requiere una pequeña cantidad de recursos del sistema. En general, esto no es motivo de preocupación, sin embargo, si se utiliza en exceso, el registro puede tener un impacto negativo en el rendimiento de la aplicación.

Seguridad:

Recientemente, se han agregado varias aplicaciones de Android al mercado de Google Play que permiten al usuario ver los registros de todas las aplicaciones en ejecución. Esta visualización no intencionada de datos puede permitir a los usuarios ver información confidencial. Como regla general, siempre elimine los registros que contienen datos no públicos *antes de* publicar su aplicación en el mercado.

Conclusión:

El registro es una parte esencial de una aplicación de Android, debido a la potencia que otorga a los desarrolladores. La capacidad de crear un registro de seguimiento útil es uno de los aspectos más desafiantes del desarrollo de software, pero la clase de registro de Android ayuda a hacerlo

mucho más fácil.

Para más documentación y ejemplos, visite Logging y use Logcat

Iniciar sesión con un enlace a la fuente directamente desde Logcat

Este es un buen truco para agregar un enlace al código, por lo que será fácil saltar al código que emitió el registro.

Con el siguiente código, esta llamada:

```
MyLogger.logWithLink("MyTag","param="+param);
```

Resultará en:

```
07-26...012/com.myapp D/MyTag: MyFrag:onStart(param=3)  (MyFrag.java:2366) // << logcat
converts this to a link to source!
```

Este es el código (dentro de una clase llamada MyLogger):

```
static StringBuilder sb0 = new StringBuilder(); // reusable string object

public static void logWithLink(String TAG, Object param) {
    StackTraceElement stack = Thread.currentThread().getStackTrace()[3];
    sb0.setLength(0);
    String c = stack.getFileName().substring(0, stack.getFileName().length() - 5); // removes
the ".java"
    sb0.append(c).append(":");
    sb0.append(stack.getMethodName()).append('(');
    if (param != null) {
        sb0.append(param);
    }
    sb0.append(") ");
    sb0.append("
(").append(stack.getFileName()).append(':').append(stack.getLineNumber()).append(')');
    Log.d(TAG, sb0.toString());
}
```

Este es un ejemplo básico, se puede extender fácilmente para emitir un enlace a la persona que llama (sugerencia: la pila será [4] en lugar de [3]), y también puede agregar otra información relevante.

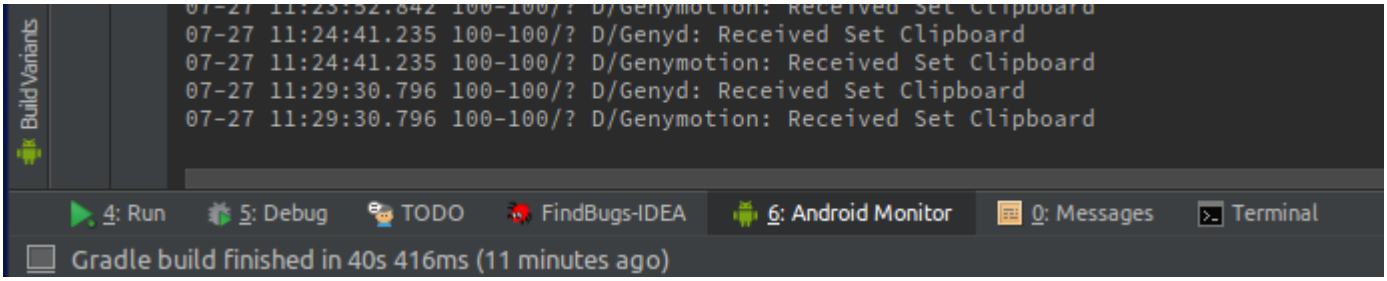
Usando el Logcat

Logcat es una herramienta de línea de comandos que descarga un registro de los mensajes del sistema, incluidos los seguimientos de la pila cuando el dispositivo emite un error y los mensajes que ha escrito desde su aplicación con la clase de registro.

La salida de Logcat se puede mostrar en el Android Monitor de Android Studio o con la línea de comandos adb.

En Android Studio

Mostrar haciendo clic en el icono "Android Monitor":



O presionando `Alt` + `6` en Windows / Linux o `CMD` + `6` en Mac.

a través de la línea de comando:

Uso simple:

```
$ adb logcat
```

Con marcas de tiempo:

```
$ adb logcat -v time
```

Filtrar por texto específico:

```
$ adb logcat -v time | grep 'searchtext'
```

Hay muchas opciones y filtros disponibles para la *línea de comandos logcat* , documentados aquí .
Un ejemplo simple pero útil es la siguiente expresión de filtro que muestra todos los mensajes de registro con "error" de nivel de prioridad, en todas las etiquetas:

```
$ adb logcat *:E
```

Generando código de registro

`Live templates Android Studio` pueden ofrecer varios accesos directos para un registro rápido. Para usar las plantillas de Live, todo lo que necesita hacer es comenzar a escribir el nombre de la plantilla y presionar `TAB` o ingresar para insertar la declaración.

Ejemplos:

- `logi` → se convierte en → `android.util.Log.i(TAG, "$METHOD_NAME$: $content$");`
  - `$METHOD_NAME$` se reemplazará automáticamente con el nombre de su método, y el cursor esperará a que se llene el contenido.
- `loge` → igual, por error
- etc. para el resto de los niveles de registro.

La lista completa de plantillas se puede encontrar en la configuración de `Android Studio` (`ALT` + `sy`

tipo "en vivo"). Y también es posible agregar sus plantillas personalizadas.

Si encuentra que `Android Studio Live templates` `Android Studio Live templates` no son suficientes para sus necesidades, puede considerar Android Postfix Plugin

Esta es una biblioteca muy útil que le ayuda a evitar escribir la línea de registro manualmente.

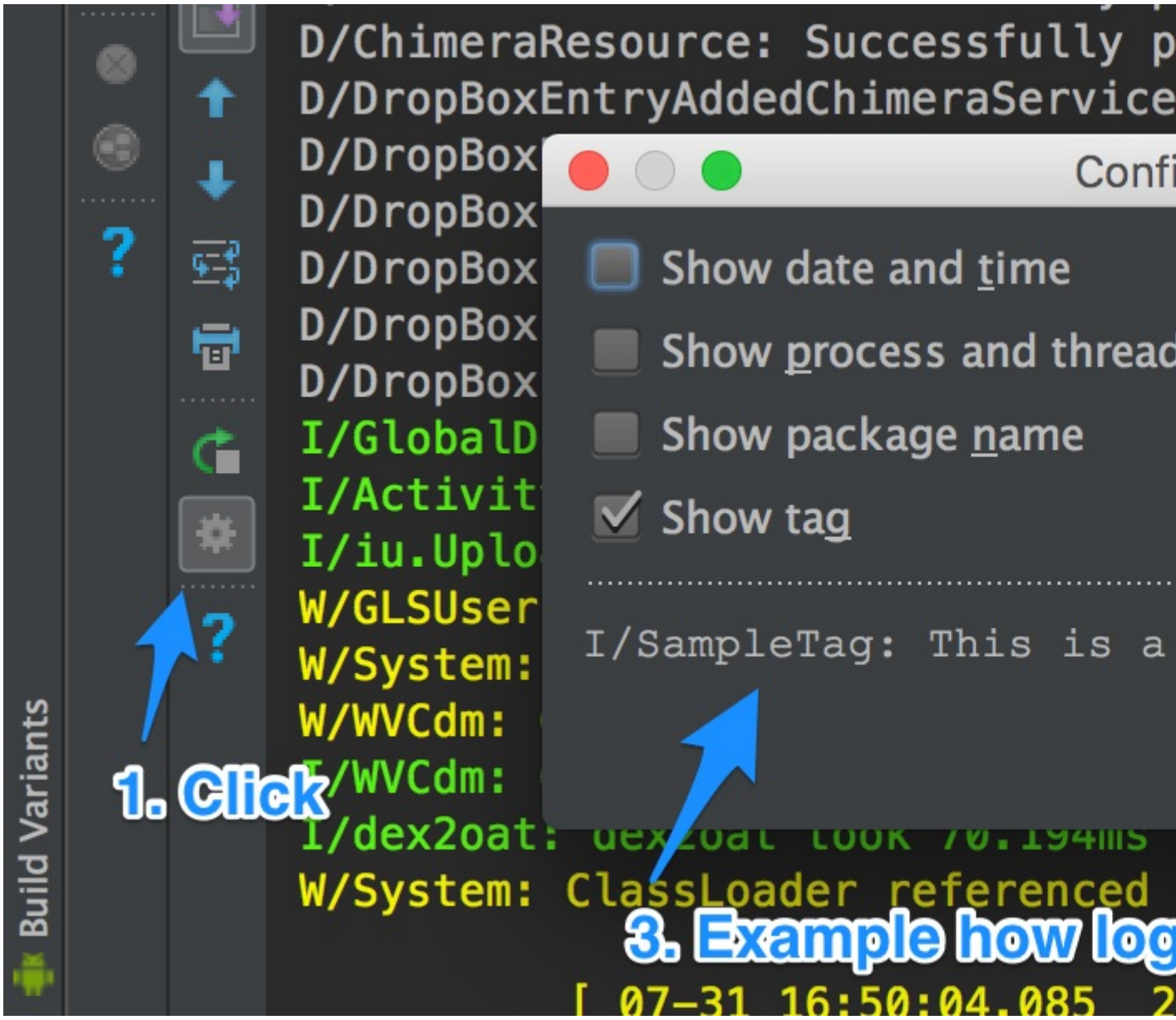
La sintaxis es absolutamente simple:

.log - Logging. Si hay una variable constante "TAG", use "TAG". Si no, use el nombre de la clase.

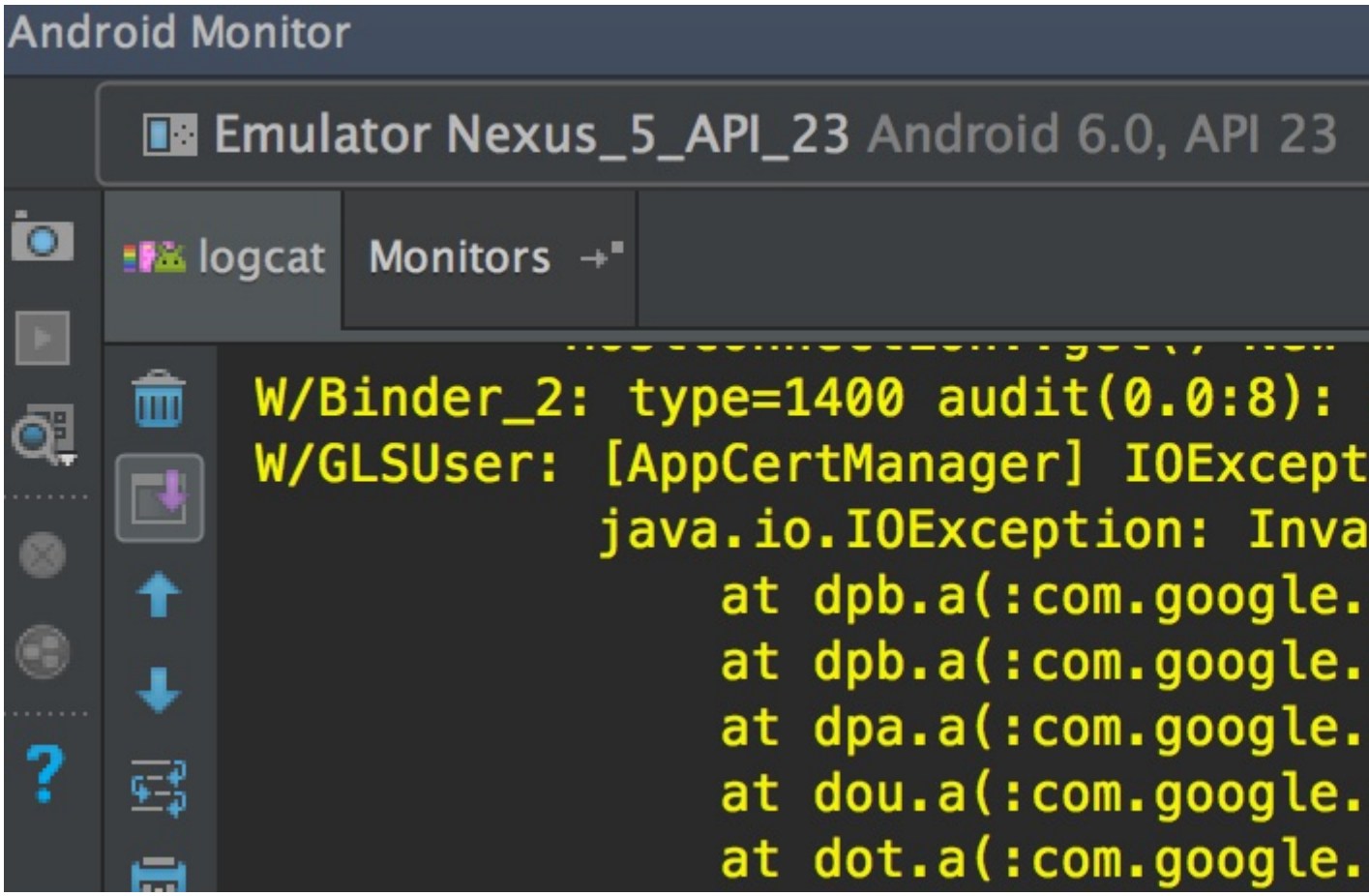
```
ublic class MyActivity extends Activity {
  static final String T= "test";
  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      new View.OnClickListener() {
          @Override
          public void onClick(View view) {

          }
      };
  }
```

Uso de Android Studio

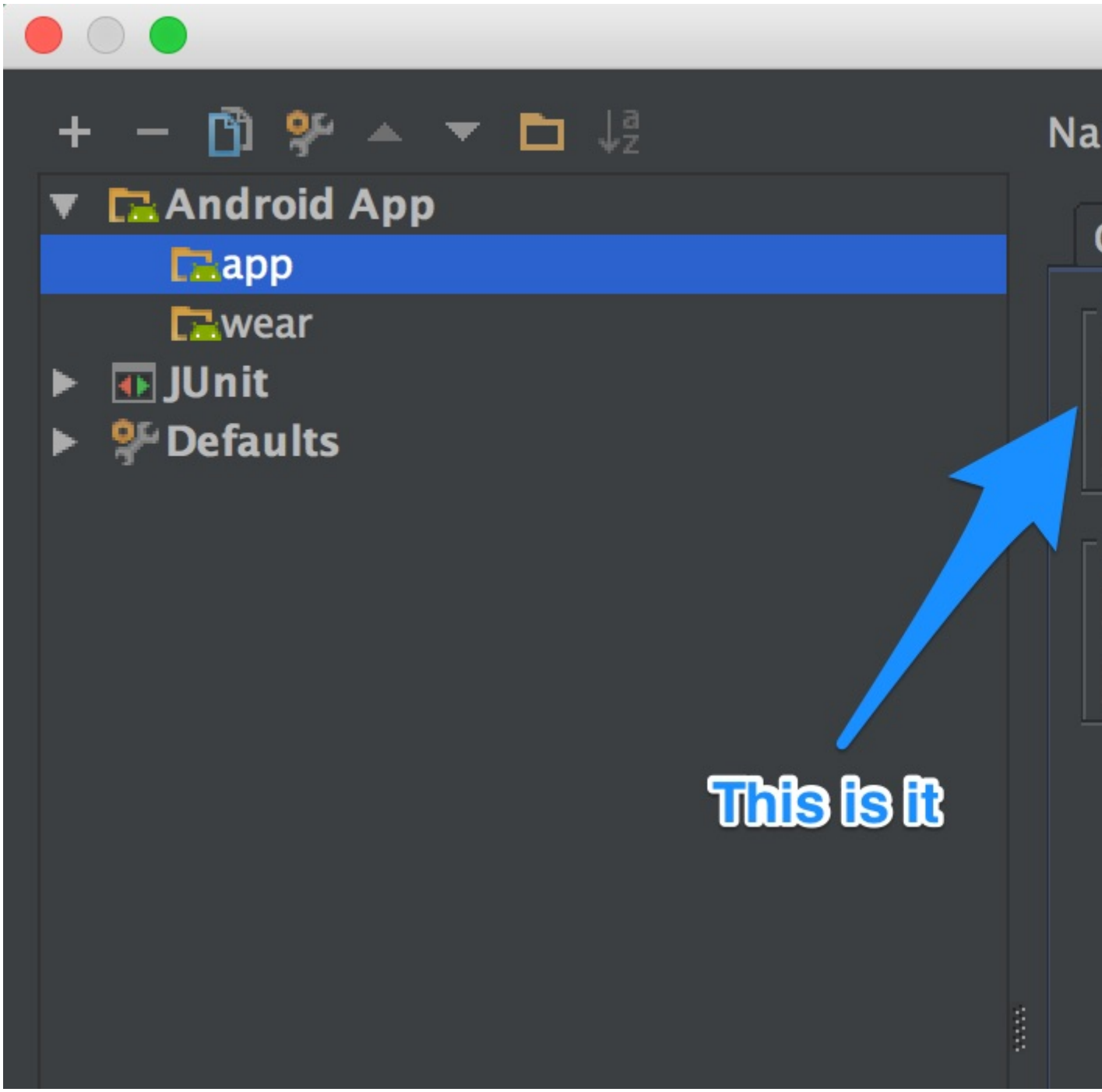
1. Ocultar / mostrar información impresa:



2. Control de verbosidad del registro:



3. Deshabilitar / habilitar la ventana de registro de apertura al iniciar la aplicación ejecutar / depurar



Borrar registros

Para borrar (vaciar) el registro completo:

```
adb logcat -c
```

Lea Registro y uso de Logcat en línea: https://riptutorial.com/es/android/topic/1552/registro-y-uso-de-logcat

Capítulo 212: Reino

Introducción

Realm Mobile Database es una alternativa a SQLite. Realm Mobile Database es mucho más rápido que un ORM y, a menudo, más rápido que SQLite sin procesar.

Beneficios

Funcionalidad fuera de línea, consultas rápidas, subprocesos seguros, aplicaciones multiplataforma, cifrado, arquitectura reactiva.

Observaciones

Cuando use Realm, debe recordar que no debe pasar las instancias RealmObjects, RealmResults y Realm entre los subprocesos. Si necesita una consulta en un hilo determinado, abra una instancia de Realm en ese hilo. Al terminar el hilo, debes cerrar el Reino.

> ***NOTA LEGAL*** : Usted comprende que el Software puede contener funciones criptográficas que pueden estar sujetas a restricciones a la exportación, y usted declara y garantiza que **no se encuentra en un país** sujeto a restricciones o embargo de exportaciones de Estados Unidos, **incluidos Cuba, Irán, Norte. Corea, Sudán, Siria o la región de Crimea** , y que usted no está en la lista del Departamento de Comercio de Personas Negadas, Partes no verificadas, o afiliado a una Entidad Restringida.

Examples

Agregando Realm a tu proyecto

Agregue la siguiente dependencia a su archivo `build.gradle` nivel de **proyecto** .

```
dependencies {
    classpath "io.realm:realm-gradle-plugin:3.1.2"
}
```

Agregue el siguiente derecho en la parte superior del archivo `build.gradle` de su nivel de **aplicación** .

```
apply plugin: 'realm-android'
```

¡Complete una sincronización de Gradle y ahora tiene Realm agregado como una dependencia para su proyecto!

Realm requiere una llamada inicial desde 2.0.0 antes de usarla. Puede hacer esto en su clase de

`Application` o en el método `onCreate` su primera Actividad.

```
Realm.init(this); // added in Realm 2.0.0
Realm.setDefaultConfiguration(new RealmConfiguration.Builder().build());
```

Modelos de reino

Los modelos de reino deben extender la clase base `RealmObject` , definen el esquema de la base de datos subyacente.

Los tipos de campo admitidos son `boolean` , `byte` , `short` , `int` , `long` , `float` , `double` , `String` , `Date` , `byte[]` , enlaces a otros `RealmObject` s, y `RealmList<T extends RealmModel>` .

```
public class Person extends RealmObject {
    @PrimaryKey //primary key is also implicitly an @Index
                //it is required for `copyToRealmOrUpdate()` to update the object.
    private long id;

    @Index //index makes queries faster on this field
    @Required //prevents `null` value from being inserted
    private String name;

    private RealmList<Dog> dogs; //->many relationship to Dog

    private Person spouse; //->one relationship to Person

    @Ignore
    private Calendar birthday; //calendars are not supported but can be ignored

    // getters, setters
}
```

Si agrega (o elimina) un nuevo campo a su RealmObject (o agrega una nueva clase de RealmObject o elimina una existente), se necesitará una **migración** . Puede establecer `deleteIfMigrationNeeded()` en su `RealmConfiguration.Builder` , o definir la migración necesaria. La migración también es necesaria al agregar (o eliminar) `@Required` , o `@Index` , o `@PrimaryKey` anotación.

Las relaciones deben establecerse manualmente, NO son automáticas basadas en claves primarias.

Desde 0.88.0, también es posible usar campos públicos en lugar de campos / getters / setters privados en las clases de RealmObject.

También es posible implementar `RealmModel` lugar de extender `RealmObject` , si la clase también se anota con `@RealmClass` .

```
@RealmClass
public class Person implements RealmModel {
    // ...
}
```

En ese caso, los métodos como `person.deleteFromRealm()` o `person.addChangeListener()` se reemplazan con `RealmObject.deleteFromRealm(person)` y `RealmObject.addChangeListener(person)` .

Las limitaciones son que, mediante un objeto `RealmObject` , solo se puede extender `RealmObject` , y no hay soporte para `transient` campos `final` , `volatile` y `transient` .

Es importante que una clase *administrada de* RealmObject solo pueda modificarse en una transacción. Un RealmObject *administrado* no se puede pasar entre hilos.

Lista de primitivas (RealmList)

El reino actualmente no admite el almacenamiento de una lista de primitivas. Está en su lista de tareas pendientes (problema GitHub # 575), pero mientras tanto, aquí hay una solución.

Cree una nueva clase para su tipo primitivo, esto usa Integer, pero cámbielo por lo que quiera almacenar.

```
public class RealmInteger extends RealmObject {
    private int val;

    public RealmInteger() {
    }

    public RealmInteger(int val) {
        this.val = val;
    }

    // Getters and setters
}
```

Ahora puedes usar esto en tu `RealmObject` .

```
public class MainObject extends RealmObject {

    private String name;
    private RealmList<RealmInteger> ints;

    // Getters and setters
}
```

Si está utilizando `GSON` para completar su `RealmObject` , deberá agregar un adaptador de tipo personalizado.

```
Type token = new TypeToken<RealmList<RealmInteger>>(){}.getType();
Gson gson = new GsonBuilder()
        .setExclusionStrategies(new ExclusionStrategy() {
            @Override
            public boolean shouldSkipField(FieldAttributes f) {
                return f.getDeclaringClass().equals(RealmObject.class);
            }

            @Override
            public boolean shouldSkipClass(Class<?> clazz) {
                return false;
```

```
            }
        })
        .registerTypeAdapter(token, new TypeAdapter<RealmList<RealmInteger>>() {

            @Override
            public void write(JsonWriter out, RealmList<RealmInteger> value) throws
IOException {
                // Empty
            }

            @Override
            public RealmList<RealmInteger> read(JsonReader in) throws IOException {
                RealmList<RealmInteger> list = new RealmList<RealmInteger>();
                in.beginArray();
                while (in.hasNext()) {
                    list.add(new RealmInteger(in.nextInt()));
                }
                in.endArray();
                return list;
            }
        })
        .create();
```

probar con recursos

```
try (Realm realm = Realm.getDefaultInstance()) {
    realm.executeTransaction(new Realm.Transaction() {
            @Override
            public void execute(Realm realm) {
                //whatever Transaction that has to be done
            }
    });
    //No need to close realm in try-with-resources
}
```

El intento con recursos solo se puede usar desde KITKAT (minSDK 19)

Consultas ordenadas

Para ordenar una consulta, en lugar de usar `findAll()` , debe usar `findAllSorted()` .

```
RealmResults<SomeObject> results = realm.where(SomeObject.class)
                                        .findAllSorted("sortField", Sort.ASCENDING);
```

Nota:

`sort()` devuelve un RealmResults completamente nuevo que está ordenado, pero una actualización de este RealmResults lo restablecerá. Si usa `sort()` , siempre debe reorganizarlo en su `RealmChangeListener` , eliminar el `RealmChangeListener` de los `RealmResults` anteriores y agregarlo a los nuevos `RealmResults` . El uso de `sort()` en un `RealmResults` devuelto por una consulta asíncrona que aún no se ha cargado fallará.

`findAllSorted()` siempre devolverá los resultados ordenados por campo, incluso si se actualiza. Se recomienda utilizar `findAllSorted()` .

Consultas asincrónicas

Cada método de consulta síncrona (como `findAll()` o `findAllSorted()`) tiene una contraparte asíncrona (`findAllAsync()` / `findAllSortedAsync()`).

Las consultas asíncronas descargan la evaluación de `RealmResults` a otro hilo. Para recibir estos resultados en el subproceso actual, el subproceso actual debe ser un subproceso looper (lea: las consultas asíncronas generalmente solo funcionan en el subproceso de la interfaz de usuario).

```
RealmChangeListener<RealmResults<SomeObject>> realmChangeListener; // field variable

realmChangeListener = new RealmChangeListener<RealmResults<SomeObject>>() {
    @Override
    public void onChange(RealmResults<SomeObject> element) {
        // asyncResults are now loaded
        adapter.updateData(element);
    }
};

RealmResults<SomeObject> asyncResults = realm.where(SomeObject.class).findAllAsync();
asyncResults.addChangeListener(realmChangeListener);
```

Usando Realm con RxJava

Para consultas, Realm proporciona el método `realmResults.asObservable()` . La observación de resultados solo es posible en los subprocesos del looper (normalmente el subproceso de la interfaz de usuario).

Para que esto funcione, su configuración debe contener lo siguiente

```
realmConfiguration = new RealmConfiguration.Builder(context)       //
                         .rxFactory(new RealmObservableFactory()) //
                            //...
                         .build();
```

Después, puede utilizar sus resultados como un observable.

```
Observable<RealmResults<SomeObject>> observable = results.asObservable();
```

Para consultas asíncronas, debe filtrar los resultados por `isLoaded()` , de modo que reciba un evento solo cuando la consulta se haya ejecutado. Este `filter()` no es necesario para consultas síncronas (`isLoaded()` siempre devuelve `true` en consultas de sincronización).

```
    Subscription subscription = RxTextView.textChanges(editText).switchMap(charSequence ->
        realm.where(SomeObject.class)
             .contains("searchField", charSequence.toString(), Case.INSENSITIVE)
             .findAllAsync()
             .asObservable())
    .filter(RealmResults::isLoaded) //
    .subscribe(objects -> adapter.updateData(objects));
```

Para las escrituras, debe usar el método `executeTransactionAsync()` o abrir una instancia de Realm en el subproceso en segundo plano, ejecutar la transacción de forma sincrónica y luego cerrar la instancia de Realm.

```
public Subscription loadObjectsFromNetwork() {
    return objectApi.getObjects()
        .subscribeOn(Schedulers.io())
        .subscribe(response -> {
            try(Realm realmInstance = Realm.getDefaultInstance()) {
                realmInstance.executeTransaction(realm ->
realm.insertOrUpdate(response.objects));
            }
        });
}
```

Uso básico

Configurando una instancia

Para usar Realm primero necesitas obtener una instancia de él. Cada instancia de Realm se asigna a un archivo en el disco. La forma más básica de obtener una instancia es la siguiente:

```
// Create configuration
RealmConfiguration realmConfiguration = new RealmConfiguration.Builder(context).build();

// Obtain realm instance
Realm realm = Realm.getInstance(realmConfiguration);
// or
Realm.setDefaultConfiguration(realmConfiguration);
Realm realm = Realm.getDefaultInstance();
```

El método `Realm.getInstance()` crea el archivo de base de datos si no se ha creado; de lo contrario, abre el archivo. El objeto `RealmConfiguration` controla todos los aspectos de cómo se crea un Reino, ya sea una base de datos en `inMemory()` , el nombre del archivo del Reino, si se debe borrar el Reino si se necesita una migración, datos iniciales, etc.

Tenga en cuenta que las llamadas a `Realm.getInstance()` se cuentan como referencia (cada llamada incrementa un contador), y el contador disminuye cuando se llama a `realm.close()` .

Cerrando una instancia

En los subprocesos en segundo plano, es *muy importante* ***cerrar*** las instancias de Realm una vez que ya no se usan (por ejemplo, la transacción está completa y la ejecución del subproceso finaliza). Si no se cierran todas las instancias de Realm en el subproceso en segundo plano, la versión se anclará y esto puede causar un gran aumento en el tamaño del archivo.

```
Runnable runnable = new Runnable() {
    Realm realm = null;
```

```
    try {
        realm = Realm.getDefaultInstance();
        // ...
    } finally {
        if(realm != null) {
            realm.close();
        }
    }
};

new Thread(runnable).start(); // background thread, like `doInBackground()` of AsyncTask
```

Vale la pena señalar que, por encima del nivel de API 19, puede reemplazar este código con solo esto:

```
try(Realm realm = Realm.getDefaultInstance()) {
    // ...
}
```

Modelos

El siguiente paso sería crear tus modelos. Aquí se puede hacer una pregunta, "¿qué es un modelo?". Un modelo es una estructura que define las propiedades de un objeto que se almacena en la base de datos. Por ejemplo, en lo siguiente modelamos un libro.

```
public class Book extends RealmObject {

    // Primary key of this entity
    @PrimaryKey
    private long id;

    private String title;

    @Index // faster queries
    private String author;

    // Standard getters & setter
    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
```

```
    }

    public void setAuthor(String author) {
        this.author = author;
    }
}
```

Tenga en cuenta que sus modelos deben extender la clase RealmObject. La clave principal también se especifica mediante la anotación `@PrimaryKey` . Las claves principales pueden ser nulas, pero solo un elemento puede tener un `null` como clave principal. También puede usar la anotación `@Ignore` para los campos que no deben persistir en el disco:

```
@Ignore
private String isbn;
```

Inserción o actualización de datos.

Para almacenar un objeto de libro en su instancia de la base de datos Realm, primero puede crear una instancia de su modelo y luego almacenarla en la base de datos a través del método `copyToRealm` . Para crear o actualizar puede usar `copyToRealmOrUpdate` . (Una alternativa más rápida es el añadido `insertOrUpdate()`).

```
// Creating an instance of the model
Book book = new Book();
book.setId(1);
book.setTitle("Walking on air");
book.setAuthor("Taylor Swift")

// Store to the database
realm.executeTransaction(new Realm.Transaction() {
    @Override
    public void execute(Realm realm) {
        realm.insertOrUpdate(book);
    }
});
```

Tenga en cuenta que todos los cambios en los datos deben ocurrir en una transacción. Otra forma de crear un objeto es mediante el siguiente patrón:

```
Book book = realm.createObject(Book.class, primaryKey);
...
```

Consultar la base de datos

- Todos los libros

  ```
  RealmResults<Book> results = realm.where(Book.class).findAll();
  ```

- Todos los libros que tengan una identificación superior a 10:

```
RealmResults<Book> results = realm.where(Book.class)
                                  .greaterThan("id", 10)
                                  .findAll();
```

- Libros de `'Taylor Swift'` o `'%Peter%'` :

```
RealmResults<Book> results = realm.where(Book.class)
                                  .beginGroup()
                                      .equalTo("author", "Taylor Swift")
                                      .or()
                                      .contains("author", "Peter")
                                  .endGroup().findAll();
```

Borrando un objeto

Por ejemplo, queremos eliminar todos los libros de Taylor Swift:

```
// Start of transaction
realm.executeTransaction(new Realm.Transaction() {
    @Override
    public void execute(Realm realm) {
        // First Step: Query all Taylor Swift books
        RealmResults<Book> results = ...

        // Second Step: Delete elements in Realm
        results.deleteAllFromRealm();
    }
});
```

Lea Reino en línea: https://riptutorial.com/es/android/topic/3187/reino

Capítulo 213: RenderScript

Introducción

RenderScript es un lenguaje de scripting que le permite escribir renderizado gráfico de alto rendimiento y código computacional en bruto. Proporciona un medio para escribir código crítico de rendimiento que el sistema compila posteriormente en código nativo para el procesador en el que puede ejecutarse. Esto podría ser la CPU, una CPU de varios núcleos o incluso la GPU. Lo que finalmente se ejecuta depende de muchos factores que no están disponibles para el desarrollador, pero también depende de la arquitectura que admita el compilador de la plataforma interna.

Examples

Empezando

RenderScript es un marco para permitir la computación paralela de alto rendimiento en Android. Los scripts que escriba se ejecutarán en todos los procesadores disponibles (por ejemplo, CPU, GPU, etc.) en paralelo, lo que le permitirá concentrarse en la tarea que desea lograr en lugar de cómo se programa y se ejecuta.

Los scripts están escritos en un lenguaje basado en C99 (C99 es una versión antigua del estándar de lenguaje de programación C). Para cada Script se crea una clase Java que le permite interactuar fácilmente con RenderScript en su código Java.

Configurando tu proyecto

Existen dos formas diferentes de acceder a RenderScript en su aplicación, con las bibliotecas de Android Framework o la Biblioteca de soporte. Incluso si no desea apuntar a dispositivos antes del nivel de API 11, siempre debe usar la implementación de la biblioteca de soporte porque garantiza la compatibilidad de los dispositivos en muchos dispositivos diferentes. Para usar la implementación de la biblioteca de soporte, debe usar al menos herramientas de compilación versión `18.1.0` .

Ahora permite configurar el archivo build.gradle de su aplicación:

```
android {
    compileSdkVersion 24
    buildToolsVersion '24.0.1'

    defaultConfig {
        minSdkVersion 8
        targetSdkVersion 24

        renderscriptTargetApi 18
        renderscriptSupportModeEnabled true
```

```
    }
}
```

- `renderscriptTargetApi` : debe establecerse en el nivel de API más antiguo de la versión, que proporciona toda la funcionalidad de RenderScript que necesita.
- `renderscriptSupportModeEnabled` : Esto permite el uso de la implementación RenderScript de la biblioteca de soporte.

Cómo funciona RenderScript

Un RenderScript típico consta de dos cosas: Kernels y Funciones. Una función es exactamente lo que suena: acepta una entrada, hace algo con esa entrada y devuelve una salida. Un Kernel es de donde viene el verdadero poder de RenderScript.

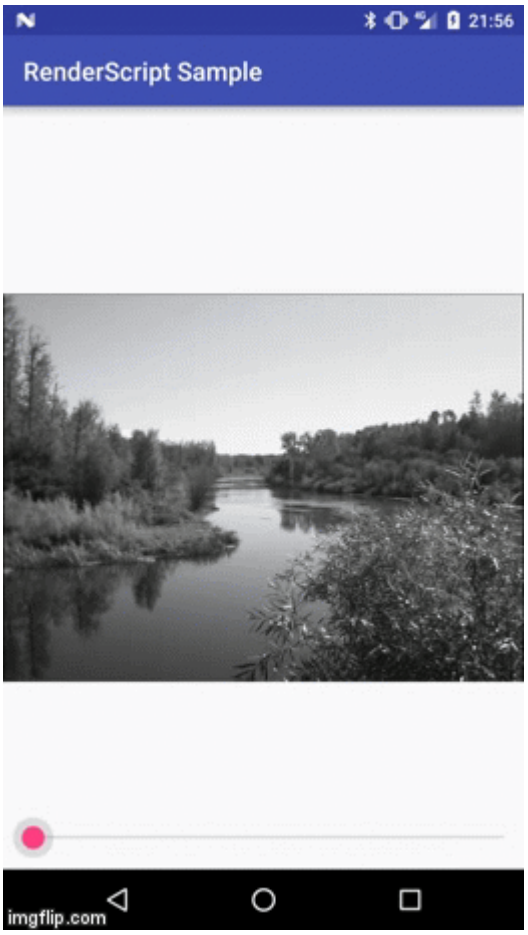
Un Kernel es una función que se ejecuta contra cada elemento dentro de una `Allocation` . Se puede usar una `Allocation` para pasar datos como un `Bitmap` o una matriz de `byte` a un `RenderScript` y también se usan para obtener un resultado de un núcleo. Los kernels pueden tomar una `Allocation` como entrada y otra como salida o pueden modificar los datos dentro de una `Allocation` .

Puede escribir sus Kernels, pero también hay muchos Kernels predefinidos que puede usar para realizar operaciones comunes como un Desenfoque de Imagen Gaussian.

Como ya se mencionó para cada archivo RenderScript, se genera una clase para interactuar con él. Estas clases siempre comienzan con el prefijo `ScriptC_` seguido del nombre del archivo RenderScript. Por ejemplo, si su archivo RenderScript se llama `example` , la clase Java generada se llamará `ScriptC_example` . Todas las secuencias de comandos predefinidas comienzan con la `Script` prefijo, por ejemplo, la secuencia de comandos de desenfoque de imagen gaussiana se llama `ScriptIntrinsicBlur` .

Escribiendo tu primer RenderScript

El siguiente ejemplo se basa en un ejemplo en GitHub. Realiza la manipulación básica de la imagen modificando la saturación de una imagen. Puede encontrar el código fuente **aquí** y verificarlo si quiere jugar con él. Aquí hay un gif rápido de cómo se verá el resultado:



Plantilla de RenderScript

Los archivos RenderScript residen en la carpeta `src/main/rs` en su proyecto. Cada archivo tiene la extensión de archivo `.rs` y debe contener dos declaraciones `#pragma` en la parte superior:

```
#pragma version(1)
#pragma rs java_package_name(your.package.name)
```

- `#pragma version(1)` : se puede usar para configurar la versión de RenderScript que está utilizando. Actualmente solo hay versión 1.
- `#pragma rs java_package_name(your.package.name)` : se puede usar para establecer el nombre del paquete de la clase Java generada para interactuar con este RenderScript en particular.

Hay otro `#pragma` que normalmente debe establecer en cada uno de sus archivos RenderScript y se usa para establecer la precisión del punto flotante. Puede establecer la precisión de punto flotante en tres niveles diferentes:

- `#pragma rs_fp_full` : esta es la configuración más estricta con la mayor precisión y también es el valor predeterminado si no se especifica nada. Debe usar esto si necesita alta precisión de punto flotante.
- `#pragma rs_fp_relaxed` : Esto no garantiza una precisión de punto flotante tan alta, pero en algunas arquitecturas permite un montón de optimizaciones que pueden hacer que sus scripts se ejecuten más rápido.

- `#pragma rs_fp_imprecise` : Esto asegura incluso menos precisión y debe usarse si la precisión de punto flotante no es realmente importante para su script.

La mayoría de los scripts solo pueden usar `#pragma rs_fp_relaxed` menos que realmente necesite una alta precisión de punto flotante.

Variables globales

Ahora, al igual que en el código C, puede definir variables globales o constantes:

```
const static float3 gMonoMult = {0.299f, 0.587f, 0.114f};

float saturationLevel = 0.0f;
```

La variable `gMonoMult` es de tipo `float3` . Esto significa que es un vector que consta de 3 números flotantes. La otra variable `float` llamada `saturationValue` no es constante, por lo tanto, puede establecerla en el tiempo de ejecución en el valor que desee. Puede usar variables como esta en sus Kernels o funciones y, por lo tanto, son otra forma de dar entrada o recibir salida de sus RenderScripts. Para cada variable no constante, se generará un método de obtención y establecimiento en la clase Java asociada.

Kernels

Pero ahora vamos a empezar a implementar el Kernel. Para los propósitos de este ejemplo, no voy a explicar las matemáticas utilizadas en el Kernel para modificar la saturación de la imagen, sino que me centraré en cómo implementar un Kernel y cómo usarlo. Al final de este capítulo, explicaré rápidamente lo que realmente está haciendo el código en este Kernel.

Kernels en general

Veamos primero el código fuente:

```
uchar4 __attribute__((kernel)) saturation(uchar4 in) {
    float4 f4 = rsUnpackColor8888(in);
    float3 dotVector = dot(f4.rgb, gMonoMult);
    float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
    return rsPackColorTo8888(newColor);
}
```

Como puede ver, parece una función normal de C con una excepción: el `__attribute__((kernel))` entre el tipo de retorno y el nombre del método. Esto es lo que le dice a RenderScript que este método es un Kernel. Otra cosa que podría notar es que este método acepta un parámetro `uchar4` y devuelve otro valor `uchar4` . `uchar4` es, como la variable `float3` que analizamos en el capítulo anterior, un vector. Contiene 4 `uchar` valores que son solo valores de bytes en el rango de 0 a 255.

Puede acceder a estos valores individuales de muchas maneras diferentes, por ejemplo, `in.r` devolverá el byte que corresponde al canal rojo de un píxel. Usamos un `uchar4` ya que cada píxel se compone de 4 valores: `r` para rojo, `g` para verde, `b` para azul y `a` para alfa, y puedes acceder a

ellos con esta taquigrafía. RenderScript también le permite tomar cualquier número de valores de un vector y crear otro vector con ellos. Por ejemplo, `in.rgb` devolvería un valor `uchar3` que solo contiene las partes roja, verde y azul del píxel sin el valor alfa.

En tiempo de ejecución, RenderScript llamará a este método Kernel para cada píxel de una imagen, por lo que el valor de retorno y el parámetro son solo un valor de `uchar4` . RenderScript ejecutará muchas de estas llamadas en paralelo en todos los procesadores disponibles, por lo que RenderScript es tan poderoso. Esto también significa que no tiene que preocuparse por los hilos o la seguridad de los hilos, simplemente puede implementar lo que quiera hacer en cada píxel y RenderScript se encarga del resto.

Cuando llama a un Kernel en Java, proporciona dos variables de `Allocation` , una que contiene los datos de entrada y otra que recibirá la salida. Se llamará al método Kernel para cada valor en la `Allocation` entrada y se escribirá el resultado en la `Allocation` salida.

Métodos de la API de RenderScript Runtime

En el Kernel anterior se utilizan algunos métodos que se proporcionan de forma inmediata. RenderScript proporciona muchos de estos métodos y son vitales para casi cualquier cosa que vaya a hacer con RenderScript. Entre ellos se encuentran métodos para realizar operaciones matemáticas como `sin()` y métodos auxiliares como `mix()` que mezclan dos valores de acuerdo con otros valores. Pero también hay métodos para operaciones más complejas cuando se trata de vectores, cuaterniones y matrices.

La **referencia** oficial de la **API de RenderScript Runtime** es el mejor recurso que existe si desea saber más acerca de un método en particular o si está buscando un método específico que realice una operación común, como calcular el producto de puntos de una matriz. Puedes encontrar esta documentación **aquí** .

Implementacion de Kernel

Ahora echemos un vistazo a los detalles de lo que está haciendo este núcleo. Aquí está la primera línea en el Kernel:

```
float4 f4 = rsUnpackColor8888(in);
```

La primera línea llama al método construida en **`rsUnpackColor8888()`** que transforma la `uchar4` valor a un `float4` valores. Cada canal de color también se transforma en el rango de `0.0f - 1.0f` donde `0.0f` corresponde a un valor de byte de `0` y `1.0f` a `255` . El propósito principal de esto es hacer que todas las matemáticas en este Kernel sean mucho más simples.

```
float3 dotVector = dot(f4.rgb, gMonoMult);
```

La siguiente línea utiliza el método incorporado en **`dot()`** para calcular el producto de punto de dos vectores. `gMonoMult` es un valor constante que definimos en algunos capítulos anteriores. Dado que ambos vectores deben ser de la misma longitud para calcular el producto de puntos y también como solo queremos afectar a los canales de color y no al canal alfa de un píxel, usamos

la abreviatura `.rgb` para obtener un nuevo vector `float3` que solo contiene el Canales de color rojo, verde y azul. Aquellos de nosotros que aún recordamos de la escuela cómo funciona el producto de puntos notaremos rápidamente que el producto de puntos debe devolver solo un valor y no un vector. Sin embargo, en el código anterior estamos asignando el resultado a un vector `float3` . Esto es de nuevo una característica de RenderScript. Cuando asigna un número unidimensional a un vector, todos los elementos del vector se ajustarán a este valor. Por ejemplo, el siguiente fragmento de `2.0f` asignará `2.0f` a cada uno de los tres valores en el vector `float3` :

```
float3 example = 2.0f;
```

Por lo tanto, el resultado del producto punto anterior se asigna a cada elemento en el vector `float3` anterior.

Ahora viene la parte en la que realmente usamos la variable global `saturationLevel` para modificar la saturación de la imagen:

```
float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
```

Esto utiliza el método integrado **`mix()`** para mezclar el color original con el vector de producto de puntos que creamos anteriormente. La forma en que se mezclan entre sí está determinada por la variable global `saturationLevel` . Por lo tanto, un nivel de `saturationLevel` de `0.0f` hará que el color resultante no forme parte de los valores de color originales y solo constará de valores en el `dotVector` que dan como resultado una imagen en blanco y negro o en gris. Un valor de `1.0f` hará que el color resultante se componga completamente de los valores de color originales y los valores superiores a `1.0f` multiplicarán los colores originales para hacerlos más brillantes e intensos.

```
return rsPackColorTo8888(newColor);
```

Esta es la última parte en el núcleo. **`rsPackColorTo8888()`** transforma el vector `float3` nuevo a un valor de `uchar4` que luego se devuelve. Los valores de byte resultantes se fijan en un rango entre 0 y 255, por lo que los valores flotantes superiores a `1.0f` darán como resultado un valor de byte de 255 y los valores inferiores a `0.0` darán como resultado un valor de byte de `0` .

Y esa es toda la implementación del Kernel. Ahora solo queda una parte: Cómo llamar a un kernel en Java.

Llamando a RenderScript en Java

Lo esencial

Como ya se mencionó anteriormente para cada archivo RenderScript, se genera una clase Java que le permite interactuar con sus scripts. Estos archivos tienen el prefijo `ScriptC_` seguido del nombre del archivo RenderScript. Para crear una instancia de estas clases, primero necesita una instancia de la clase `RenderScript` :

```
final RenderScript renderScript = RenderScript.create(context);
```

El método estático `create()` se puede usar para crear una instancia de `RenderScript` desde un `Context` . Luego puede crear una instancia de la clase Java que se generó para su script. Si llamó al archivo RenderScript `saturation.rs` , la clase se llamará `ScriptC_saturation` :

```
final ScriptC_saturation script = new ScriptC_saturation(renderScript);
```

En esta clase, ahora puede establecer el nivel de saturación y llamar al Kernel. El setter que se generó para la variable `saturationLevel` tendrá el prefijo `set_` seguido del nombre de la variable:

```
script.set_saturationLevel(1.0f);
```

También hay un prefijo getter con `get_` que le permite obtener el nivel de saturación establecido actualmente:

```
float saturationLevel = script.get_saturationLevel();
```

Los kernels que defina en su RenderScript tienen el prefijo `forEach_` seguido del nombre del método Kernel. El Kernel que hemos escrito espera una `Allocation` entrada y una `Allocation` salida como sus parámetros:

```
script.forEach_saturation(inputAllocation, outputAllocation);
```

La `Allocation` entrada debe contener la imagen de entrada, y una vez que el método `forEach_saturation` haya finalizado, la asignación de salida contendrá los datos de imagen modificados.

Una vez que tenga una instancia de `Allocation` , puede copiar datos desde y hacia esas `Allocations` utilizando los métodos `copyFrom()` y `copyTo()` . Por ejemplo, puede copiar una nueva imagen en su entrada `Asignación llamando:

```
inputAllocation.copyFrom(inputBitmap);
```

De la misma manera que puede recuperar la imagen de resultado llamando a `copyTo()` en la `Allocation` salida:

```
outputAllocation.copyTo(outputBitmap);
```

Creación de instancias de asignación

Hay muchas formas de crear una `Allocation` . Una vez que tenga una instancia de `Allocation` , puede copiar nuevos datos desde y hacia esas `Allocations` con `copyTo()` y `copyFrom()` como se explicó anteriormente, pero para crearlos inicialmente debe saber con qué tipo de datos está trabajando exactamente. Vamos a empezar con la `Allocation` entrada:

Podemos usar el método estático `createFromBitmap()` para crear rápidamente nuestra `Allocation` entrada desde un `Bitmap` :

```
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, image);
```

En este ejemplo, la imagen de entrada nunca cambia, por lo que nunca necesitamos modificar la `Allocation` entrada nuevamente. Podemos reutilizarlo cada vez que el nivel de `saturationLevel` cambie para crear un nuevo `Bitmap` salida.

Crear la `Allocation` salida es un poco más complejo. Primero necesitamos crear lo que se llama un `Type` . Un `Type` se usa para indicar una `Allocation` con qué tipo de datos está tratando. Por lo general, uno usa la clase `Type.Builder` para crear rápidamente un `Type` apropiado. Echemos un vistazo al código primero:

```
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
        .setX(inputBitmap.getWidth())
        .setY(inputBitmap.getHeight())
        .create();
```

Estamos trabajando con un `Bitmap` normal de 32 bits (o en otras palabras, 4 bytes) por píxel con 4 canales de color. Es por eso que estamos eligiendo `Element.RGBA_8888` para crear el `Type` . Luego usamos los métodos `setX()` y `setY()` para establecer el ancho y la altura de la imagen de salida en el mismo tamaño que la imagen de entrada. El método `create()` luego crea el `Type` con los parámetros que especificamos.

Una vez que tengamos el `Type` correcto, podemos crear la `Allocation` salida con el método estático `createTyped()` :

```
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);
```

Ahora casi hemos terminado. También necesitamos un `Bitmap` salida en el que podamos copiar los datos de la `Allocation` salida. Para hacer esto, usamos el método estático `createBitmap()` para crear un nuevo `Bitmap` vacío con el mismo tamaño y configuración que el `Bitmap` entrada.

```
final Bitmap outputBitmap = Bitmap.createBitmap(
        inputBitmap.getWidth(),
        inputBitmap.getHeight(),
        inputBitmap.getConfig()
);
```

Y con eso tenemos todas las piezas de rompecabezas para ejecutar nuestro RenderScript.

Ejemplo completo

Ahora pongamos todo esto junto en un ejemplo:

```
// Create the RenderScript instance
final RenderScript renderScript = RenderScript.create(context);
```

```
// Create the input Allocation
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, inputBitmap);

// Create the output Type.
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
        .setX(inputBitmap.getWidth())
        .setY(inputBitmap.getHeight())
        .create();

// And use the Type to create am output Allocation
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);

// Create an empty output Bitmap from the input Bitmap
final Bitmap outputBitmap = Bitmap.createBitmap(
        inputBitmap.getWidth(),
        inputBitmap.getHeight(),
        inputBitmap.getConfig()
);

// Create an instance of our script
final ScriptC_saturation script = new ScriptC_saturation(renderScript);

// Set the saturation level
script.set_saturationLevel(2.0f);

// Execute the Kernel
script.forEach_saturation(inputAllocation, outputAllocation);

// Copy the result data to the output Bitmap
outputAllocation.copyTo(outputBitmap);

// Display the result Bitmap somewhere
someImageView.setImageBitmap(outputBitmap);
```

Conclusión

Con esta introducción, debería estar listo para escribir sus propios núcleos RenderScript para la manipulación simple de imágenes. Sin embargo, hay algunas cosas que debes tener en cuenta:

- **RenderScript solo funciona en proyectos de aplicación** : actualmente, los archivos RenderScript no pueden formar parte de un proyecto de biblioteca.
- **Tenga cuidado con la memoria** : RenderScript es muy rápido, pero también puede requerir mucha memoria. Nunca debe haber más de una instancia de `RenderScript` en cualquier momento. También debe reutilizar lo más posible. Normalmente solo necesita crear sus instancias de `Allocation` una vez y puede reutilizarlas en el futuro. Lo mismo ocurre con los `Bitmaps` salida o sus instancias de script. Reutiliza tanto como sea posible.
- **Realice su trabajo en segundo plano** : una vez más, RenderScript es muy rápido, pero no instantáneo de ninguna manera. Cualquier Kernel, especialmente los complejos, deben ejecutarse fuera del hilo de la IU en una `AsyncTask` o algo similar. Sin embargo, en su mayor parte, no tiene que preocuparse por las fugas de memoria. Todas las clases relacionadas con RenderScript solo usan el `Context` la aplicación y, por lo tanto, no producen pérdidas de memoria. ¡Pero todavía tiene que preocuparse por las cosas habituales, como la pérdida de `View` , `Activity` o cualquier instancia de `Context` que use usted mismo!

- **Use cosas integradas** : hay muchos scripts predefinidos que realizan tareas como difuminar, combinar, convertir y cambiar el tamaño de la imagen. Y hay muchos más métodos incorporados que te ayudan a implementar tus kernels. Lo más probable es que si quieres hacer algo, ya sea un script o un método que ya hace lo que estás tratando de hacer. No reinventes la rueda.

Si quieres comenzar rápidamente y jugar con el código real, te recomiendo que eches un vistazo al ejemplo del proyecto GitHub que implementa el ejemplo exacto del que se habla en este tutorial. Puedes encontrar el proyecto **aquí** . ¡Diviértete con RenderScript!

Desenfocar una imagen

Este ejemplo muestra cómo usar la API de Renderscript para desenfocar una imagen (usando Bitmap). Este ejemplo utiliza ScriptInstrinsicBlur proporcionado por la API de Renderscript de Android (API> = 17).

```
public class BlurProcessor {

    private RenderScript rs;
    private Allocation inAllocation;
    private Allocation outAllocation;
    private int width;
    private int height;

    private ScriptIntrinsicBlur blurScript;

    public BlurProcessor(RenderScript rs) {
        this.rs = rs;
    }

    public void initialize(int width, int height) {
        blurScript = ScriptIntrinsicBlur.create(rs, Element.U8_4(rs));
        blurScript.setRadius(7f); // Set blur radius. 25 is max

        if (outAllocation != null) {
            outAllocation.destroy();
            outAllocation = null;
        }

        // Bitmap must have ARGB_8888 config for this type
        Type bitmapType = new Type.Builder(rs, Element.RGBA_8888(rs))
            .setX(width)
            .setY(height)
            .setMipmaps(false) // We are using MipmapControl.MIPMAP_NONE
            .create();

        // Create output allocation
        outAllocation = Allocation.createTyped(rs, bitmapType);

        // Create input allocation with same type as output allocation
        inAllocation = Allocation.createTyped(rs, bitmapType);
    }

    public void release() {

        if (blurScript != null) {
```

```
            blurScript.destroy();
            blurScript = null;
        }

        if (inAllocation != null) {
            inAllocation.destroy();
            inAllocation = null;
        }

        if (outAllocation != null) {
            outAllocation.destroy();
            outAllocation = null;
        }
    }

    public Bitmap process(Bitmap bitmap, boolean createNewBitmap) {
        if (bitmap.getWidth() != width || bitmap.getHeight() != height) {
            // Throw error if required
            return null;
        }

        // Copy data from bitmap to input allocations
        inAllocation.copyFrom(bitmap);

        // Set input for blur script
        blurScript.setInput(inAllocation);

        // process and set data to the output allocation
        blurScript.forEach(outAllocation);

        if (createNewBitmap) {
            Bitmap returnVal = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
            outAllocation.copyTo(returnVal);
            return returnVal;
        }

        outAllocation.copyTo(bitmap);
        return bitmap;
    }
}
```

Cada script tiene un núcleo que procesa los datos y generalmente se invoca a través del método `forEach` .

```
public class BlurActivity extends AppCompatActivity {
    private BlurProcessor blurProcessor;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // setup layout and other stuff


        blurProcessor = new BlurProcessor(Renderscript.create(getApplicationContext()));
    }

    private void loadImage(String path) {
        // Load image to bitmap
        Bitmap bitmap = loadBitmapFromPath(path);

        // Initialize processor for this bitmap
```

```
        blurProcessor.release();
        blurProcessor.initialize(bitmap.getWidth(), bitmap.getHeight());

        // Blur image
        Bitmap blurImage = blurProcessor.process(bitmap, true); // Use newBitamp as false if
you don't want to create a new bitmap
    }
}
```

Esto concluyó el ejemplo aquí. Se aconseja hacer el procesamiento en un hilo de fondo.

Desenfocar una vista

BlurBitmapTask.java

```
public class BlurBitmapTask extends AsyncTask<Bitmap, Void, Bitmap> {
    private final WeakReference<ImageView> imageViewReference;
    private final RenderScript renderScript;

    private boolean shouldRecycleSource = false;

    public BlurBitmapTask(@NonNull Context context, @NonNull ImageView imageView) {
        // Use a WeakReference to ensure
        // the ImageView can be garbage collected
        imageViewReference = new WeakReference<>(imageView);
        renderScript = RenderScript.create(context);
    }

    // Decode image in background.
    @Override
    protected Bitmap doInBackground(Bitmap... params) {
        Bitmap bitmap = params[0];
        return blurBitmap(bitmap);
    }

    // Once complete, see if ImageView is still around and set bitmap.
    @Override
    protected void onPostExecute(Bitmap bitmap) {
        if (bitmap == null || isCancelled()) {
            return;
        }

        final ImageView imageView = imageViewReference.get();
        if (imageView == null) {
            return;
        }

        imageView.setImageBitmap(bitmap);
    }

    public Bitmap blurBitmap(Bitmap bitmap) {
        // https://plus.google.com/+MarioViviani/posts/fhuzYkji9zz

        //Let's create an empty bitmap with the same size of the bitmap we want to blur
        Bitmap outBitmap = Bitmap.createBitmap(bitmap.getWidth(), bitmap.getHeight(),
                Bitmap.Config.ARGB_8888);

        //Instantiate a new Renderscript
```

```
        //Create an Intrinsic Blur Script using the Renderscript
        ScriptIntrinsicBlur blurScript = ScriptIntrinsicBlur.create(renderScript,
Element.U8_4(renderScript));

        //Create the in/out Allocations with the Renderscript and the in/out bitmaps
        Allocation allIn = Allocation.createFromBitmap(renderScript, bitmap);
        Allocation allOut = Allocation.createFromBitmap(renderScript, outBitmap);

        //Set the radius of the blur
        blurScript.setRadius(25.f);

        //Perform the Renderscript
        blurScript.setInput(allIn);
        blurScript.forEach(allOut);

        //Copy the final bitmap created by the out Allocation to the outBitmap
        allOut.copyTo(outBitmap);

        // recycle the original bitmap
        // nope, we are using the original bitmap as well :/
        if (shouldRecycleSource) {
            bitmap.recycle();
        }

        //After finishing everything, we destroy the Renderscript.
        renderScript.destroy();

        return outBitmap;
    }

    public boolean isShouldRecycleSource() {
        return shouldRecycleSource;
    }

    public void setShouldRecycleSource(boolean shouldRecycleSource) {
        this.shouldRecycleSource = shouldRecycleSource;
    }
}
```

Uso:

```
ImageView imageViewOverlayOnViewToBeBlurred
        .setImageDrawable(ContextCompat.getDrawable(this, android.R.color.transparent));
View viewToBeBlurred.setDrawingCacheQuality(View.DRAWING_CACHE_QUALITY_LOW);
viewToBeBlurred.setDrawingCacheEnabled(true);
BlurBitmapTask blurBitmapTask = new BlurBitmapTask(this, imageViewOverlayOnViewToBeBlurred);
blurBitmapTask.execute(Bitmap.createBitmap(viewToBeBlurred.getDrawingCache()));
viewToBeBlurred.setDrawingCacheEnabled(false);
```

Lea RenderScript en línea: https://riptutorial.com/es/android/topic/5214/renderscript

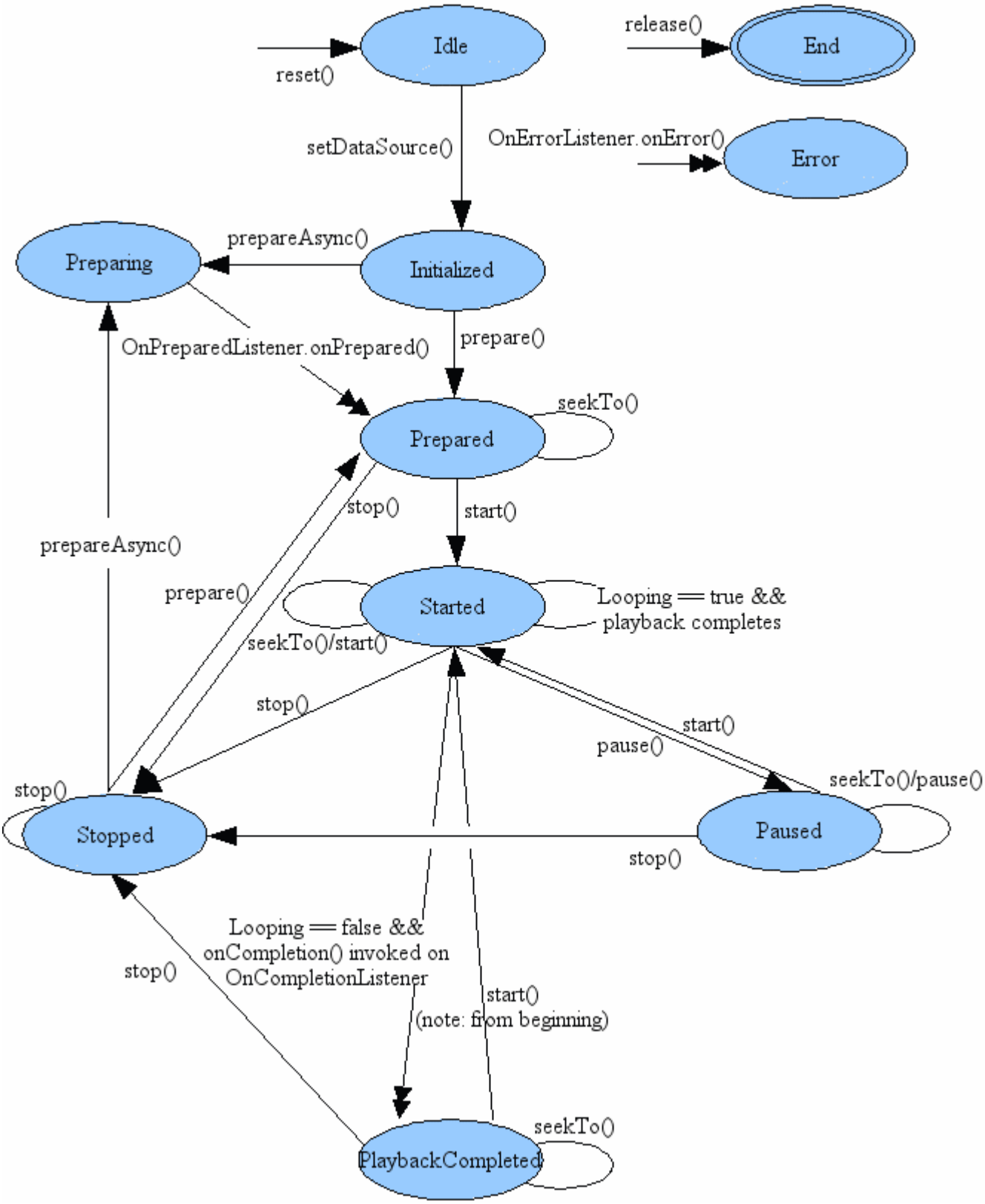
Capítulo 214: Reproductor multimedia

Sintaxis

- void setAudioStreamType (int streamtype)
- void setDataSource (contexto de contexto, uri uri)
- preparar el vacío ()
- void prepareAsync ()
- inicio vacío ()
- parada vacía ()

Observaciones

El uso de `MediaPlayer` se basa principalmente en el diagrama de estado:



Eso significa que para reproducir audio / video debe ocurrir una secuencia de acción definida en un orden específico. También establece qué acciones se pueden hacer en qué estado .

La API de MediaPlayer carece de flexibilidad (agregando decodificador personalizado y lógica de representación) y carece de soporte para la transmisión dinámica adaptativa sobre HTTP (DASH) y SmoothStreaming. Para estos, mira en ExoPlayer .

Examples

Creación básica y juego.

La clase MediaPlayer se puede usar para controlar la reproducción de archivos y secuencias de audio / video.

La creación del objeto MediaPlayer puede ser de tres tipos:

1. Medios del recurso local

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.resource);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

2. De URI local (obtenido de ContentResolver)

```
Uri myUri = ....; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

3. Desde URL externa

```
String url = "http://........"; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```

Preparación asíncrona

El `MediaPlayer$prepare()` es una llamada de bloqueo y congelará la interfaz de usuario hasta que se complete la ejecución. Para resolver este problema, se puede usar `MediaPlayer$prepareAsync()` .

```
mMediaPlayer = ... // Initialize it here
mMediaPlayer.setOnPreparedListener(new MediaPlayer.OnPreparedListener(){
    @Override
    public void onPrepared(MediaPlayer player) {
        // Called when the MediaPlayer is ready to play
        mMediaPlayer.start();
    }
}); // Set callback for when prepareAsync() finishes
mMediaPlayer.prepareAsync(); // Prepare asynchronously to not block the Main Thread
```

En las operaciones síncronas, los errores normalmente se señalarían con una excepción o un código de error, pero siempre que utilice recursos asíncronos, debe asegurarse de que se notifique a la aplicación de los errores de manera adecuada. Para MediaPlayer,

```
mMediaPlayer.setOnErrorListener(new MediaPlayer.OnErrorListener(){
        @Override
        public boolean onError(MediaPlayer mp, int what, int extra) {
            // ... react appropriately ...
            // The MediaPlayer has moved to the Error state, must be reset!
            // Then return true if the error has been handled
        }
});
```

Obteniendo tonos del sistema

Este ejemplo muestra cómo obtener el URI de los tonos de llamada del sistema (`RingtoneManager.TYPE_RINGTONE`):

```
private List<Uri> loadLocalRingtonesUris() {
        List<Uri> alarms = new ArrayList<>();
        try {
            RingtoneManager ringtoneMgr = new RingtoneManager(getActivity());
            ringtoneMgr.setType(RingtoneManager.TYPE_RINGTONE);

            Cursor alarmsCursor = ringtoneMgr.getCursor();
            int alarmsCount = alarmsCursor.getCount();
            if (alarmsCount == 0 && !alarmsCursor.moveToFirst()) {
                alarmsCursor.close();
                return null;
            }

            while (!alarmsCursor.isAfterLast() && alarmsCursor.moveToNext()) {
                int currentPosition = alarmsCursor.getPosition();
                alarms.add(ringtoneMgr.getRingtoneUri(currentPosition));
            }

        } catch (Exception ex) {
            ex.printStackTrace();
        }

        return alarms;
    }
```

La lista depende de los tipos de tonos de llamada solicitados. Las posibilidades son:

- `RingtoneManager.TYPE_RINGTONE`
- `RingtoneManager.TYPE_NOTIFICATION`
- `RingtoneManager.TYPE_ALARM`
- `RingtoneManager.TYPE_ALL = TYPE_RINGTONE | TYPE_NOTIFICATION | TYPE_ALARM`

Para obtener los tonos de llamada como `android.media.Ringtone` todos los `Uri` deben ser resueltos por el `RingtoneManager` :

```
android.media.Ringtone osRingtone = RingtoneManager.getRingtone(context, uri);
```

Para reproducir el sonido, utiliza el método:

```
public void setDataSource(Context context, Uri uri)
```

Desde `android.media.MediaPlayer` . `MediaPlayer` debe inicializarse y prepararse de acuerdo con el diagrama de estado

Obtención y configuración del volumen del sistema.

Tipos de flujo de audio

Hay diferentes perfiles de secuencias de tonos de llamada. Cada uno de ellos tiene su volumen diferente.

Cada ejemplo aquí está escrito para `AudioManager.STREAM_RING` tipo de flujo. Sin embargo, este no es el único. Los tipos de flujo disponibles son:

- `STREAM_ALARM`
- `STREAM_DTMF`
- `STREAM_MUSIC`
- `STREAM_NOTIFICATION`
- `STREAM_RING`
- `STREAM_SYSTEM`
- `STREAM_VOICE_CALL`

Ajuste de volumen

Para obtener el volumen de perfil específico, llame a:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int currentVolume = audioManager.getStreamVolume(AudioManager.STREAM_RING);
```

Este valor es muy poco útil, cuando se desconoce el valor máximo para la transmisión:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int streamMaxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM_RING);
```

La relación de esos dos valores dará un volumen relativo (0 <volumen <1):

```
float volume = ((float) currentVolume) / streamMaxVolume
```

Ajustando el volumen en un solo paso

Para aumentar el volumen de la transmisión en un paso, llame a:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
```

```
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_RAISE, 0);
```

Para reducir el volumen de la transmisión en un solo paso, llame a:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_LOWER, 0);
```

Configuración de MediaPlayer para utilizar un tipo de transmisión específico

Hay una función auxiliar de la clase `MediaPlayer` para hacer esto.
Simplemente llame a `void setAudioStreamType(int streamtype)` :

```
MediaPlayer mMedia = new MediaPlayer();
mMedia.setAudioStreamType(AudioManager.STREAM_RING);
```

Reproductor multimedia con progreso de búfer y posición de juego

```
public class SoundActivity extends Activity  {

    private MediaPlayer mediaPlayer;
    ProgressBar progress_bar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tool_sound);
        mediaPlayer = new MediaPlayer();
        mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
        progress_bar = (ProgressBar) findViewById(R.id.progress_bar);

        btn_play_stop.setEnabled(false);
        btn_play_stop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(mediaPlayer.isPlaying()) {
                    mediaPlayer.pause();
                    btn_play_stop.setImageResource(R.drawable.ic_pause_black_24dp);
                } else {
                    mediaPlayer.start();
                    btn_play_stop.setImageResource(R.drawable.ic_play_arrow_black_24px);
                }
            }
        });

        mediaPlayer.setDataSource(proxyUrl);
        mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                observer.stop();
```

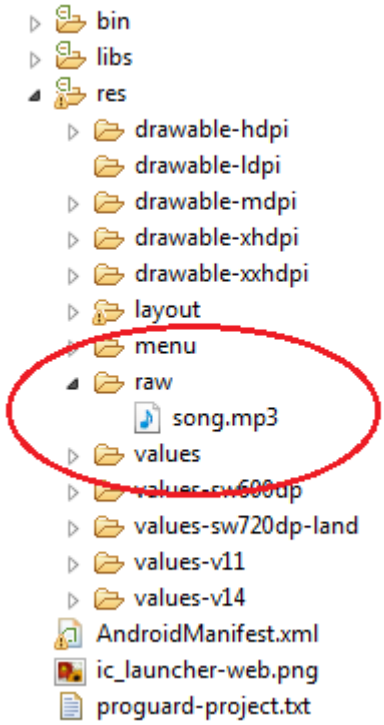
```
                progress_bar.setProgress(mp.getCurrentPosition());
                // TODO Auto-generated method stub
                mediaPlayer.stop();
                mediaPlayer.reset();
            }
        });
        mediaPlayer.setOnBufferingUpdateListener(new MediaPlayer.OnBufferingUpdateListener() {
            @Override
            public void onBufferingUpdate(MediaPlayer mp, int percent) {
                progress_bar.setSecondaryProgress(percent);
            }
        });
        mediaPlayer.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
            @Override
            public void onPrepared(MediaPlayer mediaPlayer) {
                btn_play_stop.setEnabled(true);

            }
        });
        observer = new MediaObserver();
        mediaPlayer.prepare();
        mediaPlayer.start();
        new Thread(observer).start();
    }


    private MediaObserver observer = null;

    private class MediaObserver implements Runnable {
        private AtomicBoolean stop = new AtomicBoolean(false);

        public void stop() {
            stop.set(true);
        }

        @Override
        public void run() {
            while (!stop.get()) {
                progress_bar.setProgress((int)((double)mediaPlayer.getCurrentPosition() /
(double)mediaPlayer.getDuration()*100));
                try {
                    Thread.sleep(200);
                } catch (Exception ex) {
                    Logger.log(ToolSoundActivity.this, ex);
                }

            }
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        mediaPlayer.stop();
    }
}
```

```
<LinearLayout
    android:gravity="bottom"
    android:layout_gravity="bottom"
```

```
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:weightSum="1">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageButton
            app:srcCompat="@drawable/ic_play_arrow_black_24px"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:id="@+id/btn_play_stop" />

        <ProgressBar
            android:padding="8dp"
            android:progress="0"
            android:id="@+id/progress_bar"
            style="@style/Widget.AppCompat.ProgressBar.Horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center" />

    </LinearLayout>

</LinearLayout>
```

Importar audio en androidstudio y reproducirlo

Este es un ejemplo de cómo reproducir un archivo de audio que ya tiene en su PC / computadora portátil. Primero cree un nuevo directorio en resolución y asígnele el nombre como crudo



Copie el audio que desea reproducir en esta carpeta. Puede ser un archivo .mp3 o .wav.

Ahora, por ejemplo, al hacer clic en el botón, desea reproducir este sonido, así es como se hace.

```
public class MainActivity extends AppCompatActivity {
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.aboutapp_activity);

    MediaPlayer song=MediaPlayer.create(this, R.raw.song);

    Button button=(Button)findViewById(R.id.button);
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
           song.start();
        }
    });
}
}
```

Esto reproducirá la canción solo una vez cuando se haga clic en el botón, si desea reproducir la canción en cada botón, haga clic en el código de esta manera

```
 public class MainActivity extends AppCompatActivity {
 @Override
 public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.aboutapp_activity);

   MediaPlayer song=MediaPlayer.create(this, R.raw.song);

   Button button=(Button)findViewById(R.id.button);
   button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (song.isPlaying()) {
            song.reset();
            song= MediaPlayer.create(getApplicationContext(), R.raw.song);
       }
         song.start();
     }
   });
 }
}
```

Lea Reproductor multimedia en línea: https://riptutorial.com/es/android/topic/1851/reproductor-multimedia

Capítulo 215: RestricciónDisposición

Introducción

`ConstraintLayout` es un `ViewGroup` que le permite `ViewGroup` y dimensionar widgets de una manera flexible. Es compatible con Android 2.3 (API nivel 9) y superior.

Le permite crear diseños grandes y complejos con una jerarquía de vista plana. Es similar a `RelativeLayout` en el sentido de que todas las vistas están diseñadas de acuerdo con las relaciones entre las vistas de hermanos y el diseño principal, pero es más flexible que `RelativeLayout` y más fácil de usar con el Editor de diseño de Android Studio.

Sintaxis

- **RestricciónDisposición**
    - public void addView (Ver hijo, índice int, ViewGroup.LayoutParams params)
    - Public ConstraintLayout.LayoutParams generateLayoutParams (attributeSet attrs)
    - public void onViewAdded (Vista de vista)
    - public void onViewRemoved (Vista de vista)
    - public void removeView (Vista de vista)
    - solicitud de anulación públicaLayout ()
    - checkLayoutParams booleanas protegidas (ViewGroup.LayoutParams params)
    - ConstraintLayout.LayoutParams generaDefaultLayoutParams () protegido
    - ViewGroup.LayoutParams generaLayoutParams protegidos (ViewGroup.LayoutParams params)
    - void protegido onLayout (booleano cambiado, int izquierda, int arriba, int derecha, int abajo)
    - void protegido onMeasure (int widthMeasureSpec, int heightMeasureSpec)
- **RestricciónLayout.LayoutParams**
    - public void resolutionLayoutDirection (int layoutDirection)
    - validación del vacío público ()
    - setBaseAttributes void protegidos (TypedArray a, int widthAttr, int heightAttr)

Parámetros

Parámetro	Detalles
niño	La `View` que se añadirá al diseño.
índice	El índice de la `View` en la jerarquía de diseño.
params	Los `LayoutParams` de la `View`
attrs	El `AttributeSet` que define los `LayoutParams`
ver	La `View` que se ha agregado o eliminado
cambiado	Indica si esta `View` ha cambiado de tamaño o posición
izquierda	La posición izquierda, relativa a la `View` padre
parte superior	La posición superior, relativa a la `View` principal
Correcto	La posición correcta, relativa a la `View` padre
fondo	La posición inferior, relativa a la `View` padre
anchoMedidaSpec	Los requisitos de espacio horizontal impuestos por la `View` padre.
alturaMedidaSpec	Los requisitos de espacio vertical impuestos por la `View` padre
layoutDirection	-
una	-
widthAttr	-
alturaAttr	-

Observaciones

En Google IO 2016, Google anunció un nuevo diseño de Android llamado ConstraintLayout. Presta atención porque actualmente, este diseño es una **versión Beta** .

Para más información sobre el diseño de restricciones:

https://codelabs.developers.google.com/codelabs/constraint-layout/index.html

Examples

Agregando ConstraintLayout a su proyecto

Para trabajar con ConstraintLayout, necesita Android Studio versión 2.2 o más reciente y al menos tiene la versión 32 (o superior) del repositorio de soporte de Android.

1. Agregue la biblioteca de diseño de restricciones como una dependencia en su archivo `build.gradle` :

```
dependencies {
   compile 'com.android.support.constraint:constraint-layout:1.0.2'
}
```

2. Proyecto de sincronización

Para agregar un nuevo diseño de restricción a su proyecto:

1. **Haga clic derecho** en el directorio de diseño de su módulo, luego haga clic en `New > XML > Layout XML.`
2. Ingrese un **nombre** para el diseño e ingrese `"android.support.constraint.ConstraintLayout"` para la etiqueta raíz.
3. Haga clic en **Finalizar** .

De lo contrario simplemente agregue en un archivo de diseño:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</android.support.constraint.ConstraintLayout>
```

Las cadenas

Desde `ConstraintLayout` alpha 9, las **cadenas** están disponibles. Una **cadena** es un conjunto de vistas dentro de un `ConstraintLayout` que se conectan de forma bidireccional entre ellas, es decir, **A está** conectada a **B** con una restricción y **B está** conectada a **A** con otra restricción.

Ejemplo:

```
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- this view is linked to the bottomTextView -->
    <TextView
        android:id="@+id/topTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:text="TextView"
        app:layout_constraintBottom_toTopOf="@+id/bottomTextView"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_chainPacked="true"/>

    <!-- this view is linked to the topTextView at the same time -->
    <TextView
        android:id="@+id/bottomTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom\nMkay"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/topTextView"/>

</android.support.constraint.ConstraintLayout>
```

En este ejemplo, las dos vistas están colocadas una debajo de la otra y ambas están centradas verticalmente. Puede cambiar la posición vertical de estas vistas ajustando el **sesgo de** la cadena. Agregue el siguiente código al primer elemento de una cadena:

```
app:layout_constraintVertical_bias="0.2"
```

En una cadena vertical, el primer elemento es una vista superior y en una cadena horizontal es la vista más a la izquierda. El primer elemento define el comportamiento de toda la cadena.

Las cadenas son una nueva característica y se actualizan con frecuencia. Aquí hay una documentación oficial de Android sobre cadenas.

Lea RestricciónDisposición en línea:
https://riptutorial.com/es/android/topic/5076/restricciondisposicion

Capítulo 216: RestricciónSet

Introducción

Esta clase le permite definir mediante programación un conjunto de restricciones que se utilizarán con `ConstraintLayout` . Le permite crear y guardar restricciones, y aplicarlas a un `ConstraintLayout` existente.

Examples

Restricción establecida con ContraintLayout mediante programación

```
import android.content.Context;
import android.os.Bundle;
import android.support.constraint.ConstraintLayout;
import android.support.constraint.ConstraintSet;
import android.support.transition.TransitionManager;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

public class MainActivity extends AppCompatActivity {
    ConstraintSet mConstraintSet1 = new ConstraintSet(); // create a Constraint Set
    ConstraintSet mConstraintSet2 = new ConstraintSet(); // create a Constraint Set
    ConstraintLayout mConstraintLayout; // cache the ConstraintLayout
    boolean mOld = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Context context = this;
        mConstraintSet2.clone(context, R.layout.state2); // get constraints from layout
        setContentView(R.layout.state1);
        mConstraintLayout = (ConstraintLayout) findViewById(R.id.activity_main);
        mConstraintSet1.clone(mConstraintLayout); // get constraints from ConstraintSet
    }

    public void foo(View view) {
        TransitionManager.beginDelayedTransition(mConstraintLayout);
        if (mOld = !mOld) {
            mConstraintSet1.applyTo(mConstraintLayout); // set new constraints
        }  else {
            mConstraintSet2.applyTo(mConstraintLayout); // set new constraints
        }
    }
}
```

Lea RestricciónSet en línea: https://riptutorial.com/es/android/topic/9334/restriccionset

Capítulo 217: Retrofit2

Introducción

La página oficial de Retrofit se describe como

Un cliente REST seguro para el tipo para Android y Java.

Retrofit convierte su API REST en una interfaz Java. Utiliza anotaciones para describir solicitudes HTTP, la sustitución de parámetros de URL y el soporte de parámetros de consulta están integrados de forma predeterminada. Además, proporciona funcionalidad para el cuerpo de solicitud multiparte y las cargas de archivos.

Observaciones

Dependencias para la biblioteca de retrofit:

De la documentación oficial :

Gradle:

```
dependencies {
    ...
    compile 'com.squareup.retrofit2:converter-gson:2.3.0'
    compile 'com.squareup.retrofit2:retrofit:2.3.0'
    ...
}
```

Maven

```
<dependency>
  <groupId>com.squareup.retrofit2</groupId>
  <artifactId>retrofit</artifactId>
  <version>2.3.0</version>
</dependency>
```

Examples

Una simple solicitud GET

Vamos a mostrar cómo hacer una solicitud `GET` a una API que responde con un objeto `JSON` o una matriz `JSON` . Lo primero que debemos hacer es agregar las dependencias de Retrofit y `GSON` Converter al archivo gradle de nuestro módulo.

Agregue las dependencias para la biblioteca de actualización como se describe en la sección Comentarios.

Ejemplo de objeto JSON esperado:

```
{
    "deviceId": "56V56C14SF5B4SF",
    "name": "Steven",
    "eventCount": 0
}
```

Ejemplo de matriz JSON:

```
[
    {
        "deviceId": "56V56C14SF5B4SF",
        "name": "Steven",
        "eventCount": 0
    },
    {
        "deviceId": "35A80SF3QDV7M9F",
        "name": "John",
        "eventCount": 2
    }
]
```

Ejemplo de clase de modelo correspondiente:

```
public class Device
{
    @SerializedName("deviceId")
    public String id;

    @SerializedName("name")
    public String name;

    @SerializedName("eventCount")
    public int eventCount;
}
```

Las anotaciones de `@SerializedName` aquí provienen de la biblioteca `GSON` y nos permiten `serialize` y `deserialize` esta clase a `JSON` utilizando el nombre serializado como claves. Ahora podemos crear la interfaz para la API que realmente obtendrá los datos del servidor.

```
public interface DeviceAPI
{
    @GET("device/{deviceId}")
    Call<Device> getDevice (@Path("deviceId") String deviceID);

    @GET("devices")
    Call<List<Device>> getDevices();
}
```

Hay mucho que hacer aquí en un espacio bastante compacto, así que vamos a desglosarlo:

- La anotación `@GET` viene de Retrofit y le dice a la biblioteca que estamos definiendo una solicitud GET.

- La ruta entre los paréntesis es el punto final al que debe llegar nuestra solicitud GET (estableceremos la URL base un poco más adelante).
- Los paréntesis nos permiten reemplazar partes de la ruta en el tiempo de ejecución para que podamos pasar argumentos.
- La función que estamos definiendo se llama `getDevice` y toma la identificación del dispositivo que queremos como argumento.
- La anotación `@PATH` le dice a Retrofit que este argumento debe reemplazar el marcador de posición "deviceId" en la ruta.
- La función devuelve un objeto de `Call` de tipo `Device` .

Creando una clase envoltura:

Ahora haremos una pequeña clase de envoltorio para nuestra API para mantener el código de inicialización de Retrofit bien adaptado.

```
public class DeviceAPIHelper
{
    public final DeviceAPI api;

    private DeviceAPIHelper ()
    {

        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl("http://example.com/")
                .addConverterFactory(GsonConverterFactory.create())
                .build();

        api = retrofit.create(DeviceAPI.class);
    }
}
```

Esta clase crea una instancia GSON para poder analizar la respuesta JSON, crea una instancia Retrofit con nuestra URL base y un GSONConverter y luego crea una instancia de nuestra API.

Llamando a la API:

```
// Getting a JSON object
Call<Device> callObject = api.getDevice(deviceID);
callObject.enqueue(new Callback<Response<Device>>()
{
    @Override
    public void onResponse (Call<Device> call, Response<Device> response)
    {
        if (response.isSuccessful())
        {
           Device device = response.body();
        }
    }

    @Override
    public void onFailure (Call<Device> call, Throwable t)
    {
        Log.e(TAG, t.getLocalizedMessage());
    }
});
```

```
// Getting a JSON array
Call<List<Device>> callArray = api.getDevices();
callArray.enqueue(new Callback<Response<List<Device>>()
{
    @Override
    public void onResponse (Call<List<Device>> call, Response<List<Device>> response)
    {
        if (response.isSuccessful())
        {
           List<Device> devices = response.body();
        }
    }

    @Override
    public void onFailure (Call<List<Device>> call, Throwable t)
    {
        Log.e(TAG, t.getLocalizedMessage());
    }
});
```

Esto utiliza nuestra interfaz API para crear un `Call<Device>` objeto y crear una `Call<List<Device>>` respectivamente. La llamada en `enqueue` le dice a Retrofit que haga esa llamada en un hilo de fondo y devuelva el resultado a la devolución de llamada que estamos creando aquí.

Nota: el análisis de una matriz JSON de objetos primitivos (como *String, Integer, Boolean* y *Double*) es similar al análisis de una matriz JSON. Sin embargo, no necesitas tu propia clase de modelo. Puede obtener la matriz de cadenas, por ejemplo, al tener el tipo de retorno de la llamada como `Call<List<String>>` .

Añadir registro a Retrofit2

Las solicitudes de actualización se pueden registrar mediante un interceptor. Hay varios niveles de detalle disponibles: NINGUNO, BÁSICO, LÍDERES, CUERPO. Ver el proyecto Github aquí .

1. Añadir dependencia a build.gradle:

```
compile 'com.squareup.okhttp3:logging-interceptor:3.8.1'
```

2. Agregue el interceptor de registro al crear Retrofit:

```
  HttpLoggingInterceptor loggingInterceptor = new HttpLoggingInterceptor();
  loggingInterceptor.setLevel(LoggingInterceptor.Level.BODY);
  OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
          .addInterceptor(loggingInterceptor)
          .build();
  Retrofit retrofit = new Retrofit.Builder()
          .baseUrl("http://example.com/")
          .client(okHttpClient)
          .addConverterFactory(GsonConverterFactory.create(gson))
          .build();
```

Exponer los registros en la Terminal (Android Monitor) es algo que debe evitarse en la versión de lanzamiento, ya que puede provocar la exposición no deseada de información crítica como, por

ejemplo, tokens de autenticación, etc.

Para evitar que los registros se expongan en el tiempo de ejecución, verifique la siguiente condición

```
if(BuildConfig.DEBUG){
     //your interfector code here
    }
```

Por ejemplo:

```
HttpLoggingInterceptor loggingInterceptor = new HttpLoggingInterceptor();
if(BuildConfig.DEBUG){
     //print the logs in this case
    loggingInterceptor.setLevel(LoggingInterceptor.Level.BODY);
}else{
    loggingInterceptor.setLevel(LoggingInterceptor.Level.NONE);
}

OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
      .addInterceptor(loggingInterceptor)
      .build();

Retrofit retrofit = new Retrofit.Builder()
      .baseUrl("http://example.com/")
      .client(okHttpClient)
      .addConverterFactory(GsonConverterFactory.create(gson))
      .build();
```

Subiendo un archivo a través de Multipart

Declara tu interfaz con las anotaciones de Retrofit2:

```
public interface BackendApiClient {
    @Multipart
    @POST("/uploadFile")
    Call<RestApiDefaultResponse> uploadPhoto(@Part("file\"; filename=\"photo.jpg\" ")
RequestBody photo);
}
```

Donde `RestApiDefaultResponse` es una clase personalizada que contiene la respuesta.

Construyendo la implementación de su API y encolando la llamada:

```
Retrofit retrofit = new Retrofit.Builder()
                                .addConverterFactory(GsonConverterFactory.create())
                                .baseUrl("http://<yourhost>/")
                                .client(okHttpClient)
                                .build();

BackendApiClient apiClient = retrofit.create(BackendApiClient.class);
RequestBody reqBody = RequestBody.create(MediaType.parse("image/jpeg"), photoFile);
Call<RestApiDefaultResponse> call = apiClient.uploadPhoto(reqBody);
call.enqueue(<your callback function>);
```

Reequipamiento con interceptor OkHttp

Este ejemplo muestra cómo usar un interceptor de solicitud con OkHttp. Esto tiene numerosos casos de uso tales como:

- Añadiendo `header` universal a la solicitud. Por ejemplo, autenticar una solicitud
- Depuración de aplicaciones en red.
- Recuperando `response` cruda
- Registro de transacciones de red, etc.
- Establecer agente de usuario personalizado

```
Retrofit.Builder builder = new Retrofit.Builder()
        .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
        .addConverterFactory(GsonConverterFactory.create())
        .baseUrl("https://api.github.com/");

if (!TextUtils.isEmpty(githubToken)) {
    // `githubToken`: Access token for GitHub
    OkHttpClient client = new OkHttpClient.Builder().addInterceptor(new Interceptor() {
        @Override public Response intercept(Chain chain) throws IOException {
            Request request = chain.request();
            Request newReq = request.newBuilder()
                    .addHeader("Authorization", format("token %s", githubToken))
                    .build();
            return chain.proceed(newReq);
        }
    }).build();

    builder.client(client);
}

return builder.build().create(GithubApi.class);
```

Ver el tema OkHttp para más detalles.

Encabezado y cuerpo: un ejemplo de autenticación

Las anotaciones `@Header` y `@Body` se pueden colocar en las firmas del método y Retrofit las creará automáticamente en función de sus modelos.

```
public interface MyService {
     @POST("authentication/user")
     Call<AuthenticationResponse> authenticateUser(@Body AuthenticationRequest request,
@Header("Authorization") String basicToken);
}
```

AuthenticaionRequest es nuestro modelo, un POJO, que contiene la información que necesita el servidor. Para este ejemplo, nuestro servidor desea la clave y el secreto del cliente.

```
public class AuthenticationRequest {
     String clientKey;
     String clientSecret;
}
```

Tenga en `@Header("Authorization")` que en `@Header("Authorization")` estamos especificando que estamos `@Header("Authorization")` el encabezado de Autorización. Los otros encabezados se rellenarán automáticamente, ya que Retrofit puede inferir qué se basan en el tipo de objetos que enviamos y esperamos a cambio.

Creamos nuestro servicio de Retrofit en alguna parte. Nos aseguramos de utilizar HTTPS.

```
Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https:// some example site")
            .client(client)
            .build();
MyService myService = retrofit.create(MyService.class)
```

Entonces podemos usar nuestro servicio.

```
AuthenticationRequest request = new AuthenticationRequest();
request.setClientKey(getClientKey());
request.setClientSecret(getClientSecret());
String basicToken = "Basic " + token;
myService.authenticateUser(request, basicToken);
```

Sube múltiples archivos usando Retrofit como multiparte

Una vez que haya configurado el entorno Retrofit en su proyecto, puede usar el siguiente ejemplo que muestra cómo cargar varios archivos utilizando Retrofit:

```
private void mulipleFileUploadFile(Uri[] fileUri) {
    OkHttpClient okHttpClient = new OkHttpClient();
    OkHttpClient clientWith30sTimeout = okHttpClient.newBuilder()
            .readTimeout(30, TimeUnit.SECONDS)
            .build();

    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl(API_URL_BASE)
            .addConverterFactory(new MultiPartConverter())
            .client(clientWith30sTimeout)
            .build();

    WebAPIService service = retrofit.create(WebAPIService.class); //here is the interface
which you have created for the call service
    Map<String, okhttp3.RequestBody> maps = new HashMap<>();

    if (fileUri!=null && fileUri.length>0) {
        for (int i = 0; i < fileUri.length; i++) {
            String filePath = getRealPathFromUri(fileUri[i]);
            File file1 = new File(filePath);

            if (filePath != null && filePath.length() > 0) {
                if (file1.exists()) {
                    okhttp3.RequestBody requestFile =
okhttp3.RequestBody.create(okhttp3.MediaType.parse("multipart/form-data"), file1);
                    String filename = "imagePath" + i; //key for upload file like : imagePath0
                    maps.put(filename + "\"; filename=\"" + file1.getName(), requestFile);
                }
            }
```

```
        }
    }

    String descriptionString = " string request";//
    //hear is the your json request
    Call<String> call = service.postFile(maps, descriptionString);
    call.enqueue(new Callback<String>() {
        @Override
        public void onResponse(Call<String> call,
                               Response<String> response) {
            Log.i(LOG_TAG, "success");
            Log.d("body==>", response.body().toString() + "");
            Log.d("isSuccessful==>", response.isSuccessful() + "");
            Log.d("message==>", response.message() + "");
            Log.d("raw==>", response.raw().toString() + "");
            Log.d("raw().networkResponse()", response.raw().networkResponse().toString() +
"");
        }

        @Override
        public void onFailure(Call<String> call, Throwable t) {
            Log.e(LOG_TAG, t.getMessage());
        }
    });
}

public String getRealPathFromUri(final Uri uri) { // function for file path from uri,
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT &&
DocumentsContract.isDocumentUri(mContext, uri)) {
        // ExternalStorageProvider
        if (isExternalStorageDocument(uri)) {
            final String docId = DocumentsContract.getDocumentId(uri);
            final String[] split = docId.split(":");
            final String type = split[0];

            if ("primary".equalsIgnoreCase(type)) {
                return Environment.getExternalStorageDirectory() + "/" + split[1];
            }
        }
        // DownloadsProvider
        else if (isDownloadsDocument(uri)) {

            final String id = DocumentsContract.getDocumentId(uri);
            final Uri contentUri = ContentUris.withAppendedId(
                    Uri.parse("content://downloads/public_downloads"), Long.valueOf(id));

            return getDataColumn(mContext, contentUri, null, null);
        }
        // MediaProvider
        else if (isMediaDocument(uri)) {
            final String docId = DocumentsContract.getDocumentId(uri);
            final String[] split = docId.split(":");
            final String type = split[0];

            Uri contentUri = null;
            if ("image".equals(type)) {
                contentUri = MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
            } else if ("video".equals(type)) {
                contentUri = MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
            } else if ("audio".equals(type)) {
                contentUri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
```

```
            }

            final String selection = "_id=?";
            final String[] selectionArgs = new String[]{
                    split[1]
            };

            return getDataColumn(mContext, contentUri, selection, selectionArgs);
        }
    }
    // MediaStore (and general)
    else if ("content".equalsIgnoreCase(uri.getScheme())) {

        // Return the remote address
        if (isGooglePhotosUri(uri))
            return uri.getLastPathSegment();

        return getDataColumn(mContext, uri, null, null);
    }
    // File
    else if ("file".equalsIgnoreCase(uri.getScheme())) {
        return uri.getPath();
    }

    return null;
}
```

A continuación se muestra la interfaz.

```
public interface WebAPIService {
    @Multipart
    @POST("main.php")
    Call<String> postFile(@PartMap Map<String,RequestBody> Files, @Part("json") String
description);
}
```

Descarga un archivo del servidor usando Retrofit2

Declaración de interfaz para descargar un archivo.

```
public interface ApiInterface {
    @GET("movie/now_playing")
    Call<MovieResponse> getNowPlayingMovies(@Query("api_key") String apiKey, @Query("page")
int page);

    // option 1: a resource relative to your base URL
    @GET("resource/example.zip")
    Call<ResponseBody> downloadFileWithFixedUrl();

    // option 2: using a dynamic URL
    @GET
    Call<ResponseBody> downloadFileWithDynamicUrl(@Url String fileUrl);
}
```

La opción 1 se usa para descargar un archivo del servidor que tiene una URL fija. y la opción 2 se utiliza para pasar un valor dinámico como URL completa para solicitar una llamada. Esto puede

ser útil al descargar archivos, que dependen de parámetros como el usuario o el tiempo.

Configuración de reequipamiento para hacer llamadas a API

```
public class ServiceGenerator {

    public static final String API_BASE_URL = "http://your.api-base.url/";

    private static OkHttpClient.Builder httpClient = new OkHttpClient.Builder();

    private static Retrofit.Builder builder =
            new Retrofit.Builder()
            .baseUrl(API_BASE_URL)
            .addConverterFactory(GsonConverterFactory.create());

    public static <S> S createService(Class<S> serviceClass){
        Retrofit retrofit = builder.client(httpClient.build()).build();
        return retrofit.create(serviceClass);
    }

}
```

Ahora, haga la implementación de api para descargar archivos desde el servidor

```
private void downloadFile(){
        ApiInterface apiInterface = ServiceGenerator.createService(ApiInterface.class);

        Call<ResponseBody> call = apiInterface.downloadFileWithFixedUrl();

        call.enqueue(new Callback<ResponseBody>() {
            @Override
            public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {
                if (response.isSuccessful()){
                    boolean writtenToDisk = writeResponseBodyToDisk(response.body());

                    Log.d("File download was a success? ", String.valueOf(writtenToDisk));
                }
            }

            @Override
            public void onFailure(Call<ResponseBody> call, Throwable t) {

            }
        });
    }
```

Y después de obtener respuesta en la devolución de llamada, codifique algún IO estándar para guardar el archivo en el disco. Aquí está el código:

```
private boolean writeResponseBodyToDisk(ResponseBody body) {
        try {
            // todo change the file location/name according to your needs
            File futureStudioIconFile = new File(getExternalFilesDir(null) + File.separator +
"Future Studio Icon.png");

            InputStream inputStream = null;
            OutputStream outputStream = null;
```

```
            try {
                byte[] fileReader = new byte[4096];

                long fileSize = body.contentLength();
                long fileSizeDownloaded = 0;

                inputStream = body.byteStream();
                outputStream = new FileOutputStream(futureStudioIconFile);

                while (true) {
                    int read = inputStream.read(fileReader);

                    if (read == -1) {
                        break;
                    }

                    outputStream.write(fileReader, 0, read);

                    fileSizeDownloaded += read;

                    Log.d("File Download: " , fileSizeDownloaded + " of " + fileSize);
                }

                outputStream.flush();

                return true;
            } catch (IOException e) {
                return false;
            } finally {
                if (inputStream != null) {
                    inputStream.close();
                }

                if (outputStream != null) {
                    outputStream.close();
                }
            }
        } catch (IOException e) {
            return false;
        }
    }
```

Tenga en cuenta que hemos especificado **ResponseBody** como tipo de retorno, de lo contrario, Retrofit intentará analizarlo y convertirlo, lo que no tiene sentido cuando está descargando un archivo.

Si desea más información sobre los productos de reequipamiento, acceda a este enlace, ya que es muy útil. [1]: https://futurestud.io/blog/retrofit-getting-started-and-android-client

Depurando con stetho

Agregue las siguientes dependencias a su aplicación.

```
compile 'com.facebook.stetho:stetho:1.5.0'
compile 'com.facebook.stetho:stetho-okhttp3:1.5.0'
```

En el método `onCreate` su clase de aplicación, llame a lo siguiente.

```
Stetho.initializeWithDefaults(this);
```

Al crear su instancia de `Retrofit` , cree una instancia de OkHttp personalizada.

```
OkHttpClient.Builder clientBuilder = new OkHttpClient.Builder();
clientBuilder.addNetworkInterceptor(new StethoInterceptor());
```

A continuación, establezca esta instancia de OkHttp personalizada en la instancia de actualización.

```
Retrofit retrofit = new Retrofit.Builder()
    // ...
    .client(clientBuilder.build())
    .build();
```

Ahora conecte su teléfono a su computadora, inicie la aplicación y escriba `chrome://inspect` en su navegador Chrome. Las llamadas a la red de actualización ahora deben aparecer para que las inspeccione.

Retrofit 2 Custom Xml Converter

Añadiendo dependencias en el archivo build.gradle.

```
dependencies {
    ....
    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile ('com.thoughtworks.xstream:xstream:1.4.7') {
        exclude group: 'xmlpull', module: 'xmlpull'
    }
    ....
}
```

Luego crea Converter Factory

```
public class XStreamXmlConverterFactory extends Converter.Factory {

    /** Create an instance using a default {@link com.thoughtworks.xstream.XStream} instance
for conversion. */
    public static XStreamXmlConverterFactory create() {
        return create(new XStream());
    }

    /** Create an instance using {@code xStream} for conversion. */
    public static XStreamXmlConverterFactory create(XStream xStream) {
        return new XStreamXmlConverterFactory(xStream);
    }

    private final XStream xStream;

    private XStreamXmlConverterFactory(XStream xStream) {
        if (xStream == null) throw new NullPointerException("xStream == null");
```

```
        this.xStream = xStream;
    }

    @Override
    public Converter<ResponseBody, ?> responseBodyConverter(Type type, Annotation[]
annotations, Retrofit retrofit) {

        if (!(type instanceof Class)) {
            return null;
        }

        Class<?> cls = (Class<?>) type;

        return new XStreamXmlResponseBodyConverter<>(cls, xStream);
    }

    @Override
    public Converter<?, RequestBody> requestBodyConverter(Type type,
          Annotation[] parameterAnnotations, Annotation[] methodAnnotations, Retrofit
retrofit) {

        if (!(type instanceof Class)) {
            return null;
        }

        return new XStreamXmlRequestBodyConverter<>(xStream);
    }
}
```

crear una clase para manejar la solicitud de cuerpo.

```
final class XStreamXmlResponseBodyConverter <T> implements Converter<ResponseBody, T> {

    private final Class<T> cls;
    private final XStream xStream;

    XStreamXmlResponseBodyConverter(Class<T> cls, XStream xStream) {
        this.cls = cls;
        this.xStream = xStream;
    }

    @Override
    public T convert(ResponseBody value) throws IOException {

        try {

            this.xStream.processAnnotations(cls);
            Object object =  this.xStream.fromXML(value.byteStream());
            return (T) object;

        }finally {
            value.close();
        }
    }
}
```

Crea una clase para manejar la respuesta del cuerpo.

```
final class XStreamXmlRequestBodyConverter<T> implements Converter<T, RequestBody> {
```

```
    private static final MediaType MEDIA_TYPE = MediaType.parse("application/xml; charset=UTF-
8");
    private static final String CHARSET = "UTF-8";

    private final XStream xStream;

    XStreamXmlRequestBodyConverter(XStream xStream) {
        this.xStream = xStream;
    }

    @Override
    public RequestBody convert(T value) throws IOException {

        Buffer buffer = new Buffer();

        try {
            OutputStreamWriter osw = new OutputStreamWriter(buffer.outputStream(), CHARSET);
            xStream.toXML(value, osw);
            osw.flush();
        } catch (Exception e) {
            throw new RuntimeException(e);
        }

        return RequestBody.create(MEDIA_TYPE, buffer.readByteString());
    }
}
```

Entonces, este punto podemos enviar y recibir cualquier XML, solo necesitamos crear anotaciones de XStream para las entidades.

Luego crea una instancia de Retrofit:

```
XStream xs = new XStream(new DomDriver());
xs.autodetectAnnotations(true);

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com/")
    .addConverterFactory(XStreamXmlConverterFactory.create(xs))
    .client(client)
    .build();
```

Una simple solicitud POST con GSON

Muestra JSON:

```
{
    "id": "12345",
    "type": "android"
}
```

Defina su solicitud:

```
public class GetDeviceRequest {

    @SerializedName("deviceId")
```

```
    private String mDeviceId;

    public GetDeviceRequest(String deviceId) {
        this.mDeviceId = deviceId;
    }

    public String getDeviceId() {
        return mDeviceId;
    }

}
```

Defina su servicio (puntos finales para golpear):

```
public interface Service {

    @POST("device")
    Call<Device> getDevice(@Body GetDeviceRequest getDeviceRequest);

}
```

Defina su instancia singleton del cliente de red:

```
public class RestClient {

    private static Service REST_CLIENT;

    static {
        setupRestClient();
    }

    private static void setupRestClient() {

        // Define gson
        Gson gson = new Gson();

        // Define our client
        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl("http://example.com/")
                .addConverterFactory(GsonConverterFactory.create(gson))
                .build();

        REST_CLIENT = retrofit.create(Service.class);
    }

    public static Retrofit getRestClient() {
        return REST_CLIENT;
    }

}
```

Defina un objeto modelo simple para el dispositivo:

```
public class Device {

    @SerializedName("id")
    private String mId;
```

```
    @SerializedName("type")
    private String mType;

    public String getId() {
        return mId;
    }

    public String getType() {
        return mType;
    }

}
```

Definir controlador para manejar las solicitudes del dispositivo.

```
public class DeviceController {

    // Other initialization code here...

    public void getDeviceFromAPI() {

        // Define our request and enqueue
        Call<Device> call = RestClient.getRestClient().getDevice(new
GetDeviceRequest("12345"));

        // Go ahead and enqueue the request
        call.enqueue(new Callback<Device>() {
            @Override
            public void onSuccess(Response<Device> deviceResponse) {
                // Take care of your device here
                if (deviceResponse.isSuccess()) {
                    // Handle success
                    //delegate.passDeviceObject();
                }
            }

            @Override
            public void onFailure(Throwable t) {
                // Go ahead and handle the error here
            }

        });
```

Leyendo la URL del formulario XML con Retrofit 2

Usaremos retrofit 2 y SimpleXmlConverter para obtener datos xml de url y analizar a la clase Java.

Agregue dependencia al script Gradle:

```
compile 'com.squareup.retrofit2:retrofit:2.1.0'
compile 'com.squareup.retrofit2:converter-simplexml:2.1.0'
```

Crear interfaz

También cree un contenedor de clase XML en nuestro caso. Clase Rss

```
    public interface ApiDataInterface{

        // path to xml link on web site

        @GET (data/read.xml)

        Call<Rss> getData();

}
```

Función de lectura xml

```
private void readXmlFeed() {
        try {

            // base url - url of web site
            Retrofit retrofit = new Retrofit.Builder()
                    .baseUrl(http://www.google.com/)
                    .client(new OkHttpClient())
                    .addConverterFactory(SimpleXmlConverterFactory.create())
                    .build();

            ApiDataInterface apiService = retrofit.create(ApiDataInterface.class);

            Call<Rss> call = apiService.getData();
            call.enqueue(new Callback<Rss>() {

                @Override
                public void onResponse(Call<Rss> call, Response<Rss> response) {

                    Log.e("Response success", response.message());

                }

                @Override
                public void onFailure(Call<Rss> call, Throwable t) {
                    Log.e("Response fail", t.getMessage());
                }
            });

        } catch (Exception e) {
            Log.e("Exception", e.getMessage());
        }


    }
```

Este es un ejemplo de clase Java con anotaciones SimpleXML.

Más sobre anotaciones SimpleXmlDocumentation

```
@Root (name = "rss")

public class Rss
{
```

```
    public Rss() {

    }



    public Rss(String title, String description, String link, List<Item> item, String
language) {

        this.title = title;
        this.description = description;
        this.link = link;
        this.item = item;
        this.language = language;

    }

    @Element (name = "title")
    private String title;

    @Element(name = "description")
    private String description;

    @Element(name = "link")
    private String link;

    @ElementList (entry="item", inline=true)
    private List<Item> item;

    @Element(name = "language")
    private String language;
```

Lea Retrofit2 en línea: https://riptutorial.com/es/android/topic/1132/retrofit2

Capítulo 218: Retrofit2 con RxJava

Examples

Retrofit2 con RxJava

Primero, agregue dependencias relevantes en el archivo build.gradle.

```
dependencies {
    ....
    compile 'com.squareup.retrofit2:retrofit:2.3.0'
    compile 'com.squareup.retrofit2:converter-gson:2.3.0'
    compile 'com.squareup.retrofit2:adapter-rxjava:2.3.0'
    ....
}
```

Luego crea el modelo que te gustaría recibir:

```
public class Server {
    public String name;
    public String url;
    public String apikey;
    public List<Site> siteList;
}
```

Cree una interfaz que contenga los métodos utilizados para intercambiar datos con el servidor remoto:

```
public interface ApiServerRequests {

    @GET("api/get-servers")
    public Observable<List<Server>> getServers();
}
```

Luego crea una instancia de `Retrofit` :

```
public ApiRequests DeviceAPIHelper ()
{
    Gson gson = new GsonBuilder().create();

    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("http://example.com/")
            .addConverterFactory(GsonConverterFactory.create(gson))
            .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
            .build();

    api = retrofit.create(ApiServerRequests.class);
    return api;
}
```

Luego, en cualquier parte del código, llame al método:

```
apiRequests.getServers()
   .subscribeOn(Schedulers.io()) // the observable is emitted on io thread
   .observerOn(AndroidSchedulers.mainThread()) // Methods needed to handle request in
background thread
   .subscribe(new Subscriber<List<Server>>() {
       @Override
       public void onCompleted() {

       }

       @Override
       public void onError(Throwable e) {

       }

       @Override
       public void onNext(List<Server> servers) {
           //A list of servers is fetched successfully
       }
   });
```

Reequipamiento con RxJava para obtener datos de forma asíncrona

Desde el repositorio de GitHub de RxJava, *RxJava es una implementación de Java VM de Reactive Extensions: una biblioteca para componer programas asíncronos y basados en eventos mediante el uso de secuencias observables. Extiende el patrón de observador para admitir secuencias de datos / eventos y agrega operadores que le permiten componer secuencias de forma declarativa al tiempo que abstrae preocupaciones sobre cosas como subprocesos de bajo nivel, sincronización, seguridad de subprocesos y estructuras de datos concurrentes.*

Retrofit es un cliente HTTP seguro para el tipo para Android y Java. Con esto, los desarrolladores pueden hacer que todo lo relacionado con la red sea mucho más fácil. Como ejemplo, vamos a descargar algunos JSON y los mostraremos en RecyclerView como una lista.

Empezando:

Agregue las dependencias de RxJava, RxAndroid y Retrofit en el archivo build.gradle de su nivel de aplicación: `compile "io.reactivex:rxjava:1.1.6"`

```
compile "io.reactivex:rxandroid:1.2.1"
compile "com.squareup.retrofit2:adapter-rxjava:2.0.2"
compile "com.google.code.gson:gson:2.6.2"
compile "com.squareup.retrofit2:retrofit:2.0.2"
compile "com.squareup.retrofit2:converter-gson:2.0.2"
```

Defina ApiClient y ApiInterface para intercambiar datos desde el servidor

```
public class ApiClient {

private static Retrofit retrofitInstance = null;
private static final String BASE_URL = "https://api.github.com/";

public static Retrofit getInstance() {
    if (retrofitInstance == null) {
        retrofitInstance = new Retrofit.Builder()
```

```
                .baseUrl(BASE_URL)
                .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
                .addConverterFactory(GsonConverterFactory.create())
                .build();
    }
    return retrofitInstance;
}

public static <T> T createRetrofitService(final Class<T> clazz, final String endPoint) {
    final Retrofit restAdapter = new Retrofit.Builder()
            .baseUrl(endPoint)
            .build();

    return restAdapter.create(clazz);
}

public static String getBaseUrl() {
    return BASE_URL;
}}
```

interfaz pública ApiInterface {

```
@GET("repos/{org}/{repo}/issues")
Observable<List<Issue>> getIssues(@Path("org") String organisation,
                                  @Path("repo") String repositoryName,
                                  @Query("page") int pageNumber);}
```

Tenga en cuenta que getRepos () está devolviendo un Observable y no solo una lista de problemas.

Define los modelos

Se muestra un ejemplo para esto. Puedes usar servicios gratuitos como JsonSchema2Pojo o este.

```
public class Comment {

@SerializedName("url")
@Expose
private String url;
@SerializedName("html_url")
@Expose
private String htmlUrl;

//Getters and Setters
}
```

Crear instancia de Retrofit

```
ApiInterface apiService = ApiClient.getInstance().create(ApiInterface.class);
```

Luego, use esta instancia para obtener datos del servidor

```
Observable<List<Issue>> issueObservable = apiService.getIssues(org, repo,
pageNumber);
    issueObservable.subscribeOn(Schedulers.newThread())
```

```
            .observeOn(AndroidSchedulers.mainThread())
            .map(issues -> issues)    //get issues and map to issues list
            .subscribe(new Subscriber<List<Issue>>() {
                @Override
                public void onCompleted() {
                    Log.i(TAG, "onCompleted: COMPLETED!");
                }

                @Override
                public void onError(Throwable e) {
                    Log.e(TAG, "onError: ", e);
                }

                @Override
                public void onNext(List<Issue> issues) {
                    recyclerView.setAdapter(new IssueAdapter(MainActivity.this, issues,
apiService));
                }
            });
```

Ahora, ha obtenido con éxito datos de un servidor utilizando Retrofit y RxJava.

Ejemplo de solicitudes anidadas: solicitudes múltiples, combinar resultados

Supongamos que tenemos una API que nos permite obtener metadatos de objetos en una solicitud única (`getAllPets`) y otra solicitud que tiene datos completos de un solo recurso (`getSinglePet`). ¿Cómo podemos consultarlos todos en una sola cadena?

```
public class PetsFetcher {


static class PetRepository {
    List<Integer> ids;
}

static class Pet {
    int id;
    String name;
    int weight;
    int height;
}

interface PetApi {

    @GET("pets") Observable<PetRepository> getAllPets();

    @GET("pet/{id}") Observable<Pet> getSinglePet(@Path("id") int id);

}

PetApi petApi;

Disposable petsDisposable;

public void requestAllPets() {

    petApi.getAllPets()
            .doOnSubscribe(new Consumer<Disposable>() {
```

```
                @Override public void accept(Disposable disposable) throws Exception {
                    petsDisposable = disposable;
                }
            })
            .flatMap(new Function<PetRepository, ObservableSource<Integer>>() {
                @Override
                public ObservableSource<Integer> apply(PetRepository petRepository) throws
Exception {
                    List<Integer> petIds = petRepository.ids;
                    return Observable.fromIterable(petIds);
                }
            })
            .flatMap(new Function<Integer, ObservableSource<Pet>>() {
                @Override public ObservableSource<Pet> apply(Integer id) throws Exception {
                    return petApi.getSinglePet(id);
                }
            })
            .toList()
            .toObservable()
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(new Consumer<List<Pet>>() {
                @Override public void accept(List<Pet> pets) throws Exception {
                    //use your pets here
                }
            }, new Consumer<Throwable>() {
                @Override public void accept(Throwable throwable) throws Exception {
                    //show user something goes wrong
                }
            });

}

void cancelRequests(){
    if (petsDisposable!=null){
        petsDisposable.dispose();
        petsDisposable = null;
    }
}
```

}

Lea Retrofit2 con RxJava en línea: https://riptutorial.com/es/android/topic/7632/retrofit2-con-rxjava

Capítulo 219: RoboGuice

Examples

Ejemplo simple

RoboGuice es un marco que aporta la simplicidad y facilidad de inyección de dependencia a Android, utilizando la propia biblioteca Guice de Google.

```
@ContentView(R.layout.main)
class RoboWay extends RoboActivity {
    @InjectView(R.id.name)              TextView name;
    @InjectView(R.id.thumbnail)         ImageView thumbnail;
    @InjectResource(R.drawable.icon)    Drawable icon;
    @InjectResource(R.string.app_name)  String myName;
    @Inject                             LocationManager loc;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        name.setText( "Hello, " + myName );
    }
}
```

Instalación para proyectos Gradle

Agregue el siguiente pom a la sección de dependencias de su archivo de compilación de gradle:

```
project.dependencies {
    compile 'org.roboguice:roboguice:3.+'
    provided 'org.roboguice:roboblender:3.+'
}
```

@ContentView anotación

La anotación @ContentView se puede usar para aliviar aún más el desarrollo de actividades y reemplazar la declaración setContentView:

```
@ContentView(R.layout.myactivity_layout)
public class MyActivity extends RoboActivity {
    @InjectView(R.id.text1) TextView textView;

    @Override
    protected void onCreate( Bundle savedState ) {
        textView.setText("Hello!");
    }
}
```

@InjectResource anotación

Puede inyectar cualquier tipo de recurso, cadenas, animaciones, dibujos, etc.

Para inyectar su primer recurso en una actividad, deberá:

- Heredar de RoboActivity
- Anota tus recursos con @InjectResource

Ejemplo

```
@InjectResource(R.string.app_name) String name;

@InjectResource(R.drawable.ic_launcher) Drawable icLauncher;

@InjectResource(R.anim.my_animation) Animation myAnimation;
```

@InjectView anotación

Puedes inyectar cualquier vista usando la anotación @InjectView:

Tendrá que:

- Heredar de RoboActivity
- Establece tu vista de contenido
- Anota tus vistas con @InjectView

Ejemplo

```
@InjectView(R.id.textView1) TextView textView1;

@InjectView(R.id.textView2) TextView textView2;

@InjectView(R.id.imageView1) ImageView imageView1;
```

Introducción a RoboGuice

`RoboGuice` es un marco que aporta la simplicidad y facilidad de inyección de dependencia a Android, utilizando la propia biblioteca Guice de Google.

RoboGuice 3 reduce su código de aplicación. Menos código significa menos oportunidades para los errores. También hace que su código sea más fácil de seguir: ya no está su código lleno de la mecánica de la plataforma Android, pero ahora puede centrarse en la lógica real única de su aplicación.

Para darle una idea, echar un vistazo a este sencillo ejemplo de una típica Android `Activity` :

```
class AndroidWay extends Activity {
        TextView name;
        ImageView thumbnail;
        LocationManager loc;
        Drawable icon;
        String myName;

        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
```

```
            setContentView(R.layout.main);
            name      = (TextView) findViewById(R.id.name);
            thumbnail = (ImageView) findViewById(R.id.thumbnail);
            loc       = (LocationManager) getSystemService(Activity.LOCATION_SERVICE);
            icon      = getResources().getDrawable(R.drawable.icon);
            myName    = getString(R.string.app_name);
            name.setText( "Hello, " + myName );
        }
    }
```

Este ejemplo es de 19 líneas de código. Si está intentando leer en `onCreate()` , debe omitir más de 5 líneas de inicialización repetitiva para encontrar la única que realmente importa: `name.setText()` . Y las actividades complejas pueden terminar con mucho más de este tipo de código de inicialización.

Compare esto con la misma aplicación, escrita con `RoboGuice` :

```
@ContentView(R.layout.main)
   class RoboWay extends RoboActivity {
        @InjectView(R.id.name)             TextView name;
        @InjectView(R.id.thumbnail)        ImageView thumbnail;
        @InjectResource(R.drawable.icon)   Drawable icon;
        @InjectResource(R.string.app_name) String myName;
        @Inject                            LocationManager loc;

        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            name.setText( "Hello, " + myName );
        }
    }
```

El objetivo de RoboGuice es hacer que su código sea sobre su aplicación, en lugar de sobre todo el código de inicialización y ciclo de vida que normalmente tiene que mantener en Android.

Anotaciones:

@ContentView anotación:

La anotación @ContentView se puede usar para aliviar aún más el desarrollo de actividades y reemplazar la declaración setContentView:

```
@ContentView(R.layout.myactivity_layout)
    public class MyActivity extends RoboActivity {
        @InjectView(R.id.text1) TextView textView;

        @Override
        protected void onCreate( Bundle savedState ) {
            textView.setText("Hello!");
        }
    }
```

@InjectResource anotación:

Primero necesitas una Actividad que hereda de RoboActivity. Luego, asumiendo que tiene una

animación mi_animación.xml en su carpeta res / anim, ahora puede hacer referencia a ella con una anotación:

```
public class MyActivity extends RoboActivity {
    @InjectResource(R.anim.my_animation) Animation myAnimation;
    // the rest of your code
}
```

@ Anotación en el proyecto:

Usted se asegura de que su actividad se extienda desde RoboActivity y haga anotaciones en su miembro del servicio del sistema con @Inject. Roboguice hará el resto.

```
class MyActivity extends RoboActivity {
    @Inject Vibrator vibrator;
    @Inject NotificationManager notificationManager;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // we can use the instances directly!
        vibrator.vibrate(1000L); // RoboGuice took care of the
getSystemService(VIBRATOR_SERVICE)
        notificationManager.cancelAll();
```

Además de las Vistas, los Recursos, los Servicios y otras cosas específicas de Android, RoboGuice puede inyectar objetos Java antiguos y sencillos. Por defecto, Roboguice llamará a un constructor sin argumentos en su POJO

```
class MyActivity extends RoboActivity {
    @Inject Foo foo; // this will basically call new Foo();
}
```

Lea RoboGuice en línea: https://riptutorial.com/es/android/topic/2563/roboguice

Capítulo 220: Robolectric

Introducción

La prueba unitaria consiste en tomar un código y probarlo de forma independiente sin otras dependencias o partes del sistema en ejecución (por ejemplo, la base de datos).

Robolectric es un marco de prueba unitaria que destruye el frasco del SDK de Android para que puedas probar el desarrollo de tu aplicación Android. Las pruebas se ejecutan dentro de la JVM en su estación de trabajo en segundos.

Combinarlos a ambos le permite ejecutar pruebas rápidas en el JVN aún utilizando las API de Android.

Examples

Prueba robolectrica

```
@RunWith(RobolectricTestRunner.class)
public class MyActivityTest {

  @Test
  public void clickingButton_shouldChangeResultsViewText() throws Exception {
    MyActivity activity = Robolectric.setupActivity(MyActivity.class);

    Button button = (Button) activity.findViewById(R.id.button);
    TextView results = (TextView) activity.findViewById(R.id.results);

    button.performClick();
    assertThat(results.getText().toString()).isEqualTo("Robolectric Rocks!");
  }
}
```

Configuración

Para configurar robolectric, agregue la anotación `@Config` a la clase o método de prueba.

Ejecutar con clase de aplicación personalizada

```
@RunWith(RobolectricTestRunner.class)
@Config(application = MyApplication.class)
public final class MyTest {
}
```

Establecer objetivo SDK

```
@RunWith(RobolectricTestRunner.class)
```

```
@Config(sdk = Build.VERSION_CODES.LOLLIPOP)
public final class MyTest {
}
```

Ejecutar con manifiesto personalizado

Cuando se especifique, robolectric se verá relativo al directorio actual. El valor predeterminado es `AndroidManifest.xml`

Los recursos y los activos se cargarán en relación con el manifiesto.

```
@RunWith(RobolectricTestRunner.class)
@Config(manifest = "path/AndroidManifest.xml")
public final class MyTest {
}
```

Usar calificadores

Posibles calificadores se pueden encontrar en documentos de Android .

```
@RunWith(RobolectricTestRunner.class)
public final class MyTest {

    @Config(qualifiers = "sw600dp")
    public void testForTablet() {
    }
}
```

Lea Robolectric en línea: https://riptutorial.com/es/android/topic/8743/robolectric

Capítulo 221: SearchView

Examples

AppCompat SearchView con el observador de RxBindings

build.gradle :

```
dependencies {
    compile 'com.android.support:appcompat-v7:23.3.0'
    compile 'com.jakewharton.rxbinding:rxbinding-appcompat-v7:0.4.0'
}
```

menu / menu.xml :

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/action_search" android:title="Search"
        android:icon="@android:drawable/ic_menu_search"
        app:actionViewClass="android.support.v7.widget.SearchView"
        app:showAsAction="always"/>
</menu>
```

MainActivity.java :

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);

    MenuItem searchMenuItem = menu.findItem(R.id.action_search);
    setupSearchView(searchMenuItem );

    return true;
}

private void setupSearchView(MenuItem searchMenuItem) {
    SearchView searchView = (SearchView) searchMenuItem.getActionView();
    searchView.setQueryHint(getString(R.string.search_hint)); // your hint here

    SearchAdapter searchAdapter = new SearchAdapter(this);
    searchView.setSuggestionsAdapter(searchAdapter);

    // optional: set the letters count after which the search will begin to 1
    // the default is 2
    try {
        int autoCompleteTextViewID =
getResources().getIdentifier("android:id/search_src_text", null, null);
        AutoCompleteTextView searchAutoCompleteTextView = (AutoCompleteTextView)
searchView.findViewById(autoCompleteTextViewID);
        searchAutoCompleteTextView.setThreshold(1);
    } catch (Exception e) {
        Logs.e(TAG, "failed to set search view letters threshold");
```

```
    }

    searchView.setOnSearchClickListener(v -> {
        // optional actions to search view expand
    });
    searchView.setOnCloseListener(() -> {
        // optional actions to search view close
        return false;
    });

    RxSearchView.queryTextChanges(searchView)
            .doOnEach(notification -> {
                CharSequence query = (CharSequence) notification.getValue();
                searchAdapter.filter(query);
            })
            .debounce(300, TimeUnit.MILLISECONDS) // to skip intermediate letters
            .flatMap(query -> MyWebService.search(query)) // make a search request
            .retry(3)
            .subscribe(results -> {
                searchAdapter.populateAdapter(results);
            });

    //optional: collapse the searchView on close
    searchView.setOnQueryTextFocusChangeListener((view, queryTextFocused) -> {
        if (!queryTextFocused) {
            collapseSearchView();
        }
    });
}
```

SearchAdapter.java

```
public class SearchAdapter extends CursorAdapter {
    private List<SearchResult> items = Collections.emptyList();

    public SearchAdapter(Activity activity) {
        super(activity, null, CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
    }

    public void populateAdapter(List<SearchResult> items) {
        this.items = items;
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            c.addRow(new Object[]{i});
        }
        changeCursor(c);
        notifyDataSetChanged();
    }

    public void filter(CharSequence query) {
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            SearchResult result = items.get(i);
            if (result.getText().startsWith(query.toString())) {
                c.addRow(new Object[]{i});
            }
        }
        changeCursor(c);
        notifyDataSetChanged();
    }
```

```
    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        ViewHolder holder = (ViewHolder) view.getTag();
        int position = cursor.getPosition();
        if (position < items.size()) {
            SearchResult result = items.get(position);
            // bind your view here
        }
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        LayoutInflater inflater = (LayoutInflater) context
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        View v = inflater.inflate(R.layout.search_list_item, parent, false);
        ViewHolder holder = new ViewHolder(v);

        v.setTag(holder);
        return v;
    }

    private static class ViewHolder {
        public final TextView text;

        public ViewHolder(View v) {
            this.text= (TextView) v.findViewById(R.id.text);
        }
    }
}
```

SearchView en la barra de herramientas con fragmento

menu.xml - (`res` -> `menu`)

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".HomeActivity">

    <item
        android:id="@+id/action_search"
        android:icon="@android:drawable/ic_menu_search"
        android:title="Search"
        app:actionViewClass="android.support.v7.widget.SearchView"
        app:showAsAction="always" />

</menu>
```

MainFragment.java

```
public class MainFragment extends Fragment {

    private SearchView searchView = null;
    private SearchView.OnQueryTextListener queryTextListener;

    @Nullable
```

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        return inflater.inflate(R.layout.fragment_main, container, false);
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setHasOptionsMenu(true);
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        inflater.inflate(R.menu.menu, menu);
        MenuItem searchItem = menu.findItem(R.id.action_search);
        SearchManager searchManager = (SearchManager)
getActivity().getSystemService(Context.SEARCH_SERVICE);

        if (searchItem != null) {
            searchView = (SearchView) searchItem.getActionView();
        }
        if (searchView != null) {

searchView.setSearchableInfo(searchManager.getSearchableInfo(getActivity().getComponentName()));


            queryTextListener = new SearchView.OnQueryTextListener() {
                @Override
                public boolean onQueryTextChange(String newText) {
                    Log.i("onQueryTextChange", newText);

                    return true;
                }
                @Override
                public boolean onQueryTextSubmit(String query) {
                    Log.i("onQueryTextSubmit", query);

                    return true;
                }
            };
            searchView.setOnQueryTextListener(queryTextListener);
        }
        super.onCreateOptionsMenu(menu, inflater);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_search:
                // Not implemented here
                return false;
            default:
                break;
        }
        searchView.setOnQueryTextListener(queryTextListener);
        return super.onOptionsItemSelected(item);
    }
}
```

Captura de pantalla de referencia:



Hello Android

`menu.xml` , debemos entender que depende completamente del estilo aplicado a la barra de herramientas subyacente. Para lograr el tema de la barra de herramientas, aplique los siguientes pasos.

Crea un estilo en el `styles.xml`

```
<style name="ActionBarThemeOverlay">
     <item name="android:textColorPrimary">@color/prim_color</item>
     <item name="colorControlNormal">@color/normal_color</item>
     <item name="colorControlHighlight">@color/high_color</item>
     <item name="android:textColorHint">@color/hint_color</item>
</style>
```

Aplicar el estilo a la barra de herramientas.

```
<android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        app:theme="@style/ActionBarThemeOverlay"
        app:popupTheme="@style/ActionBarThemeOverlay"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="@color/colorPrimary"
        android:title="@string/title"
        tools:targetApi="m" />
```

Esto le da el color deseado a todas las vistas correspondientes a la barra de herramientas (botón Atrás, iconos de menú y SearchView).

Lea SearchView en línea: https://riptutorial.com/es/android/topic/4786/searchview

Capítulo 222: Secure SharedPreferences

Introducción

Las preferencias compartidas son **archivos XML basados en valores clave** . Se encuentra en `/data/data/package_name/shared_prefs/<filename.xml>` .

Por lo tanto, un usuario con privilegios de root puede navegar a esta ubicación y puede cambiar sus valores. Si desea proteger los valores en sus preferencias compartidas, puede escribir un mecanismo simple de cifrado y descifrado.

Debe tener en cuenta que las Preferencias Compartidas nunca fueron diseñadas para ser seguras, es solo una forma simple de conservar los datos.

Sintaxis

1. Cifrado de cadena estática pública (entrada de cadena);
2. Descifrado de cadenas públicas estáticas (entrada de cadenas);

Parámetros

Parámetro	Definición
entrada	Valor de cadena para cifrar o descifrar.

Observaciones

Las preferencias compartidas nunca se crearon para ser seguras, es solo una forma simple de conservar los datos.

No es una buena idea usar preferencias compartidas para almacenar información crítica, como las credenciales de usuario. Para guardar las credenciales de usuario (como las contraseñas), debe utilizar otros métodos, como el `AccountManager` de Android.

Examples

Asegurar una preferencia compartida

Codec simple

Aquí, para ilustrar el principio de funcionamiento, podemos usar cifrado y descifrado simples de la siguiente manera.

```
public static String encrypt(String input) {
    // Simple encryption, not very strong!
    return Base64.encodeToString(input.getBytes(), Base64.DEFAULT);
}

public static String decrypt(String input) {
    return new String(Base64.decode(input, Base64.DEFAULT));
}
```

Técnica de Implementación

```
public static String pref_name = "My_Shared_Pref";

// To Write
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
SharedPreferences.Editor editor = preferences.edit();
editor.putString(encrypt("password"), encrypt("my_dummy_pass"));
editor.apply(); // Or commit if targeting old devices

// To Read
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
String passEncrypted = preferences.getString(encrypt("password"), encrypt("default_value"));
String password = decrypt(passEncrypted);
```

Lea Secure SharedPreferences en línea: https://riptutorial.com/es/android/topic/9887/secure-sharedpreferences

Capítulo 223: Secure SharedPreferences

Introducción

Las preferencias compartidas son **archivos XML basados en valores clave** . Se encuentra en / data / data / package_name / shared_prefs / <filename.xml>.

Por lo tanto, un usuario con privilegios de root puede navegar a esta ubicación y puede cambiar sus valores. Si desea proteger los valores en sus preferencias compartidas, puede escribir un mecanismo simple de cifrado y descifrado.

Debe tener en cuenta que las Preferencias Compartidas nunca fueron diseñadas para ser seguras, es solo una forma simple de conservar los datos.

Sintaxis

1. Cifrado de cadena estática pública (entrada de cadena);
2. Descifrado de cadenas públicas estáticas (entrada de cadenas);

Parámetros

Parámetro	Definición
entrada	Valor de cadena para cifrar o descifrar.

Observaciones

Las preferencias compartidas nunca se crearon para ser seguras, es solo una forma simple de conservar los datos.

No es una buena idea usar preferencias compartidas para almacenar información crítica, como las credenciales de usuario. Para guardar las credenciales de usuario (como las contraseñas), debe utilizar otros métodos, como el `AccountManager` de Android.

Examples

Asegurar una preferencia compartida

Codec simple

Aquí, para ilustrar el principio de funcionamiento, podemos usar cifrado y descifrado simples de la siguiente manera.

```
public static String encrypt(String input) {
    // Simple encryption, not very strong!
    return Base64.encodeToString(input.getBytes(), Base64.DEFAULT);
}

public static String decrypt(String input) {
    return new String(Base64.decode(input, Base64.DEFAULT));
}
```

Técnica de Implementación

```
public static String pref_name = "My_Shared_Pref";

// To Write
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
SharedPreferences.Editor editor = preferences.edit();
editor.putString(encrypt("password"), encrypt("my_dummy_pass"));
editor.apply(); // Or commit if targeting old devices

// To Read
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
String passEncrypted = preferences.getString(encrypt("password"), encrypt("default_value"));
String password = decrypt(passEncrypted);
```

Lea Secure SharedPreferences en línea: https://riptutorial.com/es/android/topic/9890/secure-sharedpreferences

Capítulo 224: Seguridad

Examples

Verificación de la firma de la aplicación - Detección de sabotaje

Esta técnica detalla cómo asegurarse de que su .apk haya sido firmado con su certificado de desarrollador, y aprovecha el hecho de que el certificado permanece consistente y que solo usted tiene acceso a él. Podemos dividir esta técnica en 3 simples pasos:

- Encuentra tu firma de certificado de desarrollador.
- Inserta tu firma en una constante de cadena en tu aplicación.
- Verifique que la firma en tiempo de ejecución coincida con nuestra firma incorporada de desarrollador.

Aquí está el fragmento de código:

```
private static final int VALID = 0;
private static final int INVALID = 1;

public static int checkAppSignature(Context context) {

try {
      PackageInfo packageInfo =
context.getPackageManager().getPackageInfo(context.getPackageName(),
PackageManager.GET_SIGNATURES);

      for (Signature signature : packageInfo.signatures) {

        byte[] signatureBytes = signature.toByteArray();

        MessageDigest md = MessageDigest.getInstance("SHA");

        md.update(signature.toByteArray());

        final String currentSignature = Base64.encodeToString(md.digest(), Base64.DEFAULT);

        Log.d("REMOVE_ME", "Include this string as a value for SIGNATURE:" +
currentSignature);

        //compare signatures
        if (SIGNATURE.equals(currentSignature)){
          return VALID;
        };
      }
    } catch (Exception e) {
        //assumes an issue in checking signature., but we let the caller decide on what to do.
    }

    return INVALID;

}
```

Lea Seguridad en línea: https://riptutorial.com/es/android/topic/4664/seguridad

Capítulo 225: SensorManager

Examples

Recuperando eventos del sensor

Recuperando información del sensor de los sensores a bordo:

```
public class MainActivity extends Activity implements SensorEventListener {

    private SensorManager mSensorManager;
    private Sensor accelerometer;
    private Sensor gyroscope;

    float[] accelerometerData = new float[3];
    float[] gyroscopeData = new float[3];

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

        accelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        gyroscope = mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);

    }

    @Override
    public void onResume() {
        //Register listeners for your sensors of interest
        mSensorManager.registerListener(this, accelerometer,
SensorManager.SENSOR_DELAY_FASTEST);
        mSensorManager.registerListener(this, gyroscope, SensorManager.SENSOR_DELAY_FASTEST);
        super.onResume();
    }

    @Override
    protected void onPause() {
        //Unregister any previously registered listeners
        mSensorManager.unregisterListener(this);
        super.onPause();
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        //Check the type of sensor data being polled and store into corresponding float array
        if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
            accelerometerData = event.values;
        } else if (event.sensor.getType() == Sensor.TYPE_GYROSCOPE) {
            gyroscopeData = event.values;
        }
    }

    @Override
```

```
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // TODO Auto-generated method stub
    }
}
```

Transformación del sensor al sistema de coordenadas del mundo.

Los valores de los sensores devueltos por Android corresponden al sistema de coordenadas del teléfono (por ejemplo, + Y apunta hacia la parte superior del teléfono). Podemos transformar estos valores de sensor en un sistema de coordenadas mundo (por ejemplo, + Y apunta hacia el Norte magnético, tangencial al suelo) usando la matriz de rotación de los administradores de sensores

Primero, deberá declarar e inicializar las matrices / matrices donde se almacenarán los datos (puede hacer esto en el método `onCreate` , por ejemplo):

```
float[] accelerometerData = new float[3];
float[] accelerometerWorldData = new float[3];
float[] gravityData = new float[3];
float[] magneticData = new float[3];
float[] rotationMatrix = new float[9];
```

Luego, debemos detectar los cambios en los valores de los sensores, almacenarlos en los arreglos correspondientes (si queremos usarlos más adelante / en otro lugar), luego calcular la matriz de rotación y la transformación resultante en coordenadas mundiales:

```
public void onSensorChanged(SensorEvent event) {
    sensor = event.sensor;
    int i = sensor.getType();

    if (i == Sensor.TYPE_ACCELEROMETER) {
        accelerometerData = event.values;
    } else if (i == Sensor.TYPE_GRAVITY) {
        gravityData = event.values;
    } else if (i == Sensor.TYPE_MAGNETIC) {
        magneticData = event.values;
    }

    //Calculate rotation matrix from gravity and magnetic sensor data
    SensorManager.getRotationMatrix(rotationMatrix, null, gravityData, magneticData);

    //World coordinate system transformation for acceleration
    accelerometerWorldData[0] = rotationMatrix[0] * accelerometerData[0] + rotationMatrix[1] *
accelerometerData[1] + rotationMatrix[2] * accelerometerData[2];
    accelerometerWorldData[1] = rotationMatrix[3] * accelerometerData[0] + rotationMatrix[4] *
accelerometerData[1] + rotationMatrix[5] * accelerometerData[2];
    accelerometerWorldData[2] = rotationMatrix[6] * accelerometerData[0] + rotationMatrix[7] *
accelerometerData[1] + rotationMatrix[8] * accelerometerData[2];

}
```

Decide si tu dispositivo es estático o no, usando el acelerómetro

Agregue el siguiente código al `onCreate()` / `onResume()` :

```
SensorManager sensorManager;
Sensor mAccelerometer;
final float movementThreshold = 0.5f;  // You may have to change this value.
boolean isMoving = false;
float[] prevValues = {1.0f, 1.0f, 1.0f};
float[] currValues = new float[3];

sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
mAccelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
sensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
```

Es posible que tenga que ajustar la sensibilidad adaptando el umbral de `movementThreshold` por prueba y error. Luego, anule el método `onSensorChanged()` siguiente manera:

```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor == mAccelerometer) {
        System.arraycopy(event.values, 0, currValues, 0, event.values.length);
        if ((Math.abs(currValues[0] - prevValues[0]) > movementThreshold) ||
                (Math.abs(currValues[1] - prevValues[1]) > movementThreshold) ||
                (Math.abs(currValues[2] - prevValues[2]) > movementThreshold)) {
            isMoving = true;
        } else {
            isMoving = false;
        }
        System.arraycopy(currValues, 0, prevValues, 0, currValues.length);
    }
}
```

Si desea evitar que su aplicación se instale en dispositivos que no tienen un acelerómetro, debe agregar la siguiente línea a su manifiesto:

```
<uses-feature android:name="android.hardware.sensor.accelerometer" />
```

Lea SensorManager en línea: https://riptutorial.com/es/android/topic/3344/sensormanager

Capítulo 226: Servicio

Introducción

Un servicio se ejecuta en **segundo plano** para realizar operaciones de larga ejecución o para realizar trabajos para procesos remotos. Un servicio no proporciona ninguna interfaz de usuario que se ejecuta solo en segundo plano con la entrada del usuario. Por ejemplo, un servicio puede reproducir música en segundo plano mientras el usuario está en una aplicación diferente, o puede descargar datos de Internet sin bloquear la interacción del usuario con el dispositivo Android.

Observaciones

Si no ha definido su servicio en su AndroidManifest.xml, recibirá una `ServiceNotFoundException` cuando intente iniciarlo.

Nota:

Para obtener información sobre `IntentService` , consulte aquí: Ejemplo de IntentService

Examples

Comenzando un servicio

Comenzar un servicio es muy fácil, solo llame a `startService` con una intención, desde dentro de una Actividad:

```
Intent intent = new Intent(this, MyService.class);  //substitute MyService with the name of
your service
intent.putExtra(Intent.EXTRA_TEXT, "Some text"); //add any extra data to pass to the service

startService(intent); //Call startService to start the service.
```

Ciclo de vida de un servicio

El ciclo de vida de los servicios tiene las siguientes devoluciones de llamada

- `onCreate()` :

Se ejecuta cuando el servicio se crea por primera vez para configurar las configuraciones iniciales que pueda necesitar. Este método se ejecuta solo si el servicio aún no se está ejecutando.

- `onStartCommand()` :

Ejecutado cada vez que `startService()` es invocado por otro componente, como Activity o BroadcastReceiver. Cuando use este método, el Servicio se ejecutará hasta que llame a `stopSelf()` o `stopService()` . Tenga en cuenta que independientemente de la cantidad de veces

que llame a `onStartCommand()` , los métodos `stopSelf()` y `stopService()` deben invocarse solo una vez para detener el servicio.

- `onBind()` :

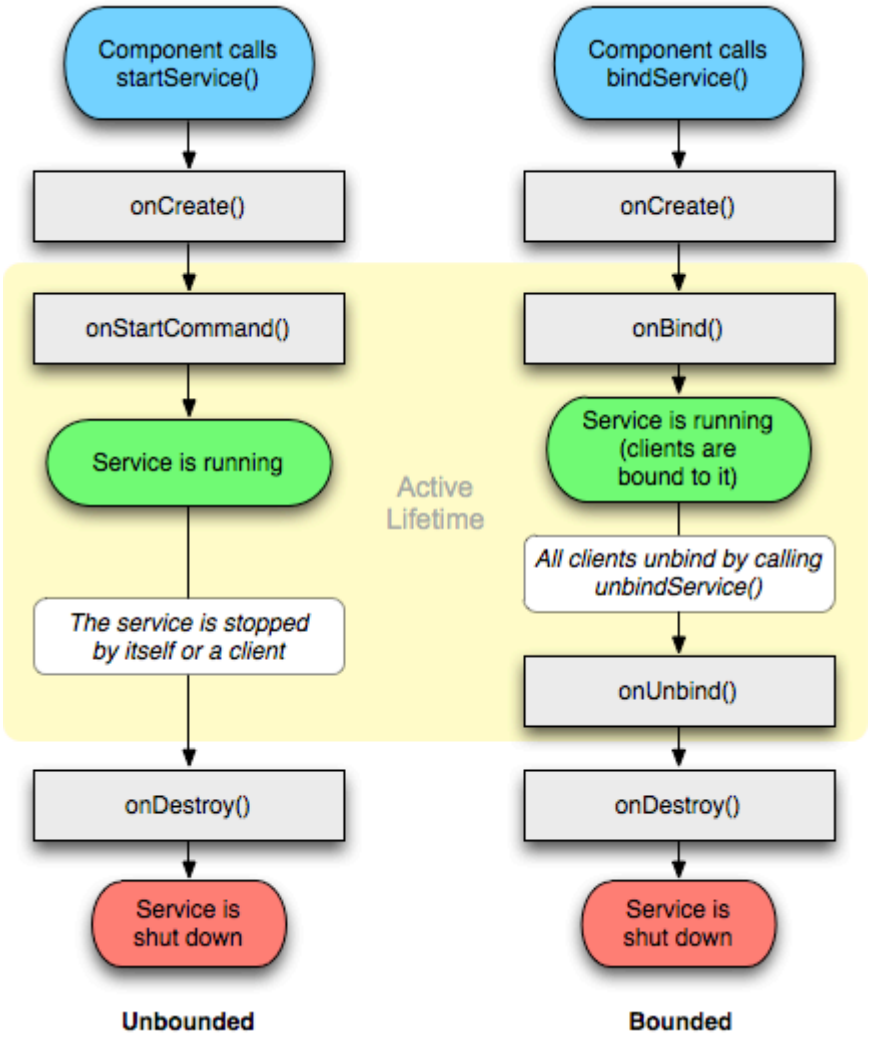
Se ejecuta cuando un componente llama a `bindService()` y devuelve una instancia de IBInder, proporcionando un canal de comunicación al Servicio. Una llamada a `bindService()` mantendrá el servicio en ejecución mientras haya clientes vinculados a él.

- `onDestroy()` :

Se ejecuta cuando el servicio ya no está en uso y permite la eliminación de los recursos que se han asignado.

Es importante tener en cuenta que durante el ciclo de vida de un servicio se pueden invocar otras devoluciones de llamada, como `onConfigurationChanged()` y `onLowMemory()`

https://developer.android.com/guide/components/services.html



Definiendo el proceso de un servicio.

El campo `android:process` define el nombre del proceso donde se ejecutará el servicio. Normalmente, todos los componentes de una aplicación se ejecutan en el proceso

predeterminado creado para la aplicación. Sin embargo, un componente puede anular el valor predeterminado con su propio atributo de proceso, lo que le permite distribuir su aplicación en varios procesos.

Si el nombre asignado a este atributo comienza con dos puntos (':'), el servicio se ejecutará en su propio proceso separado.

```
<service
  android:name="com.example.appName"
  android:process=":externalProcess" />
```

Si el nombre del proceso comienza con un carácter en minúscula, el servicio se ejecutará en un proceso global con ese nombre, siempre que tenga permiso para hacerlo. Esto permite que los componentes en diferentes aplicaciones compartan un proceso, reduciendo el uso de recursos.

Creando Servicio Bound con ayuda de Binder

Cree una clase que extienda la clase de `Service` y, en el método anulado `onBind` devuelva su instancia de carpeta local:

```
public class LocalService extends Service {
    // Binder given to clients
    private final IBinder mBinder = new LocalBinder();

    /**
     * Class used for the client Binder.  Because we know this service always
     * runs in the same process as its clients, we don't need to deal with IPC.
     */
    public class LocalBinder extends Binder {
        LocalService getService() {
            // Return this instance of LocalService so clients can call public methods
            return LocalService.this;
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }
}
```

Luego, en su actividad, vincule al servicio en la `onStart` llamada de `onStart` , utilizando la instancia de `ServiceConnection` y `onStop` en `onStop` :

```
public class BindingActivity extends Activity {
    LocalService mService;
    boolean mBound = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
```

```
    @Override
    protected void onStart() {
        super.onStart();
        // Bind to LocalService
        Intent intent = new Intent(this, LocalService.class);
        bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
    }

    @Override
    protected void onStop() {
        super.onStop();
        // Unbind from the service
        if (mBound) {
            unbindService(mConnection);
            mBound = false;
        }
    }

    /** Defines callbacks for service binding, passed to bindService() */
    private ServiceConnection mConnection = new ServiceConnection() {

        @Override
        public void onServiceConnected(ComponentName className,
                IBinder service) {
            // We've bound to LocalService, cast the IBinder and get LocalService instance
            LocalBinder binder = (LocalBinder) service;
            mService = binder.getService();
            mBound = true;
        }

        @Override
        public void onServiceDisconnected(ComponentName arg0) {
            mBound = false;
        }
    };
}
```

Creación de servicio remoto (a través de AIDL)

Describa su interfaz de acceso al servicio a través del archivo `.aidl` :

```
// IRemoteService.aidl
package com.example.android;

// Declare any non-default types here with import statements

/** Example service interface */
interface IRemoteService {
    /** Request the process ID of this service, to do evil things with it. */
    int getPid();
}
```

Ahora, después de la aplicación de compilación, las herramientas sdk generarán el archivo `.java` apropiado. Este archivo contendrá la clase `Stub` que implementa nuestra interfaz de ayuda, y que necesitamos extender:

```
public class RemoteService extends Service {
```

```
    private final IRemoteService.Stub binder = new IRemoteService.Stub() {
        @Override
        public int getPid() throws RemoteException {
            return Process.myPid();
        }
    };

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return binder;
    }
}
```

Luego en la actividad:

```
public class MainActivity extends AppCompatActivity {
    private final ServiceConnection connection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder iBinder) {
            IRemoteService service = IRemoteService.Stub.asInterface(iBinder);
            Toast.makeText(this, "Activity process: " + Process.myPid + ", Service process: "
+ getRemotePid(service), LENGTH_SHORT).show();
        }

        @Override
        public void onServiceDisconnected(ComponentName componentName) {}
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onStart() {
        super.onStart();
        Intent intent = new Intent(this, RemoteService.class);
        bindService(intent, connection, Context.BIND_AUTO_CREATE);
    }

    @Override
    protected void onStop() {
        super.onStop();
        unbindService(connection);
    }

    private int getRemotePid(IRemoteService service) {
        int result = -1;

        try {
            result = service.getPid();
        } catch (RemoteException e) {
            e.printStackTrace();
        }

        return result;
    }
```

```
}
```

Creación de un servicio independiente

Lo primero que debe hacer es agregar el servicio a `AndroidManifest.xml` , dentro de la etiqueta `<application>` :

```
<application ...>

    ...

    <service
        android:name=".RecordingService"
        <!--"enabled" tag specifies Whether or not the service can be instantiated by the
system – "true" -->
        <!--if it can be, and "false" if not. The default value is "true".-->
        android:enabled="true"
        <!--exported tag specifies Whether or not components of other applications can invoke
the -->
        <!--service or interact with it – "true" if they can, and "false" if not. When the
value-->
        <!--is "false", only components of the same application or applications with the same
user -->
        <!--ID can start the service or bind to it.-->
        android:exported="false" />

</application>
```

Si tiene la intención de administrar su clase de servicio en un paquete separado (por ejemplo: .AllServices.RecordingService), deberá especificar dónde se encuentra su servicio. Así, en el caso anterior modificaremos:

```
android:name=".RecordingService"
```

a

```
android:name=".AllServices.RecordingService"
```

o la forma más sencilla de hacerlo es especificar el nombre completo del paquete.

Luego creamos la clase de servicio real:

```
public class RecordingService extends Service {
    private int NOTIFICATION = 1; // Unique identifier for our notification

    public static boolean isRunning = false;
    public static RecordingService instance = null;


    private NotificationManager notificationManager = null;


    @Override
    public IBinder onBind(Intent intent) {
```

```
        return null;
    }

    @Override
    public void onCreate(){
        instance = this;
        isRunning = true;

        notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

        super.onCreate();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        // The PendingIntent to launch our activity if the user selects this notification
        PendingIntent contentIntent = PendingIntent.getActivity(this, 0, new Intent(this,
MainActivity.class), 0);

        // Set the info for the views that show in the notification panel.
        Notification notification = new NotificationCompat.Builder(this)
                .setSmallIcon(R.mipmap.ic_launcher)        // the status icon
                .setTicker("Service running...")           // the status text
                .setWhen(System.currentTimeMillis())       // the time stamp
                .setContentTitle("My App")                 // the label of the entry
                .setContentText("Service running...")      // the content of the entry
                .setContentIntent(contentIntent)           // the intent to send when the
entry is clicked
                .setOngoing(true)                          // make persistent (disable swipe-
away)
                .build();

        // Start service in foreground mode
        startForeground(NOTIFICATION, notification);

        return START_STICKY;
    }


    @Override
    public void onDestroy(){
        isRunning = false;
        instance = null;

        notificationManager.cancel(NOTIFICATION); // Remove notification

        super.onDestroy();
    }


    public void doSomething(){
        Toast.makeText(getApplicationContext(), "Doing stuff from service...",
Toast.LENGTH_SHORT).show();
    }

}
```

Todo lo que hace este servicio es mostrar una notificación cuando se está ejecutando, y puede mostrar brindis cuando se llama a su método `doSomething()` .

Como se dará cuenta, se implementa como un singleton , manteniendo un seguimiento de su propia instancia, pero sin el método de fábrica de singleton estático habitual, ya que los servicios son naturalmente singletons y son creados por intentos. La instancia es útil para el exterior para obtener un "identificador" del servicio cuando se está ejecutando.

Por último, necesitamos iniciar y detener el servicio desde una actividad:

```
public void startOrStopService(){
    if( RecordingService.isRunning ){
        // Stop service
        Intent intent = new Intent(this, RecordingService.class);
        stopService(intent);
    }
    else {
        // Start service
        Intent intent = new Intent(this, RecordingService.class);
        startService(intent);
    }
}
```

En este ejemplo, el servicio se inicia y se detiene con el mismo método, dependiendo de su estado actual.

También podemos invocar el método `doSomething()` desde nuestra actividad:

```
public void makeServiceDoSomething(){
    if( RecordingService.isRunning )
        RecordingService.instance.doSomething();
}
```

Lea Servicio en línea: https://riptutorial.com/es/android/topic/137/servicio

Capítulo 227: Servicio de Intención

Sintaxis

4. <service android: name = ". UploadS3IntentService" android: export = "false" />

Observaciones

Un `IntentService` proporciona una forma sencilla de descargar el trabajo en un hilo de fondo. Se encarga de todo acerca de recibir solicitudes, ponerlas en una cola, detenerse, etc. para usted. También es fácil de implementar, por lo que es perfecto para usar cuando tienes operaciones que requieren mucho tiempo y no pertenecen al subproceso principal (UI).

Examples

Creando un IntentService

Para crear un IntentService, cree una clase que extienda `IntentService` , y dentro de él, un método que `onHandleIntent` :

```
package com.example.myapp;
public class MyIntentService extends IntentService {
    @Override
     protected void onHandleIntent (Intent workIntent) {
         //Do something in the background, based on the contents of workIntent.
     }
}
```

Ejemplo de servicio de intenciones

Aquí hay un ejemplo de un `IntentService` que pretende cargar imágenes en segundo plano. Todo lo que necesita hacer para implementar un `IntentService` es proporcionar un constructor que llame al constructor `super(String)` , y debe implementar el `onHandleIntent(Intent)` .

```
public class ImageLoaderIntentService extends IntentService {

    public static final String IMAGE_URL = "url";

    /**
     * Define a constructor and call the super(String) constructor, in order to name the
worker
     * thread - this is important if you want to debug and know the name of the thread upon
     * which this Service is operating its jobs.
     */
    public ImageLoaderIntentService() {
        super("Example");
    }
```

```
    @Override
    protected void onHandleIntent(Intent intent) {
        // This is where you do all your logic - this code is executed on a background thread

        String imageUrl = intent.getStringExtra(IMAGE_URL);

        if (!TextUtils.isEmpty(imageUrl)) {
            Drawable image = HttpUtils.loadImage(imageUrl); // HttpUtils is made-up for the
example
        }

        // Send your drawable back to the UI now, so that you can use it - there are many ways
        // to achieve this, but they are out of reach for this example
    }
}
```

Para iniciar un `IntentService` , debes enviarle un `Intent` . Puedes hacerlo desde una `Activity` , por ejemplo. Por supuesto, no estás limitado a eso. Este es un ejemplo de cómo convocarías a tu nuevo `Service` de una clase de `Activity` .

```
Intent serviceIntent = new Intent(this, ImageLoaderIntentService.class); // you can use 'this'
as the first parameter if your class is a Context (i.e. an Activity, another Service, etc.),
otherwise, supply the context differently
serviceIntent.putExtra(IMAGE_URL, "http://www.example-site.org/some/path/to/an/image");
startService(serviceIntent); // if you are not using 'this' in the first line, you also have
to put the call to the Context object before startService(Intent) here
```

El `IntentService` procesa los datos de su `Intent` s de manera secuencial, de modo que puede enviar múltiples `Intent` sin preocuparse de si van a chocar entre sí. Solo se procesa una `Intent` a la vez, el resto va en una cola. Cuando todos los trabajos estén completos, el `IntentService` se cerrará automáticamente.

Ejemplo de IntentService Básico

La clase abstracta `IntentService` es una clase base para servicios, que se ejecuta en segundo plano sin ninguna interfaz de usuario. Por lo tanto, para actualizar la interfaz de usuario, tenemos que hacer uso de un receptor, que puede ser un `BroadcastReceiver` o un `ResultReceiver` :

- Se debe usar un `BroadcastReceiver` si su servicio necesita comunicarse con múltiples componentes que desean escuchar la comunicación.
- Un `ResultReceiver` : debe usarse si su servicio necesita comunicarse solo con la aplicación principal (es decir, su aplicación).

Dentro del `IntentService` , tenemos un método clave, `onHandleIntent()` , en el que realizaremos todas las acciones, por ejemplo, preparar notificaciones, crear alarmas, etc.

Si desea utilizar su propio `IntentService` , debe ampliarlo de la siguiente manera:

```
public class YourIntentService extends IntentService {
    public YourIntentService () {
        super("YourIntentService ");
    }
```

```
    @Override
    protected void onHandleIntent(Intent intent) {
        // TODO: Write your own code here.
    }
}
```

Llamar / comenzar la actividad se puede hacer de la siguiente manera:

```
Intent i = new Intent(this, YourIntentService.class);
startService(i);  // For the service.
startActivity(i); // For the activity; ignore this for now.
```

De manera similar a cualquier actividad, puede pasarle información adicional como los datos del paquete de la siguiente manera:

```
Intent passDataIntent = new Intent(this, YourIntentService.class);
msgIntent.putExtra("foo","bar");
startService(passDataIntent);
```

Ahora suponga que pasamos algunos datos a la clase `YourIntentService` . Sobre la base de estos datos, una acción se puede realizar de la siguiente manera:

```
public class YourIntentService extends IntentService {
    private String actvityValue="bar";
    String retrivedValue=intent.getStringExtra("foo");

    public YourIntentService () {
        super("YourIntentService ");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        if(retrivedValue.equals(actvityValue)){
            // Send the notification to foo.
        } else {
            // Retrieving data failed.
        }
    }
}
```

El código anterior también muestra cómo manejar las restricciones en el método `OnHandleIntent()` .

Lea Servicio de Intención en línea: https://riptutorial.com/es/android/topic/5319/servicio-de-intencion

Capítulo 228: shell adb

Introducción

`adb shell` abre un shell de Linux en un dispositivo o emulador de destino. Es la forma más potente y versátil de controlar un dispositivo Android a través de `adb` .

Este tema se dividió de ADB (Android Debug Bridge) debido a que llegó al límite de ejemplos, muchos de los cuales involucraban el comando `adb shell` .

Sintaxis

- shell adb [-e escape] [-n] [-tt] [-x] [comando]

Parámetros

Parámetro	Detalles
-mi	elegir el carácter de escape, o "ninguno"; por defecto '~'
-norte	no leer de stdin
-T	deshabilitar la asignación de PTY
-t	forzar la asignación de PTY
-X	deshabilitar códigos de salida remotos y separación stdout / stderr

Examples

Envíe texto, tecla presionada y eventos táctiles al dispositivo Android a través de ADB

ejecute el siguiente comando para insertar el texto en una vista con un enfoque (si es compatible con la entrada de texto)

6.0

Enviar texto en SDK 23+

```
adb shell "input keyboard text 'Paste text on Android Device'"
```

Si ya está conectado a su dispositivo a través de `adb` :

```
input text 'Paste text on Android Device'
```

6.0

Enviar texto antes del SDK 23

```
adb shell "input keyboard text 'Paste%stext%son%sAndroid%sDevice'"
```

No se aceptan espacios como entrada, reemplácelos con% s.

Enviar eventos

Para simular pulsando la tecla de encendido del hardware.

```
adb shell input keyevent 26
```

o alternativamente

```
adb shell input keyevent POWER
```

Incluso si no tiene una clave de hardware, puede utilizar un `keyevent` para realizar la acción equivalente

```
adb shell input keyevent CAMERA
```

Enviar evento táctil como entrada

```
adb shell input tap Xpoint Ypoint
```

Enviar evento swipe como entrada

```
adb shell input swipe Xpoint1 Ypoint1 Xpoint2 Ypoint2 [DURATION*]
```

* DURACIÓN es opcional, por defecto = 300ms. fuente

Obtenga puntos X e Y habilitando la ubicación del puntero en la opción de desarrollador.

ADB shell script de ejemplo

> Para ejecutar un script en Ubuntu, Create script.sh haga clic con el botón derecho en el archivo, agregue permiso de lectura / escritura y marque el permiso para **ejecutar el archivo como programa** .
>
> Abra el emulador de terminal y ejecute el comando ./script.sh

Script.sh

```
 for (( c=1; c<=5; c++ ))
    do
```

```
    adb shell input tap X Y
    echo "Clicked $c times"
    sleep 5s
done
```

Para una lista completa de números de eventos

- lista corta de varios eventos interesantes ADB Shell Input Events
- documentación de referencia https://developer.android.com/reference/android/view/KeyEvent.html#KEYCODE_POWER .

Listar paquetes

Imprime todos los paquetes, opcionalmente solo aquellos cuyo nombre de paquete contiene el texto en <FILTER>.

```
adb shell pm list packages [options] <FILTER>

All <FILTER>

adb shell pm list packages
```

Atributos:

`-f` para ver su archivo asociado.

`-i` Ver el instalador para los paquetes.

`-u` para incluir también paquetes desinstalados.

`-u` También incluye paquetes desinstalados.

Atributos que filtran:

`-d` para paquetes deshabilitados.

`-e` para paquetes habilitados.

`-s` para paquetes del sistema.

`-3` para paquetes de terceros.

`--user <USER_ID>` para un espacio de usuario específico para consultar.

Otorgar y revocar permisos API 23+

Una línea única que ayuda a otorgar o revocar permisos vulnerables.

- **otorgamiento**

```
adb shell pm grant <sample.package.id> android.permission.<PERMISSION_NAME>
```

- **revocando**

```
adb shell pm revoke <sample.package.id> android.permission.<PERMISSION_NAME>
```

- **Concesión de todos los permisos de tiempo de ejecución a la vez en la instalación (-g)**

```
adb install -g /path/to/sample_package.apk
```

Imprimir datos de la aplicación

Este comando imprime todos los datos relevantes de la aplicación:

- código de versión
- nombre de la versión
- permisos concedidos (Android API 23+)
- etc.

```
adb shell dumpsys package <your.package.id>
```

Grabando la pantalla

4.4

Grabación de la pantalla de dispositivos con Android 4.4 (nivel de API 19) y superior:

```
adb shell screenrecord [options] <filename>
adb shell screenrecord /sdcard/demo.mp4
```

(presiona Ctrl-C para detener la grabación)

Descarga el archivo desde el dispositivo:

```
adb pull /sdcard/demo.mp4
```

> Nota: detenga la grabación de la pantalla presionando Ctrl-C, de lo contrario, la grabación se detiene automáticamente a los tres minutos o el límite de tiempo establecido por - `--time-limit` .

```
adb shell screenrecord --size <WIDTHxHEIGHT>
```

Establece el tamaño del video: 1280x720. El valor predeterminado es la resolución de pantalla nativa del dispositivo (si es compatible), 1280x720 si no es así. Para obtener los mejores resultados, use un tamaño compatible con el codificador de codificación avanzada de video (AVC) de su dispositivo.

```
adb shell screenrecord --bit-rate <RATE>
```

Establece la tasa de bits de video para el video, en megabits por segundo. El valor predeterminado es 4Mbps. Puede aumentar la velocidad de bits para mejorar la calidad del video, pero al hacerlo se obtienen archivos de películas más grandes. El siguiente ejemplo establece la tasa de bits de grabación a 5Mbps:

```
adb shell screenrecord --bit-rate 5000000 /sdcard/demo.mp4
```

```
adb shell screenrecord --time-limit <TIME>
```

Establece el tiempo máximo de grabación, en segundos. El valor predeterminado y máximo es 180 (3 minutos).

```
adb shell screenrecord --rotate
```

Gira la salida 90 grados. Esta característica es experimental.

```
adb shell screenrecord --verbose
```

Muestra información de registro en la pantalla de línea de comandos. Si no configura esta opción, la utilidad no muestra ninguna información mientras se ejecuta.

> Nota: Esto podría no funcionar en algunos dispositivos.

4.4

El comando de grabación de pantalla no es compatible con las versiones de Android anteriores a 4.4.

> El comando screenrecord es una utilidad de shell para grabar la pantalla de dispositivos con Android 4.4 (nivel de API 19) y superior. La utilidad registra la actividad de la pantalla en un archivo MPEG-4.

Cambio de permisos de archivos usando el comando chmod

Tenga en cuenta que para poder cambiar los envíos de archivos, su dispositivo debe estar rooteado, ¡ `su` ***binario no viene con los dispositivos de fábrica!***

Convención:

```
adb shell su -c "chmod <numeric-permisson> <file>"
```

Permisos numéricos construidos a partir de secciones de usuario, grupo y mundo.

Por ejemplo, si desea que todos puedan cambiar el archivo para que todos puedan leerlo,

escribirlo y ejecutarlo, este será su comando:

```
adb shell su -c "chmod 777 <file-path>"
```

O

```
adb shell su -c "chmod 000 <file-path>"
```

Si intenta denegar cualquier permiso para ello.

1er dígito: especifica el permiso del usuario, **2º dígito** : especifica el permiso de grupo, **3º dígito** : especifica el permiso del mundo (otros).

Permisos de acceso:

```
--- :   binary value:   000,  octal value: 0 (none)
--x :   binary value:   001,  octal value: 1 (execute)
-w- :   binary value:   010,  octal value: 2 (write)
-wx :   binary value:   011,  octal value: 3 (write, execute)
r-- :   binary value:   100,  octal value: 4 (read)
r-x :   binary value:   101,  octal value: 5 (read, execute)
rw- :   binary value:   110,  octal value: 6 (read, write)
rwx :   binary value:   111,  octal value: 7 (read, write, execute)
```

Establecer fecha / hora a través de adb

6.0

El formato SET predeterminado es `MMDDhhmm[[CC]YY][.ss]` , eso es (2 dígitos cada uno)

Por ejemplo, para configurar el 17 de julio a las 10:10 am, sin cambiar el año actual, escriba:

```
adb shell 'date 07171010.00'
```

Consejo 1: el cambio de fecha no se reflejará de inmediato, y se producirá un cambio notable solo después de que el reloj del sistema avance al minuto siguiente.
Puede forzar una actualización adjuntando una `TIME_SET` intento `TIME_SET` a su llamada, de esta manera:

```
adb shell 'date 07171010.00 ; am broadcast -a android.intent.action.TIME_SET'
```

Consejo 2: para sincronizar el reloj de Android con su máquina local:

Linux:

```
adb shell date `date +%m%d%H%M%G.%S`
```

Windows (PowerShell):

```
$currentDate = Get-Date -Format "MMddHHmmyyyy.ss" # Android's preferred format
adb shell "date $currentDate"
```

Ambos consejos juntos:

```
adb shell 'date `date +%m%d%H%M%G.%S` ; am broadcast -a android.intent.action.TIME_SET'
```

6.0

El formato SET predeterminado es 'YYYYMMDD.HHmmss'

```
adb shell 'date -s 20160117.095930'
```

Consejo: para sincronizar el reloj de Android con su máquina local (basada en Linux):

```
adb shell date -s `date +%G%m%d.%H%M%S`
```

Opciones de desarrollador abierto

```
adb shell am start -n com.android.settings/.DevelopmentSettings
```

Navegará tu dispositivo / emulador a la sección `Developer Options` .

Generando una transmisión "Boot Complete"

Esto es relevante para las aplicaciones que implementan un `BootListener` . Prueba tu aplicación matando tu aplicación y luego prueba con:

```
adb shell am broadcast -a android.intent.action.BOOT_COMPLETED -c android.intent.category.HOME
-n your.app/your.app.BootListener
```

(Reemplace `your.package/your.app.BootListener` con los valores adecuados).

Ver contenido de almacenamiento externo / secundario

Ver contenido:

```
adb shell ls \$EXTERNAL_STORAGE
adb shell ls \$SECONDARY_STORAGE
```

Ver ruta:

```
adb shell echo \$EXTERNAL_STORAGE
adb shell echo \$SECONDARY_STORAGE
```

matar un proceso dentro de un dispositivo Android

A veces, el logcat de Android se ejecuta infinitamente con errores provenientes de algún proceso que usted no posee, agotando la batería o dificultando la depuración de su código.

Una forma conveniente de solucionar el problema sin reiniciar el dispositivo es localizar y detener el proceso que causa el problema.

Desde logcat

```
03-10 11:41:40.010 1550-1627/? E/SomeProcess: ....
```

Observe el número de proceso: 1550

Ahora podemos abrir una cáscara y matar el proceso. Tenga en cuenta que no podemos matar el proceso de la `root` .

```
adb shell
```

Dentro de la cáscara podemos verificar más sobre el proceso usando

```
ps -x | grep 1550
```

Y mátalo si queremos:

```
kill -9 1550
```

Lea shell adb en línea: https://riptutorial.com/es/android/topic/9408/shell-adb

Capítulo 229: ShortcutManager

Examples

Atajos de lanzadores dinámicos

```
ShortcutManager shortcutManager = getSystemService(ShortcutManager.class);

ShortcutInfo shortcut = new ShortcutInfo.Builder(this, "id1")
    .setShortLabel("Web site") // Shortcut Icon tab
    .setLongLabel("Open the web site") // Displayed When Long Pressing On App Icon
    .setIcon(Icon.createWithResource(context, R.drawable.icon_website))
    .setIntent(new Intent(Intent.ACTION_VIEW,
                   Uri.parse("https://www.mysite.example.com/")))
    .build();

shortcutManager.setDynamicShortcuts(Arrays.asList(shortcut));
```

Podemos eliminar todos los accesos directos dinámicos fácilmente llamando a:

```
 shortcutManager.removeAllDynamicShortcuts();
```

Podemos actualizar los Shorcuts dinámicos existentes usando

```
shortcutManager.updateShortcuts(Arrays.asList(shortcut);
```

Tenga en cuenta que `setDynamicShortcuts(List)` se usa para redefinir la lista completa de accesos directos dinámicos, `addDynamicShortcuts(List)` se usa para agregar accesos directos dinámicos a la lista existente de accesos directos dinámicos

Lea ShortcutManager en línea: https://riptutorial.com/es/android/topic/7661/shortcutmanager

Capítulo 230: Sincronización de datos con el adaptador de sincronización

Examples

Dummy Sync Adapter con proveedor de código auxiliar

SyncAdapter

```
/**
 * Define a sync adapter for the app.
 * <p/>
 * <p>This class is instantiated in {@link SyncService}, which also binds SyncAdapter to the
system.
 * SyncAdapter should only be initialized in SyncService, never anywhere else.
 * <p/>
 * <p>The system calls onPerformSync() via an RPC call through the IBinder object supplied by
 * SyncService.
 */
class SyncAdapter extends AbstractThreadedSyncAdapter {
    /**
     * Constructor. Obtains handle to content resolver for later use.
     */
    public SyncAdapter(Context context, boolean autoInitialize) {
        super(context, autoInitialize);
    }

    /**
     * Constructor. Obtains handle to content resolver for later use.
     */
    public SyncAdapter(Context context, boolean autoInitialize, boolean allowParallelSyncs) {
        super(context, autoInitialize, allowParallelSyncs);
    }

    @Override
    public void onPerformSync(Account account, Bundle extras, String authority,
                              ContentProviderClient provider, SyncResult syncResult) {
        //Jobs you want to perform in background.
        Log.e("" + account.name, "Sync Start");
    }

}
```

Servicio de sincronización

```
/**
 * Define a Service that returns an IBinder for the
 * sync adapter class, allowing the sync adapter framework to call
 * onPerformSync().
 */
public class SyncService extends Service {
    // Storage for an instance of the sync adapter
    private static SyncAdapter sSyncAdapter = null;
    // Object to use as a thread-safe lock
```

```
    private static final Object sSyncAdapterLock = new Object();

    /*
     * Instantiate the sync adapter object.
     */
    @Override
    public void onCreate() {
        /*
         * Create the sync adapter as a singleton.
         * Set the sync adapter as syncable
         * Disallow parallel syncs
         */
        synchronized (sSyncAdapterLock) {
            if (sSyncAdapter == null) {
                sSyncAdapter = new SyncAdapter(getApplicationContext(), true);
            }
        }
    }

    /**
     * Return an object that allows the system to invoke
     * the sync adapter.
     */
    @Override
    public IBinder onBind(Intent intent) {
        /*
         * Get the object that allows external processes
         * to call onPerformSync(). The object is created
         * in the base class code when the SyncAdapter
         * constructors call super()
         */
        return sSyncAdapter.getSyncAdapterBinder();
    }
}
```

Autenticador

```
public class Authenticator extends AbstractAccountAuthenticator {
    // Simple constructor
    public Authenticator(Context context) {
        super(context);
    }

    // Editing properties is not supported
    @Override
    public Bundle editProperties(
            AccountAuthenticatorResponse r, String s) {
        throw new UnsupportedOperationException();
    }

    // Don't add additional accounts
    @Override
    public Bundle addAccount(
            AccountAuthenticatorResponse r,
            String s,
            String s2,
            String[] strings,
            Bundle bundle) throws NetworkErrorException {
        return null;
    }
```

```
    // Ignore attempts to confirm credentials
    @Override
    public Bundle confirmCredentials(
            AccountAuthenticatorResponse r,
            Account account,
            Bundle bundle) throws NetworkErrorException {
        return null;
    }

    // Getting an authentication token is not supported
    @Override
    public Bundle getAuthToken(
            AccountAuthenticatorResponse r,
            Account account,
            String s,
            Bundle bundle) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }

    // Getting a label for the auth token is not supported
    @Override
    public String getAuthTokenLabel(String s) {
        throw new UnsupportedOperationException();
    }

    // Updating user credentials is not supported
    @Override
    public Bundle updateCredentials(
            AccountAuthenticatorResponse r,
            Account account,
            String s, Bundle bundle) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }

    // Checking features for the account is not supported
    @Override
    public Bundle hasFeatures(
            AccountAuthenticatorResponse r,
            Account account, String[] strings) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }
}
```

Servicio de Autenticador

```
/**
 * A bound Service that instantiates the authenticator
 * when started.
 */
public class AuthenticatorService extends Service {
    // Instance field that stores the authenticator object
    private Authenticator mAuthenticator;
    @Override
    public void onCreate() {
        // Create a new authenticator object
        mAuthenticator = new Authenticator(this);
    }
    /*
     * When the system binds to this Service to make the RPC call
```

```
     * return the authenticator's IBinder.
     */
    @Override
    public IBinder onBind(Intent intent) {
        return mAuthenticator.getIBinder();
    }
}
```

AndroidManifest.xml adiciones

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.READ_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />

     <service
        android:name=".syncAdapter.SyncService"
        android:exported="true">
        <intent-filter>
            <action android:name="android.content.SyncAdapter" />
        </intent-filter>
        <meta-data
            android:name="android.content.SyncAdapter"
            android:resource="@xml/syncadapter" />
    </service>

    <service android:name=".authenticator.AuthenticatorService">
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator" />
        </intent-filter>
        <meta-data
            android:name="android.accounts.AccountAuthenticator"
            android:resource="@xml/authenticator" />
    </service>

    <provider
        android:name=".provider.StubProvider"
        android:authorities="com.yourpackage.provider"
        android:exported="false"
        android:syncable="true" />
```

res / xml / authenticator.xml

```
<?xml version="1.0" encoding="utf-8"?>
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.yourpackage"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:smallIcon="@mipmap/ic_launcher" />
```

res / xml / syncadapter.xml

```
<?xml version="1.0" encoding="utf-8"?>
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.yourpackage.android"
    android:allowParallelSyncs="false"
    android:contentAuthority="com.yourpackage.provider"
    android:isAlwaysSyncable="true"
    android:supportsUploading="false"
```

```
    android:userVisible="false" />
```

StubProvider

```
/*
 * Define an implementation of ContentProvider that stubs out
 * all methods
 */
public class StubProvider extends ContentProvider {
    /*
     * Always return true, indicating that the
     * provider loaded correctly.
     */
    @Override
    public boolean onCreate() {
        return true;
    }

    /*
     * Return no type for MIME type
     */
    @Override
    public String getType(Uri uri) {
        return null;
    }

    /*
     * query() always returns no results
     *
     */
    @Override
    public Cursor query(
            Uri uri,
            String[] projection,
            String selection,
            String[] selectionArgs,
            String sortOrder) {
        return null;
    }

    /*
     * insert() always returns null (no URI)
     */
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        return null;
    }

    /*
     * delete() always returns "no rows affected" (0)
     */
    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        return 0;
    }

    /*
     * update() always returns "no rows affected" (0)
     */
    public int update(
```

```
            Uri uri,
            ContentValues values,
            String selection,
            String[] selectionArgs) {
        return 0;
    }
}
```

Llame a esta función si inicia sesión correctamente para crear una cuenta con el ID de usuario registrado.

```
public Account CreateSyncAccount(Context context, String accountName) {
        // Create the account type and default account
        Account newAccount = new Account(
                accountName, "com.yourpackage");
        // Get an instance of the Android account manager
        AccountManager accountManager =
                (AccountManager) context.getSystemService(
                        ACCOUNT_SERVICE);
        /*
         * Add the account and account type, no password or user data
         * If successful, return the Account object, otherwise report an error.
         */
        if (accountManager.addAccountExplicitly(newAccount, null, null)) {
            /*
             * If you don't set android:syncable="true" in
             * in your <provider> element in the manifest,
             * then call context.setIsSyncable(account, AUTHORITY, 1)
             * here.
             */
        } else {
            /*
             * The account exists or some other error occurred. Log this, report it,
             * or handle it internally.
             */
        }
        return newAccount;
    }
```

Forzando una sincronización

```
Bundle bundle = new Bundle();
bundle.putBoolean(ContentResolver.SYNC_EXTRAS_EXPEDITED, true);
bundle.putBoolean(ContentResolver.SYNC_EXTRAS_FORCE, true);
bundle.putBoolean(ContentResolver.SYNC_EXTRAS_MANUAL, true);
ContentResolver.requestSync(null, MyContentProvider.getAuthority(), bundle);
```

Lea Sincronización de datos con el adaptador de sincronización en línea: https://riptutorial.com/es/android/topic/1944/sincronizacion-de-datos-con-el-adaptador-de-sincronizacion

Capítulo 231: Snackbar

Sintaxis

- Snackbar make (Vista, texto CharSequence, duración int)
- Snackbar make (Vista, int int., Int int)

Parámetros

Parámetro	Descripción
ver	Vista: La vista para encontrar un padre de.
texto	CharSequence: El texto a mostrar. Se puede formatear texto.
resuelto	int: el ID de recurso del recurso de cadena a usar. Se puede formatear texto.
duración	int: cuánto tiempo se muestra el mensaje. Esto puede ser LENGTH_SHORT, LENGTH_LONG o LENGTH_INDEFINITE

Observaciones

Snackbar proporciona comentarios ligeros sobre una operación. Muestra un breve mensaje en la parte inferior de la pantalla en el dispositivo móvil y en la parte inferior izquierda en dispositivos más grandes. Los Snackbars aparecen sobre todos los demás elementos en la pantalla y solo se puede mostrar uno a la vez.

Desaparecen automáticamente después de un tiempo de espera o después de la interacción del usuario en otra parte de la pantalla, especialmente después de las interacciones que convocan una nueva superficie o actividad. Snackbar se puede deslizar fuera de la pantalla.

Antes de usar `SnackBar` , debe agregar la dependencia de la biblioteca de soporte de diseño en el archivo `build.gradle` :

```
dependencies {
    compile 'com.android.support:design:25.3.1'
}
```

Documentacion oficial

https://developer.android.com/reference/android/support/design/widget/Snackbar.html

Examples

Creando un Snackbar simple

La creación de un `Snackbar` se puede hacer de la siguiente manera:

```
Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG).show();
```

La `view` se utiliza para encontrar un padre adecuado para mostrar el `Snackbar` . Por lo general, este sería un `CoordinatorLayout` que ha definido en su XML, que permite agregar funcionalidades como deslizar para descartar y mover automáticamente otros widgets (por ejemplo, `FloatingActionButton`). Si no hay `CoordinatorLayout` , se utiliza la vista de contenido de la decoración de la ventana.

Muy a menudo también agregamos una acción al `Snackbar` . Un caso de uso común sería una acción "Deshacer".

```
Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG)
        .setAction("UNDO", new View.OnClickListener() {
            @Override
               public void onClick(View view) {
                   // put your logic here

               }
         })
        .show();
```

Puedes crear un `Snackbar` y mostrarlo más tarde:

```
Snackbar snackbar = Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG);
snackbar.show();
```

Si quieres cambiar el color del texto de la `Snackbar` :

```
Snackbar snackbar = Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG);
View view = snackbar .getView();
TextView textView = (TextView) view.findViewById(android.support.design.R.id.snackbar_text);
textView.setTextColor(Color.parseColor("#FF4500"));
snackbar.show();
```

De forma predeterminada, `Snackbar` despide con el golpe derecho. Este ejemplo muestra cómo descartar la barra de bocadillos con el golpe hacia la izquierda .

Snack Bar personalizado

Función para personalizar snackbar

```
public static Snackbar makeText(Context context, String message, int duration) {
        Activity activity = (Activity) context;
        View layout;
        Snackbar snackbar = Snackbar
                .make(activity.findViewById(android.R.id.content), message, duration);
        layout = snackbar.getView();
        //setting background color
```

```
        layout.setBackgroundColor(context.getResources().getColor(R.color.orange));
        android.widget.TextView text = (android.widget.TextView)
layout.findViewById(android.support.design.R.id.snackbar_text);
        //setting font color
        text.setTextColor(context.getResources().getColor(R.color.white));
        Typeface font = null;
        //Setting font
        font = Typeface.createFromAsset(context.getAssets(), "DroidSansFallbackanmol256.ttf");
        text.setTypeface(font);
        return snackbar;

    }
```

Llama a la función desde un fragmento o actividad.

```
 SnackBar.makeText(MyActivity.this, "Please Locate your address at Map",
Snackbar.LENGTH_SHORT).show();
```

Snackbar con devolución de llamada

Puede usar Snackbar.Callback para escuchar si la barra de aperitivos fue descartada por el usuario o el tiempo de espera.

```
Snackbar.make(getView(), "Hi snackbar!", Snackbar.LENGTH_LONG).setCallback( new
Snackbar.Callback() {
                @Override
                public void onDismissed(Snackbar snackbar, int event) {
                    switch(event) {
                        case Snackbar.Callback.DISMISS_EVENT_ACTION:
                            Toast.makeText(getActivity(), "Clicked the action",
Toast.LENGTH_LONG).show();
                            break;
                        case Snackbar.Callback.DISMISS_EVENT_TIMEOUT:
                            Toast.makeText(getActivity(), "Time out",
Toast.LENGTH_LONG).show();
                            break;
                    }
                }

                @Override
                public void onShown(Snackbar snackbar) {
                    Toast.makeText(getActivity(), "This is my annoying step-brother",
Toast.LENGTH_LONG).show();
                }
            }).setAction("Go!", new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                }
            }).show();
```

Snackbar personalizado

Este ejemplo muestra un Snackbar blanco con un icono personalizado de Deshacer.

```
Snackbar customBar = Snackbar.make(view , "Text to be displayed", Snackbar.LENGTH_LONG);
customBar.setAction("UNDO", new View.OnClickListener() {
    @Override
    public void onClick(View view) {
    //Put the logic for undo button here

    }
});

View sbView = customBar.getView();
//Changing background to White
sbView.setBackgroundColor(Color.WHITE));

TextView snackText = (TextView)
sbView.findViewById(android.support.design.R.id.snackbar_text);
if (snackText!=null) {
    //Changing text color to Black
   snackText.setTextColor(Color.BLACK);
}

TextView actionText = (TextView)
sbView.findViewById(android.support.design.R.id.snackbar_action);
if (actionText!=null) {
    // Setting custom Undo icon
    actionText.setCompoundDrawablesRelativeWithIntrinsicBounds(R.drawable.custom_undo, 0, 0,
0);
}
customBar.show();
```

Snackbar vs Tostadas: ¿Cuál debo usar?

Los brindis se utilizan generalmente cuando queremos mostrar información al usuario sobre alguna acción que haya sucedido (o no) con éxito y esta acción no requiere que el usuario realice ninguna otra acción. Como cuando un mensaje ha sido enviado, por ejemplo:

```
Toast.makeText(this, "Message Sent!", Toast.LENGTH_SHORT).show();
```

Snackbars también se utilizan para mostrar una información. Pero esta vez, podemos darle al usuario la oportunidad de tomar una acción. Por ejemplo, digamos que el usuario eliminó una imagen por error y quiere recuperarla. Podemos proporcionar un Snackbar con la acción "Deshacer". Me gusta esto:

```
Snackbar.make(getCurrentFocus(), "Picture Deleted", Snackbar.LENGTH_SHORT)
        .setAction("Undo", new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Return his picture
            }
        })
        .show();
```

Conclusión: los brindis se usan cuando no necesitamos la interacción del usuario. Snackbars se utilizan para permitir a los usuarios realizar otra acción o deshacer una acción anterior.

Snackbar personalizado (no hay necesidad de ver)

Creando un Snackbar sin la necesidad de la vista de pase a Snackbar, todo el diseño creado en Android en android.R.id.content.

```
public class CustomSnackBar {

    public static final int STATE_ERROR = 0;
    public static final int STATE_WARNING = 1;
    public static final int STATE_SUCCESS = 2;
    public static final int VIEW_PARENT = android.R.id.content;

    public CustomSnackBar(View view, String message, int actionType) {
        super();

        Snackbar snackbar = Snackbar.make(view, message, Snackbar.LENGTH_LONG);
        View sbView = snackbar.getView();
        TextView textView = (TextView)
sbView.findViewById(android.support.design.R.id.snackbar_text);
        textView.setTextColor(Color.parseColor("#ffffff"));
        textView.setTextSize(TypedValue.COMPLEX_UNIT_SP, 14);
        textView.setGravity(View.TEXT_ALIGNMENT_CENTER);
        textView.setLayoutDirection(View.LAYOUT_DIRECTION_RTL);

        switch (actionType) {
            case STATE_ERROR:
                snackbar.getView().setBackgroundColor(Color.parseColor("#F12B2B"));
                break;
            case STATE_WARNING:
                snackbar.getView().setBackgroundColor(Color.parseColor("#000000"));
                break;
            case STATE_SUCCESS:
                snackbar.getView().setBackgroundColor(Color.parseColor("#7ED321"));
                break;
        }
        snackbar.show();
    }
}
```

para la clase de llamada

nuevo CustomSnackBar (findViewById (CustomSnackBar.VIEW_PARENT), "mensaje", CustomSnackBar.STATE_ERROR);

Lea Snackbar en línea: https://riptutorial.com/es/android/topic/1500/snackbar

Capítulo 232: Sonido y Medios Android

Examples

Cómo escoger imagen y video para api> 19

Este es un código probado para imagen y video. Funcionará para todas las API menores de 19 y mayores de 19 también.

Imagen:

```
if (Build.VERSION.SDK_INT <= 19) {
                        Intent i = new Intent();
                        i.setType("image/*");
                        i.setAction(Intent.ACTION_GET_CONTENT);
                        i.addCategory(Intent.CATEGORY_OPENABLE);
                        startActivityForResult(i, 10);
                    } else if (Build.VERSION.SDK_INT > 19) {
                        Intent intent = new Intent(Intent.ACTION_PICK,
android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                        startActivityForResult(intent, 10);
                    }
```

Vídeo:

```
if (Build.VERSION.SDK_INT <= 19) {
                        Intent i = new Intent();
                        i.setType("video/*");
                        i.setAction(Intent.ACTION_GET_CONTENT);
                        i.addCategory(Intent.CATEGORY_OPENABLE);
                        startActivityForResult(i, 20);
                    } else if (Build.VERSION.SDK_INT > 19) {
                        Intent intent = new Intent(Intent.ACTION_PICK,
android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI);
                        startActivityForResult(intent, 20);
                    }
```

.

```
     @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == Activity.RESULT_OK) {

            if (requestCode == 10) {
                Uri selectedImageUri = data.getData();
                String selectedImagePath = getRealPathFromURI(selectedImageUri);
            } else if (requestCode == 20) {
                Uri selectedVideoUri = data.getData();
                String selectedVideoPath = getRealPathFromURI(selectedVideoUri);
            }

     public String getRealPathFromURI(Uri uri) {
```

```
            if (uri == null) {
                return null;
            }
            String[] projection = {MediaStore.Images.Media.DATA};
            Cursor cursor = getActivity().getContentResolver().query(uri, projection, null,
null, null);
            if (cursor != null) {
                int column_index = cursor
                        .getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
                cursor.moveToFirst();
                return cursor.getString(column_index);
            }
            return uri.getPath();
        }
```

Reproducir sonidos a través de SoundPool

```
public class PlaySound extends Activity implements OnTouchListener {
    private SoundPool soundPool;
    private int soundID;
    boolean loaded = false;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            View view = findViewById(R.id.textView1);
            view.setOnTouchListener(this);
            // Set the hardware buttons to control the music
            this.setVolumeControlStream(AudioManager.STREAM_MUSIC);
            // Load the sound
            soundPool = new SoundPool(10, AudioManager.STREAM_MUSIC, 0);
            soundPool.setOnLoadCompleteListener(new OnLoadCompleteListener() {
                    @Override
                    public void onLoadComplete(SoundPool soundPool, int sampleId,
                                    int status) {
                            loaded = true;
                    }
            });
            soundID = soundPool.load(this, R.raw.sound1, 1);

    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    // Getting the user sound settings
                    AudioManager audioManager = (AudioManager)
getSystemService(AUDIO_SERVICE);
                    float actualVolume = (float) audioManager
                                    .getStreamVolume(AudioManager.STREAM_MUSIC);
                    float maxVolume = (float) audioManager
                                    .getStreamMaxVolume(AudioManager.STREAM_MUSIC);
                    float volume = actualVolume / maxVolume;
                    // Is the sound loaded already?
                    if (loaded) {
                            soundPool.play(soundID, volume, volume, 1, 0, 1f);
                            Log.e("Test", "Played sound");
```

```
                    }
                }
                return false;
    }
}
```

Lea Sonido y Medios Android en línea: https://riptutorial.com/es/android/topic/4730/sonido-y-medios-android

Capítulo 233: SpannableString

Sintaxis

- `char charAt (int i)`
- `boolean equals (Object o)`
- `void getChars (int start, int end, char[] dest, int off)`
- `int getSpanEnd (Object what)`
- `int getSpanFlags (Object what)`
- `int getSpanStart (Object what)`
- `T[] getSpans (int queryStart, int queryEnd, Class<T> kind)`
- `int hashCode ()`
- `int length ()`
- `int nextSpanTransition (int start, int limit, Class kind)`
- void removeSpan (Objeto de qué)
- `void setSpan (Object what, int start, int end, int flags)`
- `CharSequence subSequence (int start, int end)`
- `String toString ()`
- `SpannableString valueOf (CharSequence source)`

Examples

Añadir estilos a un TextView

En el siguiente ejemplo, creamos una Actividad para mostrar un solo TextView.

El TextView utilizará un `SpannableString` como su contenido, que ilustrará algunos de los estilos disponibles.

Aquí 'lo que vamos a hacer con el texto:

- Hazlo más grande
- Negrita
- Subrayar
- Escribir en cursiva
- Huelga a través
- De colores
- Resaltado
- Mostrar como superíndice
- Mostrar como subíndice
- Mostrar como enlace
- Hazlo pulsable.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);

 SpannableString styledString
```

```
 = new SpannableString("Large\n\n"     // index 0 - 5
      + "Bold\n\n"          // index 7 - 11
      + "Underlined\n\n"    // index 13 - 23
      + "Italic\n\n"        // index 25 - 31
      + "Strikethrough\n\n" // index 33 - 46
      + "Colored\n\n"       // index 48 - 55
      + "Highlighted\n\n"   // index 57 - 68
      + "K Superscript\n\n" // "Superscript" index 72 - 83
      + "K Subscript\n\n"   // "Subscript" index 87 - 96
      + "Url\n\n"           //  index 98 - 101
      + "Clickable\n\n");   // index 103 - 112

 // make the text twice as large
 styledString.setSpan(new RelativeSizeSpan(2f), 0, 5, 0);

 // make text bold
 styledString.setSpan(new StyleSpan(Typeface.BOLD), 7, 11, 0);

 // underline text
 styledString.setSpan(new UnderlineSpan(), 13, 23, 0);

 // make text italic
 styledString.setSpan(new StyleSpan(Typeface.ITALIC), 25, 31, 0);

 styledString.setSpan(new StrikethroughSpan(), 33, 46, 0);

 // change text color
 styledString.setSpan(new ForegroundColorSpan(Color.GREEN), 48, 55, 0);

 // highlight text
 styledString.setSpan(new BackgroundColorSpan(Color.CYAN), 57, 68, 0);

 // superscript
 styledString.setSpan(new SuperscriptSpan(), 72, 83, 0);
 // make the superscript text smaller
 styledString.setSpan(new RelativeSizeSpan(0.5f), 72, 83, 0);

 // subscript
 styledString.setSpan(new SubscriptSpan(), 87, 96, 0);
 // make the subscript text smaller
 styledString.setSpan(new RelativeSizeSpan(0.5f), 87, 96, 0);

 // url
 styledString.setSpan(new URLSpan("http://www.google.com"), 98, 101, 0);

 // clickable text
 ClickableSpan clickableSpan = new ClickableSpan() {

 @Override
 public void onClick(View widget) {
// We display a Toast. You could do anything you want here.
Toast.makeText(SpanExample.this, "Clicked", Toast.LENGTH_SHORT).show();

  }
};

 styledString.setSpan(clickableSpan, 103, 112, 0);


 // Give the styled string to a TextView
 TextView textView = new TextView(this);
```

```
 // this step is mandated for the url and clickable styles.
 textView.setMovementMethod(LinkMovementMethod.getInstance());

 // make it neat
textView.setGravity(Gravity.CENTER);
textView.setBackgroundColor(Color.WHITE);

textView.setText(styledString);

setContentView(textView);

}
```

Y el resultado se verá así:



Multi cadena, con multi color.

Método: **setSpanColor**

```
public Spanned setSpanColor(String string, int color){
    SpannableStringBuilder builder = new SpannableStringBuilder();
    SpannableString ss = new SpannableString(string);
    ss.setSpan(new ForegroundColorSpan(color), 0, string.length(), 0);
    builder.append(ss);
```

```
    return ss;
}
```

Uso:

```
String a = getString(R.string.string1);
String b = getString(R.string.string2);

Spanned color1 = setSpanColor(a,Color.CYAN);
Spanned color2 = setSpanColor(b,Color.RED);
Spanned mixedColor = TextUtils.concat(color1, " ", color2);
// Now we use `mixedColor`
```

Lea SpannableString en línea: https://riptutorial.com/es/android/topic/10553/spannablestring

Capítulo 234: SQLite

Introducción

SQLite es un sistema de gestión de bases de datos relacionales escrito en C. Para comenzar a trabajar con bases de datos SQLite en el marco de Android, defina una clase que extienda SQLiteOpenHelper y personalice según sea necesario.

Observaciones

La clase `SQLiteOpenHelper` define `onCreate()` estáticos `onCreate()` y `onUpgrade()` . Estos métodos se llaman en los métodos correspondientes de una subclase de `SQLiteOpenHelper` que personaliza con sus propias tablas.

Examples

Usando la clase SQLiteOpenHelper

```
public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "Example.db";
    private static final int DATABASE_VERSION = 3;

    // For all Primary Keys _id should be used as column name
    public static final String COLUMN_ID = "_id";

    // Definition of table and column names of Products table
    public static final String TABLE_PRODUCTS = "Products";
    public static final String COLUMN_NAME = "Name";
    public static final String COLUMN_DESCRIPTION = "Description";
    public static final String COLUMN_VALUE = "Value";

    // Definition of table and column names of Transactions table
    public static final String TABLE_TRANSACTIONS = "Transactions";
    public static final String COLUMN_PRODUCT_ID = "ProductId";
    public static final String COLUMN_AMOUNT = "Amount";

    // Create Statement for Products Table
    private static final String CREATE_TABLE_PRODUCT = "CREATE TABLE " + TABLE_PRODUCTS + "
(" +
            COLUMN_ID + " INTEGER PRIMARY KEY, " +
            COLUMN_DESCRIPTION + " TEXT, " +
            COLUMN_NAME + " TEXT, " +
            COLUMN_VALUE + " REAL" +
            ");";

    // Create Statement for Transactions Table
    private static final String CREATE_TABLE_TRANSACTION = "CREATE TABLE " +
TABLE_TRANSACTIONS + "  (" +
            COLUMN_ID + " INTEGER PRIMARY KEY," +
            COLUMN_PRODUCT_ID + " INTEGER," +
            COLUMN_AMOUNT + " INTEGER," +
            " FOREIGN KEY (" + COLUMN_PRODUCT_ID + ") REFERENCES " + TABLE_PRODUCTS + "(" +
```

```
COLUMN_ID + ")" +
            ");";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // onCreate should always create your most up to date database
        // This method is called when the app is newly installed
        db.execSQL(CREATE_TABLE_PRODUCT);
        db.execSQL(CREATE_TABLE_TRANSACTION);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // onUpgrade is responsible for upgrading the database when you make
        // changes to the schema. For each version the specific changes you made
        // in that version have to be applied.
        for (int version = oldVersion + 1; version <= newVersion; version++) {
            switch (version) {

                case 2:
                    db.execSQL("ALTER TABLE " + TABLE_PRODUCTS + " ADD COLUMN " +
COLUMN_DESCRIPTION + " TEXT;");
                    break;

                case 3:
                    db.execSQL(CREATE_TABLE_TRANSACTION);
                    break;
            }
        }
    }
}
```

Insertar datos en la base de datos

```
// You need a writable database to insert data
final SQLiteDatabase database = openHelper.getWritableDatabase();

// Create a ContentValues instance which contains the data for each column
// You do not need to specify a value for the PRIMARY KEY column.
// Unique values for these are automatically generated.
final ContentValues values = new ContentValues();
values.put(COLUMN_NAME, model.getName());
values.put(COLUMN_DESCRIPTION, model.getDescription());
values.put(COLUMN_VALUE, model.getValue());

// This call performs the update
// The return value is the rowId or primary key value for the new row!
// If this method returns -1 then the insert has failed.
final int id = database.insert(
        TABLE_NAME, // The table name in which the data will be inserted
        null,       // String: optional; may be null. If your provided values is empty,
                    // no column names are known and an empty row can't be inserted.
                    // If not set to null, this parameter provides the name
                    // of nullable column name to explicitly insert a NULL
```

```
        values      // The ContentValues instance which contains the data
);
```

Método onUpgrade ()

`SQLiteOpenHelper` es una clase auxiliar para administrar la creación de bases de datos y la administración de versiones.

En esta clase, el método `onUpgrade()` es responsable de actualizar la base de datos cuando realiza cambios en el esquema. Se llama cuando el archivo de base de datos ya existe, pero su versión es inferior a la especificada en la versión actual de la aplicación. Para cada versión de la base de datos, se deben aplicar los cambios específicos que realizó.

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // Loop through each version when an upgrade occurs.
    for (int version = oldVersion + 1; version <= newVersion; version++) {
        switch (version) {

            case 2:
                // Apply changes made in version 2
                db.execSQL(
                    "ALTER TABLE " +
                    TABLE_PRODUCTS +
                    " ADD COLUMN " +
                    COLUMN_DESCRIPTION +
                    " TEXT;"
                );
                break;

            case 3:
                // Apply changes made in version 3
                db.execSQL(CREATE_TABLE_TRANSACTION);
                break;
        }
    }
}
```

Leyendo datos de un cursor

Este es un ejemplo de un método que viviría dentro de una subclase de `SQLiteOpenHelper` . Utiliza la cadena `searchTerm` para filtrar los resultados, recorre el contenido del cursor y devuelve esos contenidos en una `List` de objetos de `Product` .

Primero, defina la clase de `Product` POJO que será el contenedor para cada fila recuperada de la base de datos:

```
public class Product {
  long mId;
  String mName;
  String mDescription;
  float mValue;
  public Product(long id, String name, String description, float value) {
    mId = id;
```

```
    mName = name;
    mDescription = description;
    mValue = value;
  }
}
```

Luego, defina el método que consultará la base de datos y devuelva una `List` de objetos del `Product` :

```
public List<Product> searchForProducts(String searchTerm) {

    // When reading data one should always just get a readable database.
    final SQLiteDatabase database = this.getReadableDatabase();

    final Cursor cursor = database.query(
            // Name of the table to read from
            TABLE_NAME,

            // String array of the columns which are supposed to be read
            new String[]{COLUMN_NAME, COLUMN_DESCRIPTION, COLUMN_VALUE},

            // The selection argument which specifies which row is read.
            // ? symbols are parameters.
            COLUMN_NAME + " LIKE ?",

            // The actual parameters values for the selection as a String array.
            // ? above take the value from here
            new String[]{"%" + searchTerm + "%"},

            // GroupBy clause. Specify a column name to group similar values
            // in that column together.
            null,

            // Having clause. When using the GroupBy clause this allows you to
            // specify which groups to include.
            null,

            // OrderBy clause. Specify a column name here to order the results
            // according to that column. Optionally append ASC or DESC to specify
            // an ascending or descending order.
            null
    );

    // To increase performance first get the index of each column in the cursor
    final int idIndex = cursor.getColumnIndex(COLUMN_ID);
    final int nameIndex = cursor.getColumnIndex(COLUMN_NAME);
    final int descriptionIndex = cursor.getColumnIndex(COLUMN_DESCRIPTION);
    final int valueIndex = cursor.getColumnIndex(COLUMN_VALUE);

    try {

        // If moveToFirst() returns false then cursor is empty
        if (!cursor.moveToFirst()) {
            return new ArrayList<>();
        }

        final List<Product> products = new ArrayList<>();

        do {
```

```
            // Read the values of a row in the table using the indexes acquired above
            final long id = cursor.getLong(idIndex);
            final String name = cursor.getString(nameIndex);
            final String description = cursor.getString(descriptionIndex);
            final float value = cursor.getFloat(valueIndex);

            products.add(new Product(id, name, description, value));

        } while (cursor.moveToNext());

        return products;

    } finally {
        // Don't forget to close the Cursor once you are done to avoid memory leaks.
        // Using a try/finally like in this example is usually the best way to handle this
        cursor.close();

        // close the database
        database.close();
    }
}
```

Cree un contrato, asistente y proveedor para SQLite en Android

DBContract.java

```
//Define the tables and columns of your local database
public final class DBContract {
/*Content Authority its a name for the content provider, is convenient to use the package app
name to be unique on the device */

    public static final String CONTENT_AUTHORITY = "com.yourdomain.yourapp";

    //Use CONTENT_AUTHORITY to create all the database URI's that the app will use to link the
content provider.
    public static final Uri BASE_CONTENT_URI = Uri.parse("content://" + CONTENT_AUTHORITY);

    /*the name of the uri that can be the same as the name of your table.
    this will translate to content://com.yourdomain.yourapp/user/ as a valid URI
    */
    public static final String PATH_USER = "User";

    // To prevent someone from accidentally instantiating the contract class,
    // give it an empty constructor.
    public DBContract () {}

    //Intern class that defines the user table
    public static final class UserEntry implements BaseColumns {
        public static final URI CONTENT_URI =
BASE_CONTENT_URI.buildUpon().appendPath(PATH_USER).build();

        public static final String CONTENT_TYPE =
ContentResolver.CURSOR_DIR_BASE_TYPE+"/"+CONTENT_AUTHORITY+"/"+PATH_USER;

        //Name of the table
        public static final String TABLE_NAME="User";

        //Columns of the user table
        public static final String COLUMN_Name="Name";
```

```
        public static final String COLUMN_Password="Password";

        public static Uri buildUri(long id){
            return ContentUris.withAppendedId(CONTENT_URI,id);
        }
}
```

DBHelper.java

```
public class DBHelper extends SQLiteOpenHelper{

//if you change the schema of the database, you must increment this number
private static final int DATABASE_VERSION=1;
static final String DATABASE_NAME="mydatabase.db";
private static DBHelper mInstance=null;
public static DBHelper getInstance(Context ctx){
    if(mInstance==null){
    mInstance= new DBHelper(ctx.getApplicationContext());
    }
    return mInstance;
}

public DBHelper(Context context){
    super(context,DATABASE_NAME,null,DATABASE_VERSION);
}

public int GetDatabase_Version() {
    return DATABASE_VERSION;
}

@Override
public void onCreate(SQLiteDatabase sqLiteDatabase){
//Create the table users
final String SQL_CREATE_TABLE_USERS="CREATE TABLE "+UserEntry.TABLE_NAME+ " ("+
UserEntry._ID+" INTEGER PRIMARY KEY, "+
UserEntry.COLUMN_Name+" TEXT , "+
UserEntry.COLUMN_Password+" TEXT "+
" ); ";

sqLiteDatabase.execSQL(SQL_CREATE_TABLE_USERS);

}

@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {
    sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + UserEntry.TABLE_NAME);
}

}
```

DBProvider.java

```
public class DBProvider extends ContentProvider {

    private static final UriMatcher sUriMatcher = buildUriMatcher();
    private DBHelper mDBHelper;
    private Context mContext;

    static final int USER = 100;
```

```
    static UriMatcher buildUriMatcher() {

        final UriMatcher matcher = new UriMatcher(UriMatcher.NO_MATCH);
        final String authority = DBContract.CONTENT_AUTHORITY;

        matcher.addURI(authority, DBContract.PATH_USER, USER);

        return matcher;
    }

    @Override
    public boolean onCreate() {
        mDBHelper = new DBHelper(getContext());
        return false;
    }

    public PeaberryProvider(Context context) {
        mDBHelper = DBHelper.getInstance(context);
        mContext = context;
    }

    @Override
    public String getType(Uri uri) {
        // determine what type of Uri is
        final int match = sUriMatcher.match(uri);

        switch (match) {
            case USER:
                return DBContract.UserEntry.CONTENT_TYPE;

            default:
                throw new UnsupportedOperationException("Uri unknown: " + uri);
        }
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String selection, String[]
selectionArgs,
                        String sortOrder) {
        Cursor retCursor;
        try {
            switch (sUriMatcher.match(uri)) {
                case USER: {
                    retCursor = mDBHelper.getReadableDatabase().query(
                            DBContract.UserEntry.TABLE_NAME,
                            projection,
                            selection,
                            selectionArgs,
                            null,
                            null,
                            sortOrder
                    );
                    break;
                }
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
        } catch (Exception ex) {
            Log.e("Cursor", ex.toString());
        } finally {
```

```
            mDBHelper.close();
        }
        return null;
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        final SQLiteDatabase db = mDBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        Uri returnUri;
        try {

            switch (match) {
                case USER: {
                    long _id = db.insert(DBContract.UserEntry.TABLE_NAME, null, values);
                    if (_id > 0)
                        returnUri = DBContract.UserEntry.buildUri(_id);
                    else
                        throw new android.database.SQLException("Error at inserting row in " +
uri);
                    break;
                }
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
            mContext.getContentResolver().notifyChange(uri, null);
            return returnUri;
        } catch (Exception ex) {
            Log.e("Insert", ex.toString());
            db.close();
        } finally {
            db.close();
        }
        return null;
    }

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        final SQLiteDatabase db = DBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        int deletedRows;
        if (null == selection) selection = "1";
        try {
            switch (match) {
                case USER:
                    deletedRows = db.delete(
                            DBContract.UserEntry.TABLE_NAME, selection, selectionArgs);
                    break;
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
            if (deletedRows != 0) {
                mContext.getContentResolver().notifyChange(uri, null);
            }
            return deletedRows;
        } catch (Exception ex) {
            Log.e("Insert", ex.toString());
        } finally {
            db.close();
        }
        return 0;
```

```
    }

    @Override
    public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)
{
        final SQLiteDatabase db = mDBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        int updatedRows;
        try {
            switch (match) {
                case USER:
                    updatedRows = db.update(DBContract.UserEntry.TABLE_NAME, values,
selection, selectionArgs);
                    break;
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
            if (updatedRows != 0) {
                mContext.getContentResolver().notifyChange(uri, null);
            }
            return updatedRows;
        } catch (Exception ex) {
            Log.e("Update", ex.toString());
        } finally {
            db.close();
        }
        return -1;
    }

}
```

Cómo utilizar:

```
public void InsertUser() {
    try {
        ContentValues userValues = getUserData("Jhon","XXXXX");
        DBProvider dbProvider = new DBProvider(mContext);
        dbProvider.insert(UserEntry.CONTENT_URI, userValues);

    } catch (Exception ex) {
        Log.e("Insert", ex.toString());
    }
}

public ContentValues getUserData(String name, String pass) {
    ContentValues userValues = new ContentValues();
    userValues.put(UserEntry.COLUMN_Name, name);
    userValues.put(UserEntry.COLUMN_Password, pass);
    return userValues;
}
```

Actualizar una fila en una tabla

```
// You need a writable database to update a row
final SQLiteDatabase database = openHelper.getWritableDatabase();

// Create a ContentValues instance which contains the up to date data for each column
// Unlike when inserting data you need to specify the value for the PRIMARY KEY column as well
```

```
final ContentValues values = new ContentValues();
values.put(COLUMN_ID, model.getId());
values.put(COLUMN_NAME, model.getName());
values.put(COLUMN_DESCRIPTION, model.getDescription());
values.put(COLUMN_VALUE, model.getValue());

// This call performs the update
// The return value tells you how many rows have been updated.
final int count = database.update(
        TABLE_NAME,         // The table name in which the data will be updated
        values,             // The ContentValues instance with the new data
        COLUMN_ID + " = ?", // The selection which specifies which row is updated. ? symbols
are parameters.
        new String[] {      // The actual parameters for the selection as a String[].
                String.valueOf(model.getId())
        }
);
```

Realizando una Transacción

Las transacciones se pueden utilizar para realizar múltiples cambios en la base de datos de forma atómica. Cualquier transacción normal sigue este patrón:

```
// You need a writable database to perform transactions
final SQLiteDatabase database = openHelper.getWritableDatabase();

// This call starts a transaction
database.beginTransaction();

// Using try/finally is essential to reliably end transactions even
// if exceptions or other problems occur.
try {

    // Here you can make modifications to the database
    database.insert(TABLE_CARS, null, productValues);
    database.update(TABLE_BUILDINGS, buildingValues, COLUMN_ID + " = ?", new String[] {
String.valueOf(buildingId) });

    // This call marks a transaction as successful.
    // This causes the changes to be written to the database once the transaction ends.
    database.setTransactionSuccessful();
} finally {
    // This call ends a transaction.
    // If setTransactionSuccessful() has not been called then all changes
    // will be rolled back and the database will not be modified.
    database.endTransaction();
}
```

Llamar a `beginTransaction()` dentro de una transacción activa no tiene efecto.

Eliminar fila (s) de la tabla

Para borrar todas las filas de la tabla

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();
```

```
db.delete(TABLE_NAME, null, null);
db.close();
```

Para eliminar todas las filas de la tabla y obtener el recuento de la fila eliminada en valor de retorno

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

int numRowsDeleted = db.delete(TABLE_NAME, String.valueOf(1), null);
db.close();
```

Para eliminar filas con la condición `WHERE`

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

String whereClause = KEY_NAME + " = ?";
String[] whereArgs = new String[]{String.valueOf(KEY_VALUE)};

//for multiple condition, join them with AND
//String whereClause = KEY_NAME1 + " = ? AND " + KEY_NAME2 + " = ?";
//String[] whereArgs = new String[]{String.valueOf(KEY_VALUE1), String.valueOf(KEY_VALUE2)};

int numRowsDeleted = db.delete(TABLE_NAME, whereClause, whereArgs);
db.close();
```

Almacenar imagen en SQLite

Configurando la base de datos

```
public class DatabaseHelper extends SQLiteOpenHelper {
    // Database Version
    private static final int DATABASE_VERSION = 1;

    // Database Name
    private static final String DATABASE_NAME = "database_name";

    // Table Names
    private static final String DB_TABLE = "table_image";

    // column names
    private static final String KEY_NAME = "image_name";
    private static final String KEY_IMAGE = "image_data";

    // Table create statement
    private static final String CREATE_TABLE_IMAGE = "CREATE TABLE " + DB_TABLE + "("+
                        KEY_NAME + " TEXT," +
                        KEY_IMAGE + " BLOB);";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
```

```
    public void onCreate(SQLiteDatabase db) {

        // creating table
        db.execSQL(CREATE_TABLE_IMAGE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // on upgrade drop older tables
        db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);

        // create new table
        onCreate(db);
    }
}
```

Insertar en la base de datos:

```
public void addEntry( String name, byte[] image) throws SQLiteException{
    SQLiteDatabase database = this.getWritableDatabase();
    ContentValues cv = new  ContentValues();
    cv.put(KEY_NAME,     name);
    cv.put(KEY_IMAGE,     image);
    database.insert( DB_TABLE, null, cv );
}
```

Recuperando datos :

```
 byte[] image = cursor.getBlob(1);
```

Nota:

1. Antes de insertarlo en la base de datos, primero debe convertir su imagen de mapa de bits en una matriz de bytes y luego aplicarla mediante la consulta de la base de datos.
2. Al recuperar de la base de datos, ciertamente tiene una matriz de bytes de imagen, lo que debe hacer es convertir la matriz de bytes a la imagen original. Entonces, tienes que hacer uso de BitmapFactory para decodificar.

A continuación hay una clase de servicios públicos que espero les pueda ayudar:

```
public class DbBitmapUtility {

    // convert from bitmap to byte array
    public static byte[] getBytes(Bitmap bitmap) {
        ByteArrayOutputStream stream = new ByteArrayOutputStream();
        bitmap.compress(CompressFormat.PNG, 0, stream);
        return stream.toByteArray();
    }

    // convert from byte array to bitmap
    public static Bitmap getImage(byte[] image) {
        return BitmapFactory.decodeByteArray(image, 0, image.length);
    }
}
```

Crear base de datos desde la carpeta de activos

Coloque su archivo dbname.sqlite o dbname.db en la carpeta de activos de su proyecto.

```
public class Databasehelper extends SQLiteOpenHelper {
        public static final String TAG = Databasehelper.class.getSimpleName();
        public static int flag;
        // Exact Name of you db file that you put in assets folder with extension.
        static String DB_NAME = "dbname.sqlite";
        private final Context myContext;
        String outFileName = "";
        private String DB_PATH;
        private SQLiteDatabase db;

        public Databasehelper(Context context) {
            super(context, DB_NAME, null, 1);
            this.myContext = context;
            ContextWrapper cw = new ContextWrapper(context);
            DB_PATH = cw.getFilesDir().getAbsolutePath() + "/databases/";
            Log.e(TAG, "Databasehelper: DB_PATH " + DB_PATH);
            outFileName = DB_PATH + DB_NAME;
            File file = new File(DB_PATH);
            Log.e(TAG, "Databasehelper: " + file.exists());
            if (!file.exists()) {
                file.mkdir();
            }
        }

        /**
         * Creates a empty database on the system and rewrites it with your own database.
         */
        public void createDataBase() throws IOException {
            boolean dbExist = checkDataBase();
            if (dbExist) {
                //do nothing - database already exist
            } else {
                //By calling this method and empty database will be created into the default
system path
                //of your application so we are gonna be able to overwrite that database with
our database.
                this.getReadableDatabase();
                try {
                    copyDataBase();
                } catch (IOException e) {
                    throw new Error("Error copying database");
                }
            }
        }

        /**
         * Check if the database already exist to avoid re-copying the file each time you open
the application.
         *
         * @return true if it exists, false if it doesn't
         */
        private boolean checkDataBase() {
            SQLiteDatabase checkDB = null;
            try {
                checkDB = SQLiteDatabase.openDatabase(outFileName, null,
SQLiteDatabase.OPEN_READWRITE);
```

```
            } catch (SQLiteException e) {
                try {
                    copyDataBase();
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            }

            if (checkDB != null) {
                checkDB.close();
            }
            return checkDB != null ? true : false;
        }

        /**
         * Copies your database from your local assets-folder to the just created empty
database in the
         * system folder, from where it can be accessed and handled.
         * This is done by transfering bytestream.
         */

        private void copyDataBase() throws IOException {

            Log.i("Database",
                    "New database is being copied to device!");
            byte[] buffer = new byte[1024];
            OutputStream myOutput = null;
            int length;
            // Open your local db as the input stream
            InputStream myInput = null;
            try {
                myInput = myContext.getAssets().open(DB_NAME);
                // transfer bytes from the inputfile to the
                // outputfile
                myOutput = new FileOutputStream(DB_PATH + DB_NAME);
                while ((length = myInput.read(buffer)) > 0) {
                    myOutput.write(buffer, 0, length);
                }
                myOutput.close();
                myOutput.flush();
                myInput.close();
                Log.i("Database",
                        "New database has been copied to device!");
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        public void openDataBase() throws SQLException {
            //Open the database
            String myPath = DB_PATH + DB_NAME;
            db = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READWRITE);
            Log.e(TAG, "openDataBase: Open " + db.isOpen());
        }

        @Override
        public synchronized void close() {
            if (db != null)
                db.close();
            super.close();
        }
```

```
        public void onCreate(SQLiteDatabase arg0) {

        }

        @Override
        public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {

        }
    }
```

Aquí está cómo puede acceder al objeto de base de datos a su actividad.

```
// Create Databasehelper class object in your activity.
   private Databasehelper db;
```

Luego, en el método onCreate, inicialícelo y llame al método createDatabase () como se muestra a continuación.

```
db = new Databasehelper(MainActivity.this);
        try {
            db.createDataBase();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

Realice todas las operaciones de inserción, actualización, eliminación y selección como se muestra a continuación.

```
String query = "select Max(Id) as Id from " + TABLE_NAME;
        db.openDataBase();
        int count = db.getId(query);
        db.close();
```

Exportando e importando una base de datos

Es posible que desee importar y exportar su base de datos para bacukups, por ejemplo. No te olvides de los permisos.

```
public void exportDatabase(){
    try
    {
        File sd = Environment.getExternalStorageDirectory();
        File data = Environment.getDataDirectory();

        String currentDBPath = "//data//MY.PACKAGE.NAME//databases//MY_DATABASE_NAME";
        String backupDBPath = "MY_DATABASE_FILE.db";
        File currentDB = new File(data, currentDBPath);
        File backupDB = new File(sd, backupDBPath);

        FileChannel src = new FileInputStream(currentDB).getChannel();
        FileChannel dst = new FileOutputStream(backupDB).getChannel();
        dst.transferFrom(src, 0, src.size());
        src.close();
```

```
        dst.close();

        Toast.makeText(c, c.getResources().getString(R.string.exporterenToast),
Toast.LENGTH_SHORT).show();
    }
    catch (Exception e) {
        Toast.makeText(c, c.getResources().getString(R.string.portError),
Toast.LENGTH_SHORT).show();
        Log.d("Main", e.toString());
    }
}

public void importDatabase(){
    try
    {
        File sd = Environment.getExternalStorageDirectory();
        File data = Environment.getDataDirectory();

        String currentDBPath = "//data//" + "MY.PACKAGE.NAME" + "//databases//" +
"MY_DATABASE_NAME";
        String backupDBPath = "MY_DATABASE_FILE.db";
        File backupDB = new File(data, currentDBPath);
        File currentDB = new File(sd, backupDBPath);

        FileChannel src = new FileInputStream(currentDB).getChannel();
        FileChannel dst = new FileOutputStream(backupDB).getChannel();
        dst.transferFrom(src, 0, src.size());
        src.close();
        dst.close();
        Toast.makeText(c, c.getResources().getString(R.string.importerenToast),
Toast.LENGTH_LONG).show();
    }
    catch (Exception e) {
        Toast.makeText(c, c.getResources().getString(R.string.portError),
Toast.LENGTH_SHORT).show();
    }
}
```

Inserto a granel

Aquí hay un ejemplo de cómo insertar grandes porciones de datos a la vez. Todos los datos que desea insertar se recopilan dentro de una matriz ContentValues.

```
@Override
public int bulkInsert(Uri uri, ContentValues[] values) {
    int count = 0;
    String table = null;

    int uriType = IChatContract.MessageColumns.uriMatcher.match(uri);
    switch (uriType) {
        case IChatContract.MessageColumns.MESSAGES:
            table = IChatContract.MessageColumns.TABLE_NAME;
            break;
    }
    mDatabase.beginTransaction();
    try {
        for (ContentValues cv : values) {
            long rowID = mDatabase.insert(table, " ", cv);
            if (rowID <= 0) {
```

```
                throw new SQLException("Failed to insert row into " + uri);
            }
        }
        mDatabase.setTransactionSuccessful();
        getContext().getContentResolver().notifyChange(uri, null);
        count = values.length;
    } finally {
        mDatabase.endTransaction();
    }
    return count;
}
```

Y aquí hay un ejemplo de cómo usarlo:

```
ContentResolver resolver = mContext.getContentResolver();
ContentValues[] valueList = new ContentValues[object.size()];
//add whatever you like to the valueList
resolver.bulkInsert(IChatContract.MessageColumns.CONTENT_URI, valueList);
```

Lea SQLite en línea: https://riptutorial.com/es/android/topic/871/sqlite

Capítulo 235: SyncAdapter con periódicamente hacer sincronización de datos

Introducción

El componente del adaptador de sincronización en su aplicación encapsula el código para las tareas que transfieren datos entre el dispositivo y un servidor. En función de la programación y los desencadenantes que proporciona en su aplicación, el marco del adaptador de sincronización ejecuta el código en el componente del adaptador de sincronización.

Recientemente trabajé en SyncAdapter. Quiero compartir mi conocimiento con otros, puede ayudar a otros.

Examples

Adaptador de sincronización con cada valor mínimo de solicitud del servidor.

```
<provider
        android:name=".DummyContentProvider"
        android:authorities="sample.map.com.ipsyncadapter"
        android:exported="false" />

    <!-- This service implements our SyncAdapter. It needs to be exported, so that the system
    sync framework can access it. -->
    <service android:name=".SyncService"
        android:exported="true">
        <!-- This intent filter is required. It allows the system to launch our sync service
        as needed. -->
        <intent-filter>
            <action android:name="android.content.SyncAdapter" />
        </intent-filter>
        <!-- This points to a required XML file which describes our SyncAdapter. -->
        <meta-data android:name="android.content.SyncAdapter"
            android:resource="@xml/syncadapter" />
    </service>

    <!-- This implements the account we'll use as an attachment point for our SyncAdapter.
Since
    our SyncAdapter doesn't need to authenticate the current user (it just fetches a public
RSS
    feed), this account's implementation is largely empty.

    It's also possible to attach a SyncAdapter to an existing account provided by another
    package. In that case, this element could be omitted here. -->
    <service android:name=".AuthenticatorService"
        >
        <!-- Required filter used by the system to launch our account service. -->
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator" />
```

```
        </intent-filter>
        <!-- This points to an XMLf ile which describes our account service. -->
        <meta-data android:name="android.accounts.AccountAuthenticator"
            android:resource="@xml/authenticator" />
    </service>
```

Este código necesita ser agregado en el archivo de manifiesto.

En el código anterior tenemos syncservice y conteprovider y authenticatorservice.

En la aplicación, necesitamos crear el paquete xml para agregar archivos xml de syncadpter y authenticator. **authenticator.xml**

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="@string/R.String.accountType"
    android:icon="@mipmap/ic_launcher"
    android:smallIcon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    />
```

syncadapter

```
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"
    android:contentAuthority="@string/R.String.contentAuthority"
    android:accountType="@string/R.String.accountType"
    android:userVisible="true"
    android:allowParallelSyncs="true"
    android:isAlwaysSyncable="true"
    android:supportsUploading="false"/>
```

Autenticador

```
import android.accounts.AbstractAccountAuthenticator;
import android.accounts.Account;
import android.accounts.AccountAuthenticatorResponse;
import android.accounts.NetworkErrorException;
import android.content.Context;
import android.os.Bundle;

public class Authenticator extends AbstractAccountAuthenticator {
    private Context mContext;
    public Authenticator(Context context) {
        super(context);
         this.mContext=context;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse accountAuthenticatorResponse,
String s) {
        return null;
    }

    @Override
    public Bundle addAccount(AccountAuthenticatorResponse accountAuthenticatorResponse, String
s, String s1, String[] strings, Bundle bundle) throws NetworkErrorException {
        return null;
```

```
    }

    @Override
    public Bundle confirmCredentials(AccountAuthenticatorResponse
accountAuthenticatorResponse, Account account, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle getAuthToken(AccountAuthenticatorResponse accountAuthenticatorResponse,
Account account, String s, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public String getAuthTokenLabel(String s) {
        return null;
    }

    @Override
    public Bundle updateCredentials(AccountAuthenticatorResponse accountAuthenticatorResponse,
Account account, String s, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle hasFeatures(AccountAuthenticatorResponse accountAuthenticatorResponse,
Account account, String[] strings) throws NetworkErrorException {
        return null;
    }
}
```

AuthenticatorService

```
public class AuthenticatorService extends Service {

    private Authenticator authenticator;

    public AuthenticatorService() {
        super();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        IBinder ret = null;
        if (intent.getAction().equals(AccountManager.ACTION_AUTHENTICATOR_INTENT)) ;
        ret = getAuthenticator().getIBinder();
        return ret;
    }

    public Authenticator getAuthenticator() {
        if (authenticator == null)
            authenticator = new Authenticator(this);
        return authenticator;
    }
}
```

IpDataDBHelper

```
 public class IpDataDBHelper extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION=1;
    private static final String DATABASE_NAME="ip.db";
    public static final String TABLE_IP_DATA="ip";

    public static final String COLUMN_ID="_id";
    public static final String COLUMN_IP="ip";
    public static final String COLUMN_COUNTRY_CODE="country_code";
    public static final String COLUMN_COUNTRY_NAME="country_name";
    public static final String COLUMN_CITY="city";
    public static final String COLUMN_LATITUDE="latitude";
    public static final String COLUMN_LONGITUDE="longitude";

    public IpDataDBHelper(Context context, String name, SQLiteDatabase.CursorFactory factory,
int version) {
        super(context, DATABASE_NAME, factory, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        String CREATE_TABLE="CREATE TABLE " + TABLE_IP_DATA + "( " + COLUMN_ID + " INTEGER
PRIMARY KEY ,"
                + COLUMN_IP + " INTEGER ," + COLUMN_COUNTRY_CODE + " INTEGER ," +
COLUMN_COUNTRY_NAME +
                " TEXT ," + COLUMN_CITY + " TEXT ," + COLUMN_LATITUDE + " INTEGER ," +
COLUMN_LONGITUDE + " INTEGER)";
        sqLiteDatabase.execSQL(CREATE_TABLE);
        Log.d("SQL",CREATE_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + TABLE_IP_DATA);
        onCreate(sqLiteDatabase);
    }

    public long AddIPData(ContentValues values)
    {
        SQLiteDatabase sqLiteDatabase =getWritableDatabase();
        long insertedRow=sqLiteDatabase.insert(TABLE_IP_DATA,null,values);
        return insertedRow;
    }

    public Cursor getAllIpData()
    {
        String[]
projection={COLUMN_ID,COLUMN_IP,COLUMN_COUNTRY_CODE,COLUMN_COUNTRY_NAME,COLUMN_CITY,COLUMN_LATITUDE,CO

        SQLiteDatabase sqLiteDatabase =getReadableDatabase();
        Cursor cursor =
sqLiteDatabase.query(TABLE_IP_DATA,projection,null,null,null,null,null);
        return cursor;
    }

    public int deleteAllIpData()
    {
        SQLiteDatabase sqLiteDatabase=getWritableDatabase();
        int rowDeleted=sqLiteDatabase.delete(TABLE_IP_DATA,null,null);
        return rowDeleted;
    }
}
```

Actividad principal

```
     public class MainActivity extends AppCompatActivity {

        private static final String ACCOUNT_TYPE="sample.map.com.ipsyncadapter";
        private static final String AUTHORITY="sample.map.com.ipsyncadapter";
        private static final String ACCOUNT_NAME="Sync";

        public TextView mIp,mCountryCod,mCountryName,mCity,mLatitude,mLongitude;
        CursorAdapter cursorAdapter;
        Account mAccount;
        private String TAG=this.getClass().getCanonicalName();
        ListView mListView;
        public SharedPreferences mSharedPreferences;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            mListView = (ListView) findViewById(R.id.list);
            mIp=(TextView)findViewById(R.id.txt_ip);
            mCountryCod=(TextView)findViewById(R.id.txt_country_code);
            mCountryName=(TextView)findViewById(R.id.txt_country_name);
            mCity=(TextView)findViewById(R.id.txt_city);
            mLatitude=(TextView)findViewById(R.id.txt_latitude);
            mLongitude=(TextView)findViewById(R.id.txt_longitude);
            mSharedPreferences=getSharedPreferences("MyIp",0);

//Using shared preference iam displaying values in text view.
            String txtIp=mSharedPreferences.getString("ipAdr","");
            String txtCC=mSharedPreferences.getString("CCode","");
            String txtCN=mSharedPreferences.getString("CName","");
            String txtC=mSharedPreferences.getString("City","");
            String txtLP=mSharedPreferences.getString("Latitude","");
            String txtLN=mSharedPreferences.getString("Longitude","");

            mIp.setText(txtIp);
            mCountryCod.setText(txtCC);
            mCountryName.setText(txtCN);
            mCity.setText(txtC);
            mLatitude.setText(txtLP);
            mLongitude.setText(txtLN);

            mAccount=createSyncAccount(this);
//In this code i am using content provider to save data.
           /* Cursor
cursor=getContentResolver().query(MyIPContentProvider.CONTENT_URI,null,null,null,null);
            cursorAdapter=new SimpleCursorAdapter(this,R.layout.list_item,cursor,new String
[]{"ip","country_code","country_name","city","latitude","longitude"},
                                                        new int[]
{R.id.txt_ip,R.id.txt_country_code,R.id.txt_country_name,R.id.txt_city,R.id.txt_latitude,R.id.txt_longi


            mListView.setAdapter(cursorAdapter);

getContentResolver().registerContentObserver(MyIPContentProvider.CONTENT_URI,true,new
StockContentObserver(new Handler()));
    */
            Bundle settingBundle=new Bundle();
            settingBundle.putBoolean(ContentResolver.SYNC_EXTRAS_MANUAL,true);
            settingBundle.putBoolean(ContentResolver.SYNC_EXTRAS_EXPEDITED,true);
            ContentResolver.requestSync(mAccount,AUTHORITY,settingBundle);
```

```
            ContentResolver.setSyncAutomatically(mAccount,AUTHORITY,true);
            ContentResolver.addPeriodicSync(mAccount,AUTHORITY,Bundle.EMPTY,60);
        }

        private Account createSyncAccount(MainActivity mainActivity) {
            Account account=new Account(ACCOUNT_NAME,ACCOUNT_TYPE);
            AccountManager
accountManager=(AccountManager)mainActivity.getSystemService(ACCOUNT_SERVICE);
            if(accountManager.addAccountExplicitly(account,null,null))
            {

            }else
            {

            }
            return account;
        }


        private class StockContentObserver extends ContentObserver {
            @Override
            public void onChange(boolean selfChange, Uri uri) {
                Log.d(TAG, "CHANGE OBSERVED AT URI: " + uri);

cursorAdapter.swapCursor(getContentResolver().query(MyIPContentProvider.CONTENT_URI, null,
null, null, null));
            }

            public StockContentObserver(Handler handler) {
                super(handler);

            }
        }
        @Override
        protected void onResume() {
            super.onResume();
            registerReceiver(syncStaredReceiver, new IntentFilter(SyncAdapter.SYNC_STARTED));
            registerReceiver(syncFinishedReceiver, new
IntentFilter(SyncAdapter.SYNC_FINISHED));
        }

        @Override
        protected void onPause() {
            super.onPause();
            unregisterReceiver(syncStaredReceiver);
            unregisterReceiver(syncFinishedReceiver);
        }
        private BroadcastReceiver syncFinishedReceiver = new BroadcastReceiver() {

            @Override
            public void onReceive(Context context, Intent intent) {
                Log.d(TAG, "Sync finished!");
                Toast.makeText(getApplicationContext(), "Sync Finished",
Toast.LENGTH_SHORT).show();
            }
        };
        private BroadcastReceiver syncStaredReceiver = new BroadcastReceiver() {

            @Override
            public void onReceive(Context context, Intent intent) {
                Log.d(TAG, "Sync started!");
```

```
                Toast.makeText(getApplicationContext(), "Sync started...",
Toast.LENGTH_SHORT).show();
            }
        };
    }
```

MyIPContentProvider

```
public class MyIPContentProvider extends ContentProvider {

public static final int IP_DATA=1;
private static final String AUTHORITY="sample.map.com.ipsyncadapter";
private static final String TABLE_IP_DATA="ip_data";
public static final Uri CONTENT_URI=Uri.parse("content://" + AUTHORITY + '/' + TABLE_IP_DATA);
private static final UriMatcher URI_MATCHER= new UriMatcher(UriMatcher.NO_MATCH);

static
{
    URI_MATCHER.addURI(AUTHORITY,TABLE_IP_DATA,IP_DATA);
}

private IpDataDBHelper myDB;

@Override
public boolean onCreate() {
    myDB=new IpDataDBHelper(getContext(),null,null,1);
    return false;
}

@Nullable
@Override
public Cursor query(Uri uri, String[] strings, String s, String[] strings1, String s1) {
    int uriType=URI_MATCHER.match(uri);
    Cursor cursor=null;
    switch (uriType)
    {
        case IP_DATA:
            cursor=myDB.getAllIpData();
            break;
        default:
            throw new IllegalArgumentException("UNKNOWN URL");
    }
    cursor.setNotificationUri(getContext().getContentResolver(), uri);
    return cursor;
}

@Nullable
@Override
public String getType(Uri uri) {
    return null;
}

@Nullable
@Override
public Uri insert(Uri uri, ContentValues contentValues) {
    int uriType=URI_MATCHER.match(uri);
    long id=0;
    switch (uriType)
    {
        case IP_DATA:
```

```
            id=myDB.AddIPData(contentValues);
            break;
        default:
            throw new IllegalArgumentException("UNKNOWN URI :" +uri);
    }
    getContext().getContentResolver().notifyChange(uri,null);
    return Uri.parse(contentValues + "/" + id);
}

@Override
public int delete(Uri uri, String s, String[] strings) {
    int uriType=URI_MATCHER.match(uri);
    int rowsDeleted=0;

    switch (uriType)
    {
        case IP_DATA:
            rowsDeleted=myDB.deleteAllIpData();
            break;
        default:
            throw new IllegalArgumentException("UNKNOWN URI :" +uri);
    }
    getContext().getContentResolver().notifyChange(uri,null);
    return rowsDeleted;
}

@Override
public int update(Uri uri, ContentValues contentValues, String s, String[] strings) {
    return 0;
}
```

}

SyncAdapter

```
public class SyncAdapter extends AbstractThreadedSyncAdapter {
ContentResolver mContentResolver;
Context mContext;
public static final String SYNC_STARTED="Sync Started";
public static final String SYNC_FINISHED="Sync Finished";
private static final String TAG=SyncAdapter.class.getCanonicalName();
public SharedPreferences mSharedPreferences;

public SyncAdapter(Context context, boolean autoInitialize) {
    super(context, autoInitialize);
    this.mContext=context;
    mContentResolver=context.getContentResolver();
    Log.i("SyncAdapter","SyncAdapter");
}

@Override
public void onPerformSync(Account account, Bundle bundle, String s, ContentProviderClient
contentProviderClient, SyncResult syncResult) {

    Intent intent = new Intent(SYNC_STARTED);
    mContext.sendBroadcast(intent);

    Log.i(TAG, "onPerformSync");

    intent = new Intent(SYNC_FINISHED);
```

```
    mContext.sendBroadcast(intent);
    mSharedPreferences =mContext.getSharedPreferences("MyIp",0);
    SharedPreferences.Editor editor=mSharedPreferences.edit();

    mContentResolver.delete(MyIPContentProvider.CONTENT_URI,null,null);

    String data="";

    try {
        URL url =new URL("https://freegeoip.net/json/");
        Log.d(TAG, "URL :"+url);
        HttpURLConnection connection=(HttpURLConnection)url.openConnection();
        Log.d(TAG,"Connection :"+connection);
        connection.connect();
        Log.d(TAG,"Connection 1:"+connection);
        InputStream inputStream=connection.getInputStream();
        data=getInputData(inputStream);
        Log.d(TAG,"Data :"+data);

        if (data != null || !data.equals("null")) {
            JSONObject jsonObject = new JSONObject(data);


            String ipa = jsonObject.getString("ip");
            String country_code = jsonObject.getString("country_code");
            String country_name = jsonObject.getString("country_name");
            String region_code=jsonObject.getString("region_code");
            String region_name=jsonObject.getString("region_name");
            String zip_code=jsonObject.getString("zip_code");
            String time_zone=jsonObject.getString("time_zone");
            String metro_code=jsonObject.getString("metro_code");

            String city = jsonObject.getString("city");
            String latitude = jsonObject.getString("latitude");
            String longitude = jsonObject.getString("longitude");
           /* ContentValues values = new ContentValues();
            values.put("ip", ipa);
            values.put("country_code", country_code);
            values.put("country_name", country_name);
            values.put("city", city);
            values.put("latitude", latitude);
            values.put("longitude", longitude);*/
            //Using cursor adapter for results.
            //mContentResolver.insert(MyIPContentProvider.CONTENT_URI, values);

            //Using Shared preference for results.
            editor.putString("ipAdr",ipa);
            editor.putString("CCode",country_code);
            editor.putString("CName",country_name);
            editor.putString("City",city);
            editor.putString("Latitude",latitude);
            editor.putString("Longitude",longitude);
            editor.commit();

        }
        }catch(Exception e){
            e.printStackTrace();
        }

}
```

```
private String getInputData(InputStream inputStream) throws IOException {
    StringBuilder builder=new StringBuilder();
    BufferedReader bufferedReader=new BufferedReader(new InputStreamReader(inputStream));
    //String data=null;
    /*Log.d(TAG,"Builder 2:"+ bufferedReader.readLine());
    while ((data=bufferedReader.readLine())!= null);
    {
        builder.append(data);
        Log.d(TAG,"Builder :"+data);
    }
    Log.d(TAG,"Builder 1 :"+data);
    bufferedReader.close();*/
    String data=bufferedReader.readLine();
    bufferedReader.close();
    return data.toString();
}
```

}

SyncService

```
public class SyncService extends Service {
private static SyncAdapter syncAdapter=null;
private static final Object syncAdapterLock=new Object();

@Override
public void onCreate() {
    synchronized (syncAdapterLock)
    {
        if(syncAdapter==null)
        {
            syncAdapter =new SyncAdapter(getApplicationContext(),true);
        }
    }
}

@Nullable
@Override
public IBinder onBind(Intent intent) {
    return syncAdapter.getSyncAdapterBinder();
}
```

}

Lea SyncAdapter con periódicamente hacer sincronización de datos en línea: https://riptutorial.com/es/android/topic/10774/syncadapter-con-periodicamente-hacer-sincronizacion-de-datos

Capítulo 236: TabLayout

Examples

Usando un TabLayout sin un ViewPager

La mayoría de las veces se utiliza un `TabLayout` junto con un ViewPager para obtener la funcionalidad de deslizamiento que viene con él.

Es posible usar un `TabLayout` sin un `ViewPager` usando un `TabLayout.OnTabSelectedListener` .

Primero, agregue un `TabLayout` al archivo XML de su actividad:

```
<android.support.design.widget.TabLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:id="@+id/tabLayout" />
```

Para navegar dentro de una `Activity` , rellene manualmente la interfaz de usuario según la pestaña seleccionada.

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.tabLayout);
tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
    @Override
    public void onTabSelected(TabLayout.Tab tab) {
        int position = tab.getPosition();
        switch (tab.getPosition()) {
            case 1:
                getSupportFragmentManager().beginTransaction()
                    .replace(R.id.fragment_container, new ChildFragment()).commit();
                break;
            // Continue for each tab in TabLayout
    }

    @Override
    public void onTabUnselected(TabLayout.Tab tab) {

    }

    @Override
    public void onTabReselected(TabLayout.Tab tab) {

    }
});
```

Lea TabLayout en línea: https://riptutorial.com/es/android/topic/7601/tablayout

Capítulo 237: Tarjeta electrónica

Examples

Tarjeta inteligente de envío y recepción.

Para la conexión, aquí hay un fragmento de código para ayudarlo a comprender:

```
//Allows you to enumerate and communicate with connected USB devices.
UsbManager mUsbManager = (UsbManager) getSystemService(Context.USB_SERVICE);
//Explicitly asking for permission
final String ACTION_USB_PERMISSION = "com.android.example.USB_PERMISSION";
PendingIntent mPermissionIntent = PendingIntent.getBroadcast(this, 0, new
Intent(ACTION_USB_PERMISSION), 0);
HashMap<String, UsbDevice> deviceList = mUsbManager.getDeviceList();

UsbDevice device = deviceList.get("//the device you want to work with");
if (device != null) {
    mUsbManager.requestPermission(device, mPermissionIntent);
}
```

Ahora debe comprender que, en java, la comunicación se realiza mediante el paquete javax.smarcard, que no está disponible para Android, por lo que puede obtener una idea de cómo puede comunicarse o enviar / recibir APDU (comando de tarjeta inteligente).

Ahora como se dice en la respuesta mencionada anteriormente

No puede simplemente enviar una APDU (comando de tarjeta inteligente) a través del punto final de carga y esperar recibir una APDU de respuesta a través del punto extremo de entrada. Para obtener los puntos finales, vea el fragmento de código a continuación:

```
UsbEndpoint epOut = null, epIn = null;
UsbInterface usbInterface;

UsbDeviceConnection connection = mUsbManager.openDevice(device);

    for (int i = 0; i < device.getInterfaceCount(); i++) {
        usbInterface = device.getInterface(i);
        connection.claimInterface(usbInterface, true);

        for (int j = 0; j < usbInterface.getEndpointCount(); j++) {
            UsbEndpoint ep = usbInterface.getEndpoint(j);

            if (ep.getType() == UsbConstants.USB_ENDPOINT_XFER_BULK) {
                if (ep.getDirection() == UsbConstants.USB_DIR_OUT) {
                    // from host to device
                    epOut = ep;

                } else if (ep.getDirection() == UsbConstants.USB_DIR_IN) {
                    // from device to host
                    epIn = ep;
                }
            }
```

```
        }
    }
```

Ahora tiene los puntos finales de entrada y salida para enviar y recibir comandos de APDU y bloques de respuesta de APDU:

Para enviar comandos, vea el fragmento de código a continuación:

```
public void write(UsbDeviceConnection connection, UsbEndpoint epOut, byte[] command) {
    result = new StringBuilder();
    connection.bulkTransfer(epOut, command, command.length, TIMEOUT);
    //For Printing logs you can use result variable
    for (byte bb : command) {
        result.append(String.format(" %02X ", bb));
    }
}
```

Y para recibir / leer una respuesta, vea el fragmento de código a continuación:

```
public int read(UsbDeviceConnection connection, UsbEndpoint epIn) {
result = new StringBuilder();
final byte[] buffer = new byte[epIn.getMaxPacketSize()];
int byteCount = 0;
byteCount = connection.bulkTransfer(epIn, buffer, buffer.length, TIMEOUT);

//For Printing logs you can use result variable
if (byteCount >= 0) {
    for (byte bb : buffer) {
        result.append(String.format(" %02X ", bb));
    }

    //Buffer received was : result.toString()
} else {
    //Something went wrong as count was : " + byteCount
}

return byteCount;
}
```

Ahora, si ve esta respuesta aquí, el primer comando a enviar es:

PC_to_RDR_IccPowerOn comando para activar la tarjeta.

que puede crear leyendo la sección 6.1.1 de la documentación de Especificaciones de clase de dispositivo USB aquí.

Ahora vamos a tomar un ejemplo de este comando como el que se encuentra aquí: `62000000000000000000` Cómo puede enviar esto:

```
write(connection, epOut, "62000000000000000000");
```

Ahora, después de haber enviado con éxito el comando APDU, puede leer la respuesta usando:

```
read(connection, epIn);
```

Y recibir algo como

```
80 18000000 00 00 00 00 00 3BBF11008131FE45455041000000000000000000000000F1
```

Ahora, la respuesta recibida en el código aquí estará en la variable de `result` del método `read()` del código

Lea Tarjeta electrónica en línea: https://riptutorial.com/es/android/topic/10945/tarjeta-electronica

Capítulo 238: Teclado

Examples

Oculta el teclado cuando el usuario toca cualquier otro lugar en la pantalla

Añade código en tu **actividad** .

Esto también funcionaría para **Fragmento** , **no es necesario** agregar este código en **Fragmento** .

```
@Override
public boolean dispatchTouchEvent(MotionEvent ev) {
    View view = getCurrentFocus();
    if (view != null && (ev.getAction() == MotionEvent.ACTION_UP || ev.getAction() ==
MotionEvent.ACTION_MOVE) && view instanceof EditText &&
!view.getClass().getName().startsWith("android.webkit.")) {
        int scrcoords[] = new int[2];
        view.getLocationOnScreen(scrcoords);
        float x = ev.getRawX() + view.getLeft() - scrcoords[0];
        float y = ev.getRawY() + view.getTop() - scrcoords[1];
        if (x < view.getLeft() || x > view.getRight() || y < view.getTop() || y >
view.getBottom())

((InputMethodManager)this.getSystemService(Context.INPUT_METHOD_SERVICE)).hideSoftInputFromWindow((this
0);
    }
    return super.dispatchTouchEvent(ev);
}
```

Registrar una devolución de llamada para abrir y cerrar el teclado

La idea es medir un diseño antes y después de cada cambio y si hay un cambio significativo, puede estar algo seguro de que es la tecla programable.

```
// A variable to hold the last content layout hight
private int mLastContentHeight = 0;

private ViewTreeObserver.OnGlobalLayoutListener keyboardLayoutListener = new
ViewTreeObserver.OnGlobalLayoutListener() {
    @Override public void onGlobalLayout() {
        int currentContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();

        if (mLastContentHeight > currentContentHeight + 100) {
            Timber.d("onGlobalLayout: Keyboard is open");
            mLastContentHeight = currentContentHeight;
        } else if (currentContentHeight > mLastContentHeight + 100) {
            Timber.d("onGlobalLayout: Keyboard is closed");
            mLastContentHeight = currentContentHeight;
        }
    }
};
```

luego, en nuestro `onCreate` establezca el valor inicial de `mLastContentHeight`

```
mLastContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();
```

y agregar el oyente

```
rootView.getViewTreeObserver().addOnGlobalLayoutListener(keyboardLayoutListener);
```

No olvides eliminar al oyente al `destroy`

```
rootView.getViewTreeObserver().removeOnGlobalLayoutListener(keyboardLayoutListener);
```

Lea Teclado en línea: https://riptutorial.com/es/android/topic/5606/teclado

Capítulo 239: Tema DayNight (AppCompat v23.2 / API 14+)

Examples

Adición del tema DayNight a una aplicación

El tema DayNight le da a una aplicación la genial capacidad de cambiar los esquemas de color según la hora del día y la última ubicación conocida del dispositivo.

Agregue lo siguiente a su `styles.xml` :

```
<style name="AppTheme" parent="Theme.AppCompat.DayNight">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
</style>
```

Los temas que puede ampliar para agregar la capacidad de cambio de tema de día por la noche son los siguientes:

- `"Theme.AppCompat.DayNight"`
- `"Theme.AppCompat.DayNight.NoActionBar"`
- `"Theme.AppCompat.DayNight.DarkActionBar"`

Además de `colorPrimary` , `colorPrimaryDark` y `colorAccent` , también puede agregar cualquier otro color que desee cambiar, por ejemplo, `textColorPrimary` o `textColorSecondary` . También puede agregar los colores personalizados de su aplicación a este `style` .

Para que el cambio de tema funcione, debe definir un `colors.xml` predeterminado en el directorio `res/values` y otro `colors.xml` en el directorio `res/values-night` y definir adecuadamente los colores día / noche.

Para cambiar el tema, llame al `AppCompatDelegate.setDefaultNightMode(int)` desde su código Java. (Esto cambiará el esquema de color para toda la aplicación, no solo una actividad o fragmento). Por ejemplo:

```
AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
```

Puede pasar cualquiera de los siguientes tres de acuerdo a su elección:

- `AppCompatDelegate.MODE_NIGHT_NO` : esto establece el tema predeterminado para su aplicación y toma los colores definidos en el directorio `res/values` . Se recomienda utilizar colores claros para este tema.
- `AppCompatDelegate.MODE_NIGHT_YES` : establece un tema nocturno para su aplicación y toma los colores definidos en el directorio `res/values-night` . Se recomienda utilizar colores oscuros

para este tema.

- `AppCompatDelegate.MODE_NIGHT_AUTO` : este cambia automáticamente los colores de la aplicación según la hora del día y los colores que haya definido en los directorios de `values` y `values-night` .

También es posible obtener el estado actual del modo nocturno utilizando el método `getDefaultNightMode()` . Por ejemplo:

```
int modeType = AppCompatDelegate.getDefaultNightMode();
```

Sin embargo, tenga en cuenta que el cambio de tema no persistirá si mata la aplicación y la vuelve a abrir. Si lo hace, el tema volverá a `AppCompatDelegate.MODE_NIGHT_AUTO` , que es el valor predeterminado. Si desea que el cambio de tema persista, asegúrese de almacenar el valor en las preferencias compartidas y de cargar el valor almacenado cada vez que se abra la aplicación después de que se haya destruido.

Lea Tema DayNight (AppCompat v23.2 / API 14+) en línea: https://riptutorial.com/es/android/topic/7650/tema-daynight--appcompat-v23-2---api-14plus-

Capítulo 240: Tema, Estilo, Atributo

Examples

Usa un tema personalizado a nivel mundial

En themes.xml:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

En AndroidManifest.xml:

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">

    <!-- Activity declarations here -->

</application>
```

Definir colores primarios, primarios oscuros y de acento.

Puedes personalizar la paleta de colores de tu tema .

Usando APIs de **framework**

5.0

```
<style name="AppTheme" parent="Theme.Material">
    <item name="android:colorPrimary">@color/primary</item>
    <item name="android:colorPrimaryDark">@color/primary_dark</item>
    <item name="android:colorAccent">@color/accent</item>
</style>
```

Uso de la **biblioteca de soporte de Appcompat** (y `AppCompatActivity`)

2.1.x

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="colorPrimary">@color/primary</item>
    <item name="colorPrimaryDark">@color/primary_dark</item>
    <item name="colorAccent">@color/accent</item>
</style>
```

Usar tema personalizado por actividad

En themes.xml:

```
<style name="MyActivityTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

En AndroidManifest.xml:

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/Theme.AppCompat">

    <activity
        android:name=".MyActivity"
        android:theme="@style/MyActivityTheme" />

</application>
```

Color de desplazamiento superior (API 21+)

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorEdgeEffect">@color/my_color</item>
</style>
```

Color de ondulación (API 21+)

5.0

La animación de rizado se muestra cuando el usuario presiona vistas seleccionables.

Puede usar el mismo color de ondulación que utiliza su aplicación al asignar `?android:colorControlHighlight` en sus vistas. Puede personalizar este color cambiando el atributo `android:colorControlHighlight` en su tema:

Este efecto de color se puede cambiar:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorControlHighlight">@color/my_color</item>
</style>
```

O, si está utilizando un tema material:

```
<style name="AppTheme" parent="android:Theme.Material.Light">
    <item name="android:colorControlHighlight">@color/your_custom_color</item>
</style>
```

Barra de estado de luz (API 23+)

Este atributo puede cambiar el fondo de los iconos de la barra de estado (en la parte superior de

la pantalla) a blanco.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowLightStatusBar">true</item>
</style>
```

Navegación translúcida y barras de estado (API 19+)

La barra de navegación (en la parte inferior de la pantalla) puede ser transparente. Aquí está la manera de lograrlo.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentNavigation">true</item>
</style>
```

La barra de estado (parte superior de la pantalla) se puede hacer transparente, aplicando este atributo al estilo:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentStatus">true</item>
</style>
```

Color de la barra de navegación (API 21+)

5.0

Este atributo se usa para cambiar la barra de navegación (una, que contiene Atrás, botón Inicio reciente). Por lo general, es negro, sin embargo su color puede ser cambiado.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:navigationBarColor">@color/my_color</item>
</style>
```

Herencia del tema

Al definir los temas, uno usualmente usa el tema provisto por el sistema, y luego los cambios modifican el aspecto para que se ajuste a su propia aplicación. Por ejemplo, así es como se `Theme.AppCompat` tema `Theme.AppCompat` :

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

Este tema ahora tiene todas las propiedades del tema `Theme.AppCompat` estándar, excepto los que cambiamos explícitamente.

También hay un atajo cuando se hereda, usualmente usado cuando uno hereda de su propio

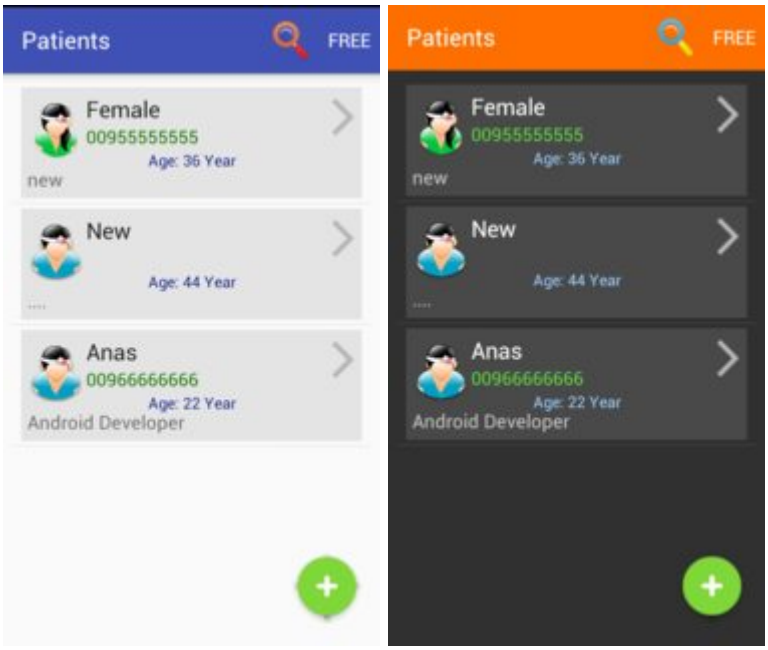
tema:

```
<style name="AppTheme.Red">
    <item name="colorAccent">@color/red</item>
</style>
```

Ya que ya tiene `AppTheme.` al comienzo de su nombre, lo hereda automáticamente, sin necesidad de definir el tema `parent` . Esto es útil cuando necesita crear estilos específicos para una parte (por ejemplo, una sola actividad) de su aplicación.

Temas múltiples en una aplicación

Usando más de un tema en su aplicación de Android, puede agregar colores personalizados a cada tema, para ser así:



Primero, tenemos que agregar nuestros temas a `style.xml` como este:

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">

</style>

<!--  -->
<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >

</style>
......
```

Arriba puedes ver **OneTheme** y **TwoTheme** .

Ahora, vaya a su `AndroidManifest.xml` y agregue esta línea: `android:theme="@style/OneTheme"` a su etiqueta de *aplicación* , esto hará que **OneTheme sea** el tema predeterminado:

```
<application
```

```
        android:theme="@style/OneTheme"
        ...>
```

Cree un nuevo archivo xml llamado `attrs.xml` y agregue este código:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <attr name="custom_red" format="color" />
    <attr name="custom_blue" format="color" />
    <attr name="custom_green" format="color" />
</resources>
<!-- add all colors you need (just color's name) -->
```

Vuelva a `style.xml` y agregue estos colores con sus valores para cada tema:

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="custom_red">#8b030c</item>
    <item name="custom_blue">#0f1b8b</item>
    <item name="custom_green">#1c7806</item>
</style>

<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >
    <item name="custom_red">#ff606b</item>
    <item name="custom_blue">#99cfff</item>
    <item name="custom_green">#62e642</item>
</style>
```

Ahora que tiene colores personalizados para cada tema, agreguemos estos colores a nuestras vistas.

Agregue color **custom_blue** al TextView usando "? Attr /":

Ve a tu imagen y añade este color:

```
<TextView>
    android:id="@+id/txte_view"
    android:textColor="?attr/custom_blue" />
```

Mow podemos cambiar el tema solo con una sola línea `setTheme(R.style.TwoTheme);` esta línea debe ser antes `setContentView()` método en el `onCreate()` método, como este `Activity.java` :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setTheme(R.style.TwoTheme);
    setContentView(R.layout.main_activity);
    ....
}
```

Cambio de tema para todas las actividades a

la vez.

Si queremos cambiar el tema para todas las actividades, tenemos que crear una nueva clase llamada MyActivity extiende la clase `AppCompatActivity` (o clase de `Activity`) y agregue la línea `setTheme(R.style.TwoTheme);` al método **onCreate ()** :

```
public class MyActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (new MySettings(this).isDarkTheme())
            setTheme(R.style.TwoTheme);
    }
}
```

Finalmente, vaya a todas sus actividades y añada todos ellos a la clase base **MyActivity** :

```
public class MainActivity extends MyActivity {
    ....
}
```

Para cambiar el tema, solo vaya a **MyActivity** y cambie `R.style.TwoTheme` a su tema (`R.style.OneTheme` , `R.style.ThreeTheme` ....).

Lea Tema, Estilo, Atributo en línea: https://riptutorial.com/es/android/topic/1843/tema--estilo--atributo

Capítulo 241: TensorFlow

Introducción

TensorFlow fue diseñado teniendo en cuenta las plataformas móviles e integradas. Tenemos código de ejemplo y soporte de compilación que puede probar ahora para estas plataformas:

Android iOS Raspberry Pi

Observaciones

Trabajo apreciado por MindRocks

Examples

Cómo utilizar

Instala Bazel desde aquí . Bazel es el sistema de compilación principal para TensorFlow. Ahora, edite WORKSPACE, podemos encontrar el archivo WORKSPACE en el directorio raíz de TensorFlow que hemos clonado anteriormente.

```
# Uncomment and update the paths in these entries to build the Android demo.
#android_sdk_repository(
#    name = "androidsdk",
#    api_level = 23,
#    build_tools_version = "25.0.1",
#    # Replace with path to Android SDK on your system
#    path = "<PATH_TO_SDK>",
#)
#
#android_ndk_repository(
#    name="androidndk",
#    path="<PATH_TO_NDK>",
#    api_level=14)
```

Como a continuación con nuestra ruta sdk y ndk:

```
android_sdk_repository(
    name = "androidsdk",
    api_level = 23,
    build_tools_version = "25.0.1",
    # Replace with path to Android SDK on your system
    path = "/Users/amitshekhar/Library/Android/sdk/",
)
android_ndk_repository(
    name="androidndk",
    path="/Users/amitshekhar/Downloads/android-ndk-r13/",
    api_level=14)
```

Lea TensorFlow en línea: https://riptutorial.com/es/android/topic/9991/tensorflow

Capítulo 242: TextInputLayout

Introducción

TextInputLayout se introdujo para mostrar la etiqueta flotante en EditText. TextInputLayout debe incluir EditText para mostrar la etiqueta flotante.

Observaciones

`TextInputLayout` es un diseño que envuelve un `EditText` (o descendiente) para mostrar una etiqueta flotante cuando la sugerencia está oculta debido a que el usuario ingresa texto. Además, `TextInputLayout` permite mostrar un mensaje de error debajo de `EditText` .

Asegúrese de que la siguiente dependencia se agregue al archivo `build.gradle` su aplicación en las dependencias:

```
compile 'com.android.support:design:25.3.1'
```

Examples

Uso básico

Es el uso básico de `TextInputLayout` .
Asegúrese de agregar la dependencia en el archivo `build.gradle` como se describe en la sección de comentarios.

Ejemplo:

```
<android.support.design.widget.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

    <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/username"/>

</android.support.design.widget.TextInputLayout>
```

Errores de manejo

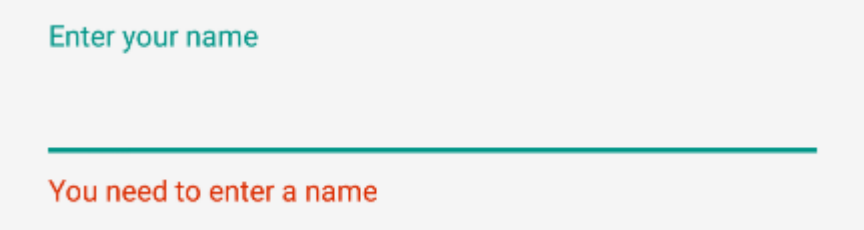
Puede usar `TextInputLayout` para mostrar mensajes de error de acuerdo con las pautas de diseño del material utilizando los métodos `setError` y `setErrorEnabled` .

Para mostrar el error debajo del uso EditText:

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setErrorEnabled(true);
til.setError("You need to enter a name");
```

Para habilitar el error en `TextInputLayout` , puede utilizar la `app:errorEnabled="true"` en xml o `til.setErrorEnabled(true);` como se muestra arriba.

Obtendrás:



Agregando el conteo de personajes

TextInputLayout tiene un contador de caracteres para un EditText definido dentro de él.
El contador se renderizará debajo de EditText.

Solo use los `setCounterEnabled()` y `setCounterMaxLength` :

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setCounterEnabled(true);
til.setCounterMaxLength(15);
```

o los `app:counterEnabled` y `app:counterMaxLength` en el xml.

```
<android.support.design.widget.TextInputLayout
    app:counterEnabled="true"
    app:counterMaxLength="15">

    <EditText/>

</android.support.design.widget.TextInputLayout>
```

La visibilidad de la contraseña cambia

Con un tipo de contraseña de entrada, también puede habilitar un icono que puede mostrar u ocultar todo el texto usando el atributo `passwordToggleEnabled` .

También puede personalizar el mismo valor predeterminado utilizando estos atributos:

- `passwordToggleDrawable` : para cambiar el icono de ojo predeterminado
- `passwordToggleTint` : para aplicar un tinte a la visibilidad de la contraseña, se puede alternar dibujable.
- `passwordToggleTintMode` : para especificar el modo de fusión utilizado para aplicar el tinte de fondo.

Ejemplo:

```
<android.support.design.widget.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:passwordToggleContentDescription="@string/description"
        app:passwordToggleDrawable="@drawable/another_toggle_drawable"
        app:passwordToggleEnabled="true">

            <EditText/>

</android.support.design.widget.TextInputLayout>
```

TextInputEditText

`TextInputEditText` es un `EditText` con una solución adicional para mostrar una sugerencia en el IME cuando está en el modo 'extraer' .

El **modo Extraer** es el modo al que cambia el editor de teclado cuando hace clic en un texto de edición cuando el espacio es demasiado pequeño (por ejemplo, paisaje en un teléfono inteligente).
En este caso, el uso de un `EditText` mientras se está editando el texto se puede ver que el IME no le da una pista de lo que está editando

`TextInputEditText` soluciona este problema al proporcionar un texto de sugerencia mientras el IME del dispositivo del usuario está en modo Extraer.

Ejemplo:

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Description"
    >
    <android.support.design.widget.TextInputEditText
        android:id="@+id/description"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

</android.support.design.widget.TextInputLayout>
```

Personalizando la apariencia de TextInputLayout

Puede personalizar la apariencia de `TextInputLayout` y su `EditText` incrustado definiendo estilos personalizados en su `styles.xml` . Los estilos definidos se pueden agregar como estilos o temas a su `TextInputLayout` .

Ejemplo para personalizar el aspecto de la pista:

`styles.xml` :

```
<!--Floating label text style-->
<style name="MyHintStyle" parent="TextAppearance.AppCompat.Small">
    <item name="android:textColor">@color/black</item>
</style>
```

```
<!--Input field style-->
<style name="MyEditText" parent="Theme.AppCompat.Light">
    <item name="colorControlNormal">@color/indigo</item>
    <item name="colorControlActivated">@color/pink</item>
</style>
```

Para aplicar el estilo, actualice su TextInputLayout y EditText de la siguiente manera

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:hintTextAppearance="@style/MyHintStyle">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/Title"
        android:theme="@style/MyEditText" />

</android.support.design.widget.TextInputLayout>
```

Ejemplo para personalizar el color de acento de `TextInputLayout` . El color de acento afecta el color de la línea de base del texto de `EditText` y el color del texto del texto de sugerencia flotante:

`styles.xml` :

```
<style name="TextInputLayoutWithPrimaryColor" parent="Widget.Design.TextInputLayout">
    <item name="colorAccent">@color/primary</item>
</style>
```

archivo de diseño:

```
<android.support.design.widget.TextInputLayout
            android:id="@+id/textInputLayout_password"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:theme="@style/TextInputLayoutWithPrimaryColor">

            <android.support.design.widget.TextInputEditText
                android:id="@+id/textInputEditText_password"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="@string/login_hint_password"
                android:inputType="textPassword" />

</android.support.design.widget.TextInputLayout>
```

Lea TextInputLayout en línea: https://riptutorial.com/es/android/topic/5652/textinputlayout

Capítulo 243: Texto a voz (TTS)

Examples

Base de texto a voz

layout_text_to_speech.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter text here!"
        android:id="@+id/textToSpeak"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@id/textToSpeak"
        android:id="@+id/btnSpeak"/>

</RelativeLayout>
```

AndroidTextToSpeechActivity.java

```
public class AndroidTextToSpeechActivity extends Activity implements
        TextToSpeech.OnInitListener {

    EditText textToSpeak = null;
    Button btnSpeak = null;
    TextToSpeech tts;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        textToSpeak = findViewById(R.id.textToSpeak);
        btnSpeak = findViewById(R.id.btnSpeak);
        btnSpeak.setEnabled(false);
        tts = new TextToSpeech(this, this);
        btnSpeak.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    speakOut();
                }
            });
    }

    @Override
    public void onDestroy() {
        // Don't forget to shutdown tts!
```

```
        if (tts != null) {
            tts.stop();
            tts.shutdown();
        }
        super.onDestroy();
    }

    @Override
    public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {
            int result = tts.setLanguage(Locale.US);

            if (result == TextToSpeech.LANG_MISSING_DATA
                    || result == TextToSpeech.LANG_NOT_SUPPORTED) {
                Log.e("TTS", "This Language is not supported");
            } else {
                btnSpeak.setEnabled(true);
                speakOut();
            }
        } else {
            Log.e("TTS", "Initilization Failed!");
        }
    }

    private void speakOut() {
        String text = textToSpeak.getText().toString();
        if(text == null || text.isEmpty())
            return;

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            String utteranceId=this.hashCode() + "";
            tts.speak(text, TextToSpeech.QUEUE_FLUSH, null, utteranceId);
        } else {
            tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);
        }
    }
}
```

El idioma que se va a hablar se puede establecer al proporcionar una configuración `Locale` al método `setLanguage()` :

```
tts.setLanguage(Locale.CHINESE); // Chinese language
```

El número de idiomas admitidos varía entre los niveles de Android. El método `isLanguageAvailable()` se puede usar para verificar si un determinado idioma es compatible:

```
tts.isLanguageAvailable(Locale.CHINESE);
```

El nivel de tono de voz se puede establecer utilizando el método `setPitch()` . Por defecto, el valor del tono es 1.0. Use valores menores que 1.0 para disminuir el nivel de tono o valores mayores que 1.0 para aumentar el nivel de tono:

```
tts.setPitch(0.6);
```

La velocidad de voz se puede configurar utilizando `setSpeechRate()` . La velocidad de voz

predeterminada es 1.0. La velocidad de voz se puede duplicar configurándola en 2.0 o en la mitad configurándola en 0.5:

```
tts.setSpeechRate(2.0);
```

Implementación de TextToSpeech en las APIs.

La implementación observable en frío, emite verdadero cuando el motor TTS termina de hablar, comienza a hablar cuando se suscribe. Tenga en cuenta que el nivel 21 de la API introduce una forma diferente de actuar:

```
public class RxTextToSpeech {

@Nullable RxTTSObservableOnSubscribe audio;

WeakReference<Context> contextRef;

public RxTextToSpeech(Context context) {
    this.contextRef = new WeakReference<>(context);
}

public void requestTTS(FragmentActivity activity, int requestCode) {
    Intent checkTTSIntent = new Intent();
    checkTTSIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
    activity.startActivityForResult(checkTTSIntent, requestCode);
}

public void cancelCurrent() {
    if (audio != null) {
        audio.dispose();
        audio = null;
    }
}

public Observable<Boolean> speak(String textToRead) {
    audio = new RxTTSObservableOnSubscribe(contextRef.get(), textToRead, Locale.GERMANY);
    return Observable.create(audio);
}


public static class RxTTSObservableOnSubscribe extends UtteranceProgressListener
        implements ObservableOnSubscribe<Boolean>,
        Disposable, Cancellable, TextToSpeech.OnInitListener {

    volatile boolean disposed;
    ObservableEmitter<Boolean> emitter;
    TextToSpeech textToSpeech;
    String text = "";
    Locale selectedLocale;
    Context context;

    public RxTTSObservableOnSubscribe(Context context, String text, Locale locale) {
        this.selectedLocale = locale;
        this.context = context;
        this.text = text;
    }
```

```
    @Override public void subscribe(ObservableEmitter<Boolean> e) throws Exception {
        this.emitter = e;
        if (context == null) {
            this.emitter.onError(new Throwable("nullable context, cannot execute " + text));
        } else {
            this.textToSpeech = new TextToSpeech(context, this);
        }
    }

    @Override @DebugLog public void dispose() {
        if (textToSpeech != null) {
            textToSpeech.setOnUtteranceProgressListener(null);
            textToSpeech.stop();
            textToSpeech.shutdown();
            textToSpeech = null;
        }
        disposed = true;
    }

    @Override public boolean isDisposed() {
        return disposed;
    }

    @Override public void cancel() throws Exception {
        dispose();
    }

    @Override public void onInit(int status) {

        int languageCode = textToSpeech.setLanguage(selectedLocale);

        if (languageCode == android.speech.tts.TextToSpeech.LANG_COUNTRY_AVAILABLE) {
            textToSpeech.setPitch(1);
            textToSpeech.setSpeechRate(1.0f);
            textToSpeech.setOnUtteranceProgressListener(this);
            performSpeak();
        } else {
            emitter.onError(new Throwable("language " + selectedLocale.getCountry() + " is not
supported"));
        }
    }

    @Override public void onStart(String utteranceId) {
        //no-op
    }

    @Override public void onDone(String utteranceId) {
        this.emitter.onNext(true);
        this.emitter.onComplete();
    }

    @Override public void onError(String utteranceId) {
        this.emitter.onError(new Throwable("error TTS " + utteranceId));
    }

    void performSpeak() {

        if (isAtLeastApiLevel(21)) {
            speakWithNewApi();
        } else {
            speakWithOldApi();
```

```
        }
    }

    @RequiresApi(api = 21) void speakWithNewApi() {
        Bundle params = new Bundle();
        params.putString(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID, "");
        textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, params, uniqueId());
    }

    void speakWithOldApi() {
        HashMap<String, String> map = new HashMap<>();
        map.put(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID, uniqueId());
        textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, map);
    }

    private String uniqueId() {
        return UUID.randomUUID().toString();
    }
}

public static boolean isAtLeastApiLevel(int apiLevel) {
    return Build.VERSION.SDK_INT >= apiLevel;
}
```

}

Lea Texto a voz (TTS) en línea: https://riptutorial.com/es/android/topic/3381/texto-a-voz--tts-

Capítulo 244: tostada

Introducción

Un Toast proporciona comentarios simples sobre una operación en una pequeña ventana emergente y desaparece automáticamente después de un tiempo de espera. Solo llena la cantidad de espacio requerido para el mensaje y la actividad actual permanece visible e interactiva.

Sintaxis

- Toast makeText (contexto de contexto, texto CharSequence, duración int)
- Toast makeText (contexto de contexto, int resId, int duration)
- void setGravity (int gravity, int xOffset, int yOffset)
- espectáculo nulo ()

Parámetros

Parámetro	Detalles
contexto	El contexto para mostrar su Toast. `this` se usa comúnmente en una Actividad y `getActivity()` se usa comúnmente en un Fragmento
texto	Una secuencia de caracteres que especifica qué texto se mostrará en el Toast. Se puede usar cualquier objeto que implemente CharSequence, incluido un String
resuelto	Un ID de recurso que se puede usar para proporcionar una Cadena de recursos para mostrar en el Toast
duración	Bandera de enteros que representa la duración del Toast. Las opciones son `Toast.LENGTH_SHORT` y `Toast.LENGTH_LONG`
gravedad	Entero que especifica la posición o "gravedad" de la tostada. Ver opciones aquí
xOffset	Especifica el desplazamiento horizontal para la posición Toast.
yOffset	Especifica el desplazamiento vertical para la posición Toast.

Observaciones

Un brindis proporciona información simple sobre una operación en una pequeña ventana emergente. Solo llena la cantidad de espacio requerido para el mensaje y la actividad actual permanece visible e interactiva.

Una alternativa más reciente a Toast es SnackBar. SnackBar ofrece un estilo visual actualizado y permite al usuario descartar el mensaje o tomar medidas adicionales. Consulte la documentación de SnackBar para más detalles.

Documentación oficial:

https://developer.android.com/reference/android/widget/Toast.html

Examples

Establecer posición de una tostada

Aparece una notificación estándar de tostadas en la parte inferior de la pantalla alineada en el centro horizontal. Puede cambiar esta posición con `setGravity(int, int, int)` . Esto acepta tres parámetros: una constante de gravedad, un desplazamiento de posición x y un desplazamiento de posición y.

Por ejemplo, si decides que la tostada debería aparecer en la esquina superior izquierda, puedes establecer la gravedad de esta manera:

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

Mostrando un mensaje de brindis

En Android, un Toast es un elemento simple de la interfaz de usuario que se puede usar para dar retroalimentación contextual a un usuario.

Para mostrar un mensaje simple de Toast, podemos hacer lo siguiente.

```
// Declare the parameters to use for the Toast

Context context = getApplicationContext();
// in an Activity, you may also use "this"
// in a fragment, you can use getActivity()

CharSequence message = "I'm an Android Toast!";
int duration = Toast.LENGTH_LONG; // Toast.LENGTH_SHORT is the other option

// Create the Toast object, and show it!
Toast myToast = Toast.makeText(context, message, duration);
myToast.show();
```

O, para mostrar un Toast en línea, sin aferrarte al objeto Toast puedes:

```
Toast.makeText(context, "Ding! Your Toast is ready.", Toast.LENGTH_SHORT).show();
```

IMPORTANTE: asegúrese de que se llame al método `show()` desde el subproceso de la interfaz de usuario. Si está intentando mostrar un `Toast` desde un hilo diferente, por ejemplo, puede usar el método `runOnUiThread` de una `Activity` .

Si no lo hace, es decir, intentar modificar la interfaz de usuario mediante la creación de un Toast, se emitirá una `RuntimeException` que se verá así:

```
java.lang.RuntimeException: Can't create handler inside thread that has not called
Looper.prepare()
```

La forma más sencilla de manejar esta excepción es simplemente usando runOnUiThread: la sintaxis se muestra a continuación.

```
   runOnUiThread(new Runnable() {
        @Override
        public void run() {
            // Your code here
        }
    });
```

Creando un brindis personalizado

Si no desea utilizar la vista predeterminada de Toast, puede proporcionar la suya propia utilizando el `setView(View)` en un objeto `Toast` .

Primero, cree el diseño XML que le gustaría usar en su Toast.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/toast_layout_root"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="8dp"
    android:background="#111">

    <TextView android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#FFF"/>

    <TextView android:id="@+id/description"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#FFF"/>

</LinearLayout>
```

Luego, cuando cree su Toast, infle su Vista personalizada desde XML, y llame a `setView`

```
// Inflate the custom view from XML
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.custom_toast_layout,
                               (ViewGroup) findViewById(R.id.toast_layout_root));

// Set the title and description TextViews from our custom layout
TextView title = (TextView) layout.findViewById(R.id.title);
title.setText("Toast Title");
```

```
TextView description = (TextView) layout.findViewById(R.id.description);
description.setText("Toast Description");

// Create and show the Toast object

Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
toast.setView(layout);
toast.show();
```

Forma segura de subprocesos de mostrar Toast (aplicación amplia)

```
public class MainApplication extends Application {

    private static Context context; //application context

    private Handler mainThreadHandler;
    private Toast toast;

    public Handler getMainThreadHandler() {
        if (mainThreadHandler == null) {
            mainThreadHandler = new Handler(Looper.getMainLooper());
        }
        return mainThreadHandler;
    }

    @Override public void onCreate() {
        super.onCreate();
        context = this;
    }

    public static MainApplication getApp(){
        return (MainApplication) context;
    }

    /**
     * Thread safe way of displaying toast.
     * @param message
     * @param duration
     */
    public void showToast(final String message, final int duration) {
        getMainThreadHandler().post(new Runnable() {
            @Override
            public void run() {
                if (!TextUtils.isEmpty(message)) {
                    if (toast != null) {
                        toast.cancel(); //dismiss current toast if visible
                        toast.setText(message);
                    } else {
                        toast = Toast.makeText(App.this, message, duration);
                    }
                    toast.show();
                }
            }
        });
    }
```

Recuerde agregar `MainApplication` en `manifest` .

Ahora llámalo desde cualquier hilo para mostrar un mensaje de brindis.

```
MainApplication.getApp().showToast("Some message", Toast.LENGTH_LONG);
```

Mostrar mensaje de tostada sobre el teclado suave

De forma predeterminada, Android mostrará los mensajes de Toast en la parte inferior de la pantalla, incluso si se muestra el teclado. Esto mostrará un mensaje de Toast justo encima del teclado.

```
public void showMessage(final String message, final int length) {
    View root = findViewById(android.R.id.content);
    Toast toast = Toast.makeText(this, message, length);
    int yOffset = Math.max(0, root.getHeight() - toast.getYOffset());
    toast.setGravity(Gravity.TOP | Gravity.CENTER_HORIZONTAL, 0, yOffset);
    toast.show();
}
```

Hilo seguro de mostrar un mensaje de Toast (para AsyncTask)

Si no desea extender la Aplicación y mantener seguros los mensajes de su tostada, asegúrese de mostrarlos en la sección de ejecución posterior de sus Tareas Asíncronas.

```
public class MyAsyncTask extends AsyncTask <Void, Void, Void> {

    @Override
    protected Void doInBackground(Void... params) {
        // Do your background work here
    }

    @Override
    protected void onPostExecute(Void aVoid) {
        // Show toast messages here
        Toast.makeText(context, "Ding! Your Toast is ready.",   Toast.LENGTH_SHORT).show();
    }

}
```

Lea tostada en línea: https://riptutorial.com/es/android/topic/1741/tostada

Capítulo 245: Transiciones de elementos compartidos

Introducción

Aquí encontrará ejemplos para la transición entre `Activities` o `Fragments` usando un elemento compartido. Un ejemplo de este comportamiento es la aplicación Google Play Store que traduce el ícono de una aplicación de la lista a la vista de detalles de la aplicación.

Sintaxis

- transaction.addSharedElement (sharedElementView, "targetTransitionName");
- fragment.setSharedElementEnterTransition (new CustomTransaction ());

Examples

Transición de elementos compartidos entre dos fragmentos

En este ejemplo, uno de los dos `ImageViews` diferentes debe traducirse del `ChooserFragment` al `DetailFragment` .

En el diseño de `ChooserFragment` necesitamos los atributos únicos de nombre de `transitionName` :

```
<ImageView
    android:id="@+id/image_first"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_first"
    android:transitionName="fistImage" />

<ImageView
    android:id="@+id/image_second"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_second"
    android:transitionName="secondImage" />
```

En la clase `ChooserFragments` , debemos pasar la `View` que se hizo clic y una ID a la `Activity` principal que está manejando el reemplazo de los fragmentos (necesitamos la ID para saber qué recurso de imagen se debe mostrar en `DetailFragment`). La forma de pasar la información a una actividad de los padres en detalle seguramente está cubierta en otra documentación.

```
view.findViewById(R.id.image_first).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (mCallback != null) {
            mCallback.showDetailFragment(view, 1);
```

```
        }
    }
});

view.findViewById(R.id.image_second).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (mCallback != null) {
            mCallback.showDetailFragment(view, 2);
        }
     }
});
```

En `DetailFragment` , el `ImageView` del elemento compartido también necesita el atributo de `transitionName` `DetailFragment` único.

```
<ImageView
    android:id="@+id/image_shared"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:transitionName="sharedImage" />
```

En el método `onCreateView()` de `DetailFragment` , tenemos que decidir qué recurso de imagen se debe mostrar (si no lo hacemos, el elemento compartido desaparecerá después de la transición).

```
public static DetailFragment newInstance(Bundle args) {
    DetailFragment fragment = new DetailFragment();
    fragment.setArguments(args);
    return fragment;
}

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    View view = inflater.inflate(R.layout.fragment_detail, container, false);

    ImageView sharedImage = (ImageView) view.findViewById(R.id.image_shared);

    // Check which resource should be shown.
    int type = getArguments().getInt("type");

    // Show image based on the type.
    switch (type) {
        case 1:
            sharedImage.setBackgroundResource(R.drawable.ic_first);
            break;

        case 2:
            sharedImage.setBackgroundResource(R.drawable.ic_second);
            break;
    }

    return view;
}
```

La `Activity` principal está recibiendo las devoluciones de llamada y se ocupa de la sustitución de los fragmentos.

```
@Override
public void showDetailFragment(View sharedElement, int type) {
    // Get the chooser fragment, which is shown in the moment.
    Fragment chooserFragment = getFragmentManager().findFragmentById(R.id.fragment_container);

    // Set up the DetailFragment and put the type as argument.
    Bundle args = new Bundle();
    args.putInt("type", type);
    Fragment fragment = DetailFragment.newInstance(args);

    // Set up the transaction.
    FragmentTransaction transaction = getFragmentManager().beginTransaction();

    // Define the shared element transition.
    fragment.setSharedElementEnterTransition(new DetailsTransition());
    fragment.setSharedElementReturnTransition(new DetailsTransition());

    // The rest of the views are just fading in/out.
    fragment.setEnterTransition(new Fade());
    chooserFragment.setExitTransition(new Fade());

    // Now use the image's view and the target transitionName to define the shared element.
    transaction.addSharedElement(sharedElement, "sharedImage");

    // Replace the fragment.
    transaction.replace(R.id.fragment_container, fragment,
fragment.getClass().getSimpleName());

    // Enable back navigation with shared element transitions.
    transaction.addToBackStack(fragment.getClass().getSimpleName());

    // Finally press play.
    transaction.commit();
}
```

No hay que olvidar - la `Transition` misma. Este ejemplo mueve y escala el elemento compartido.

```
@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class DetailsTransition extends TransitionSet {

    public DetailsTransition() {
        setOrdering(ORDERING_TOGETHER);
        addTransition(new ChangeBounds()).
            addTransition(new ChangeTransform()).
            addTransition(new ChangeImageTransform());
    }

}
```

Lea Transiciones de elementos compartidos en línea:
https://riptutorial.com/es/android/topic/8933/transiciones-de-elementos-compartidos

Capítulo 246: TransitionDrawable

Examples

Añadir transición o cross-fade entre dos imágenes.

Paso 1: Crea una transición dibujable en XML

Guarde este archivo `transition.xml` en la carpeta `res/drawable` de su proyecto.

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/image1"/>
    <item android:drawable="@drawable/image2"/>
</transition>
```

La imagen1 y la imagen2 son las dos imágenes que queremos hacer la transición y también deben colocarse en su carpeta `res/drawable` .

Paso 2: Agregue el código para ImageView en su diseño XML para mostrar el dibujo anterior.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <ImageView
        android:id="@+id/image_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/image1"/>

</LinearLayout>
```

Paso 3: Acceda a la transición XML dibujable en el método onCreate () de su Actividad e inicie la transición en el evento onClick ().

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    imageView = (ImageView) findViewById(R.id.image_view);
```

```
    transitionDrawable = (TransitionDrawable)
        ContextCompat.getDrawable(this, R.drawable.transition);

    birdImageView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(final View view) {
            birdImageView.setImageDrawable(transitionDrawable);
            transitionDrawable.startTransition(1000);
        }
    });
}
```

Animar vistas de color de fondo (cambio de color) con TransitionDrawable

```
public void setCardColorTran(View view) {
   ColorDrawable[] color = {new ColorDrawable(Color.BLUE), new ColorDrawable(Color.RED)};
   TransitionDrawable trans = new TransitionDrawable(color);
    if(Build.VERSION.SDK_INT < android.os.Build.VERSION_CODES.JELLY_BEAN) {
        view.setBackgroundDrawable(trans);
        }else {
        view.setBackground(trans);
    }
    trans.startTransition(5000);
}
```

Lea TransitionDrawable en línea: https://riptutorial.com/es/android/topic/6088/transitiondrawable

Capítulo 247: Ubicación

Introducción

Las API de ubicación de Android se utilizan en una amplia variedad de aplicaciones para diferentes propósitos, como encontrar la ubicación del usuario, notificar cuando un usuario ha abandonado un área general (Geofencing) y ayudar a interpretar la actividad del usuario (caminar, correr, conducir, etc.).

Sin embargo, las API de ubicación de Android no son el único medio de adquirir la ubicación del usuario. Lo siguiente le dará ejemplos de cómo usar el `LocationManager` de Android y otras bibliotecas de ubicación comunes.

Observaciones

Para crear aplicaciones conscientes de la ubicación en Android, hay dos caminos:

- `LocationManager` nativo de código abierto de Android
- `FusedLocationProviderApi` de Google, que forma parte de los servicios de Google Play

Gerente de locación

Pros

- Control más granular.
- Disponible en todos los dispositivos.
- Parte del framework Android

Contras

- El consumo de la batería es un problema, si no se administra correctamente
- Requiere lógica para cambiar los proveedores de ubicación, si el dispositivo no puede encontrar una ubicación (por ejemplo, GPS deficiente dentro de un edificio)

Caracteristicas

- Oyente NMEA
- Escucha del estado del GPS
- Escuche los cambios de estado del proveedor (por ejemplo, el GPS está apagado por el usuario)
- Lista de proveedores para elegir la fuente de ubicación de

Proveedores

GPS

- Permisos requeridos:
  - `ACCESS_FINE_LOCATION`
- Precisión: 10m - 100m
- Requisitos de potencia: ALTA
- Disponibilidad: Mundial (con clara vista del cielo).
- **NOTAS** :
  - Las actualizaciones de ubicación generalmente se realizan una vez por segundo, pero en situaciones donde el GPS no se ha utilizado durante un tiempo y el A-GPS no está disponible, puede tomar varios minutos para que se reciba una ubicación.
  - En los casos en que se obstruye la vista clara del cielo, los puntos GPS no se agruparán muy bien (los puntos de ubicación "saltan") y la precisión puede ser engañosa en ciertas áreas debido al efecto del " Cañón urbano ".

Red

- Permisos requeridos:
  - `ACCESS_COARSE_LOCATION` o `ACCESS_FINE_LOCATION`
- Precisión: 100m - 1000m +
- Requisitos de alimentación: BAJO - MEDIO
- Disponibilidad: Dentro del rango de torre celular o señal wifi.
- **NOTAS:**
  - Las actualizaciones de ubicación ocurren con menos frecuencia que el GPS
  - Las actualizaciones de ubicación generalmente no se agrupan bien (los puntos de ubicación "saltan") y la precisión puede variar según el número de factores diferentes (número de señales wifi, intensidad de la señal, tipo de torre celular, etc.)

Pasivo

- Permisos requeridos:
  - `ACCESS_FINE_LOCATION`
- Precisión: 10m - 1000m +
- Requisitos de alimentación: NINGUNO
- Disponibilidad: solo cuando otra aplicación recibe una ubicación de GPS o red
- **NOTAS:**
  - No confíe en esto para darle actualizaciones continuas. Esto escucha de forma pasiva a otras aplicaciones que realizan solicitudes de ubicación y devuelve esas ubicaciones.
  - No devuelve los puntos generados por FusedLocationProviderApi, solo los puntos de ubicación subyacentes utilizados para generarlos.

FusedLocationProviderApi

Pros

- Ofrece menos consumo de batería "fuera de la caja"
- Maneja bien el mal GPS

- Recibe actualizaciones más regularmente

Contras

- Menos control granular sobre GPS
- Puede no estar disponible en todos los dispositivos o en ciertos países
- Requiere dependencia de biblioteca de terceros

Caracteristicas

- Uso bien administrado de los proveedores de ubicación para un ahorro óptimo de la masa
- Normalmente genera puntos más precisos que el proveedor de ubicación de red
- Actualizaciones más frecuentes de la biblioteca, permitiendo más mejoras.
- No es necesario especificar qué tipo de proveedor utilizar

UbicaciónSolicitud de Niveles de Prioridad

PRIORITY_HIGH_ACCURACY

- Permisos requeridos:
  - `ACCESS_FINE_LOCATION` para una ubicación más precisa o `ACCESS_COARSE_LOCATION` para una ubicación menos precisa
- Precisión: 10m - 100m
- Requisitos de potencia: ALTA
- Disponibilidad: Dondequiera que esté disponible Google Play Services.
- **NOTAS:**
  - Si no se usa `ACCESS_FINE_LOCATION` , esto no usará el GPS para generar actualizaciones de ubicación, pero seguirá encontrando un punto bastante preciso en las condiciones correctas.
  - Si se usa `ACCESS_FINE_LOCATION` , puede o no usar GPS para generar puntos de ubicación, dependiendo de la precisión con la que actualmente puede rastrear el dispositivo dadas las condiciones ambientales.
  - Aunque esto puede reportar actualizaciones de ubicación más precisas que las otras configuraciones, todavía es propenso al efecto " Urban Canyon ".

PRIORITY_BALANCED_POWER_ACCURACY

- Permisos requeridos:
  - `ACCESS_FINE_LOCATION` para una ubicación más precisa o `ACCESS_COARSE_LOCATION` para una ubicación menos precisa
- Precisión: 100m - 1000m +
- Requisitos de energía: MEDIO
- Disponibilidad: Dondequiera que esté disponible Google Play Services.
- **NOTAS:**
  - Las mismas notas que `PRIORITY_HIGH_ACCURACY`
  - Aunque es poco probable, esta configuración puede seguir utilizando el GPS para generar una ubicación.

PRIORITY_LOW_POWER

- Permisos requeridos:
  - `ACCESS_FINE_LOCATION` o `ACCESS_COARSE_LOCATION`
- Precisión: 100m - 1000m +
- Requisitos de alimentación: BAJA
- Disponibilidad: Dondequiera que esté disponible Google Play Services.
- **NOTAS:**
  - Probablemente no use GPS, pero no está probado hasta ahora.
  - Las actualizaciones no suelen ser muy precisas
  - Usado generalmente para detectar cambios significativos en la ubicación.

PRIORITY_NO_POWER

- Permisos requeridos:
  - `ACCESS_FINE_LOCATION` o `ACCESS_COARSE_LOCATION`
- Precisión: 10m - 1000m +
- Requisitos de alimentación: NINGUNO
- Disponibilidad: Dondequiera que esté disponible Google Play Services.
- **NOTAS:**
  - Funciona de forma casi idéntica al `LocationManager PASSIVE_PROVIDER`
  - Informa sobre las actualizaciones de Google Play Services cuando se recibe, donde `PASSIVE_PROVIDER` informa sobre las actualizaciones de ubicación subyacentes utilizadas

Solución de problemas

OnLocationChanged () nunca llamado

Ya que este parece ser un problema común con la obtención de ubicaciones de Android, pondré una lista rápida de correcciones comunes:

---

1. Revisa tu manifiesto!

   Uno de los problemas más comunes es que nunca se dieron los permisos correctos. Si está utilizando GPS (con o sin red), use `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>` , o use `<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>` . FusedLocationApi de Google requiere `ACCESS_FINE_LOCATION` .

---

2. (Para Android 6+) Verifique los permisos de tiempo de ejecución !

   Compruebe y solicite permisos! Si nunca te dan permisos, terminarás con colisiones o, lo que es peor (si estás detectando todas las excepciones), ¡terminarás sin ninguna indicación de nada! No importa si el usuario le otorga permiso al inicio de la aplicación, siempre verifique si tiene permisos para todas las llamadas. El usuario puede acceder fácilmente a sus ajustes y revocarlos.

---

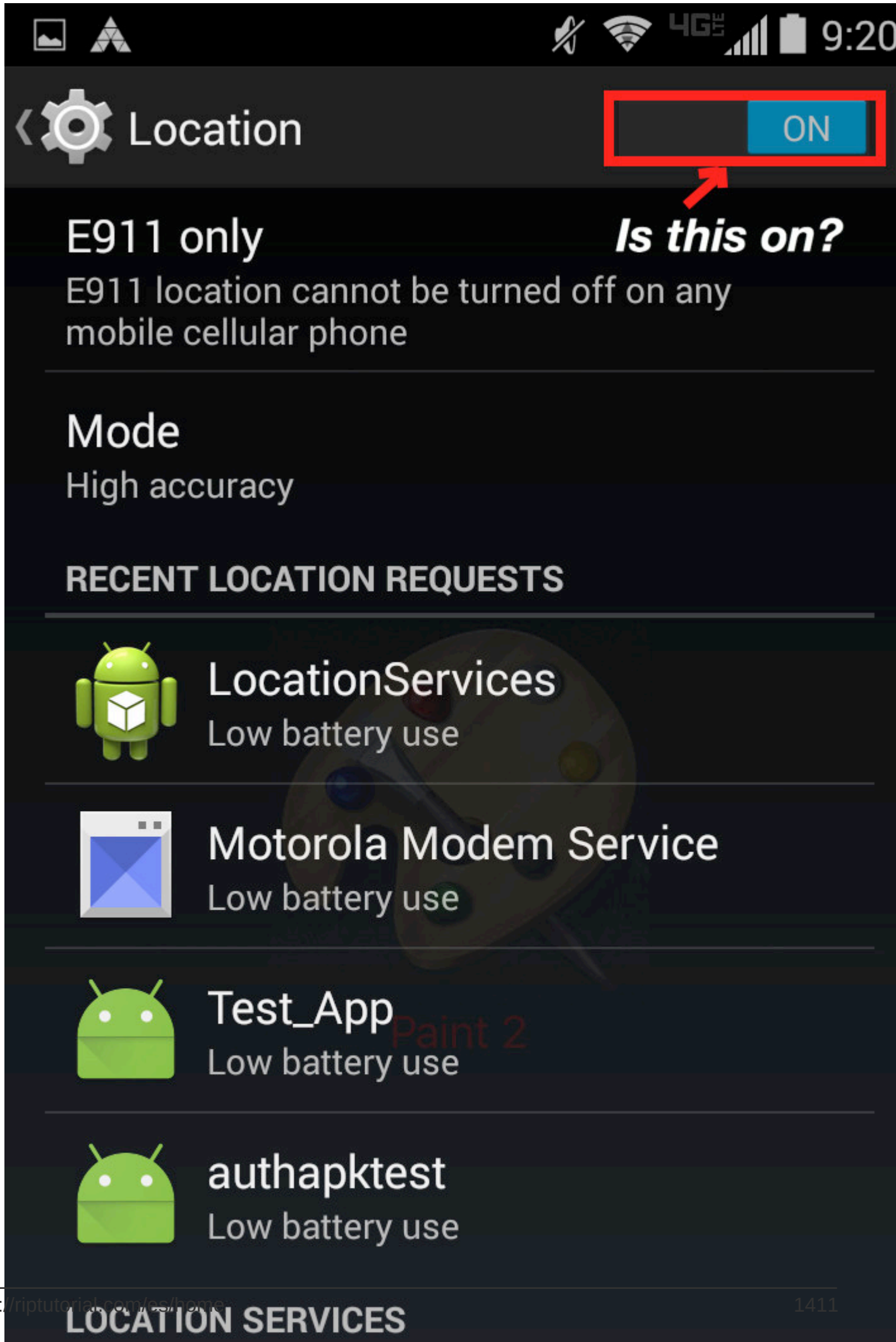
3. ¡Comprueba tu código!

   ¿Estás seguro de que estás pasando en el oyente correcto? ¿ `IntentService` ese `BroadcastReceiver` o `IntentService` a su manifiesto? ¿Está utilizando `PendingIntent.getService()` en una clase `BroadcastReceiver` , o `getBroadcast()` en una clase `IntentService` ? ¿Está seguro de que no está desregistrando a su oyente en otra parte de su código inmediatamente después de realizar la solicitud?

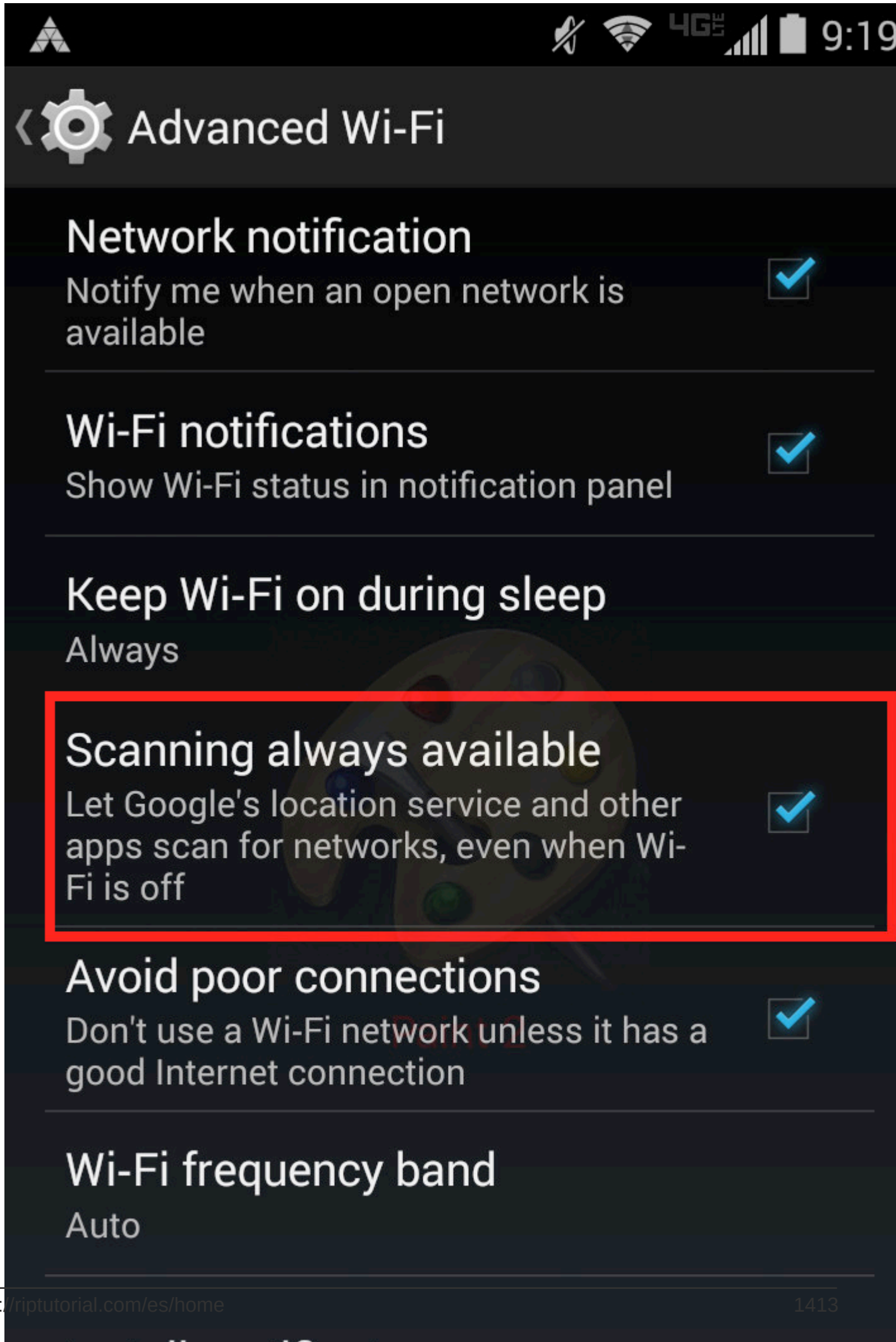
---

4. Compruebe la configuración del dispositivo!

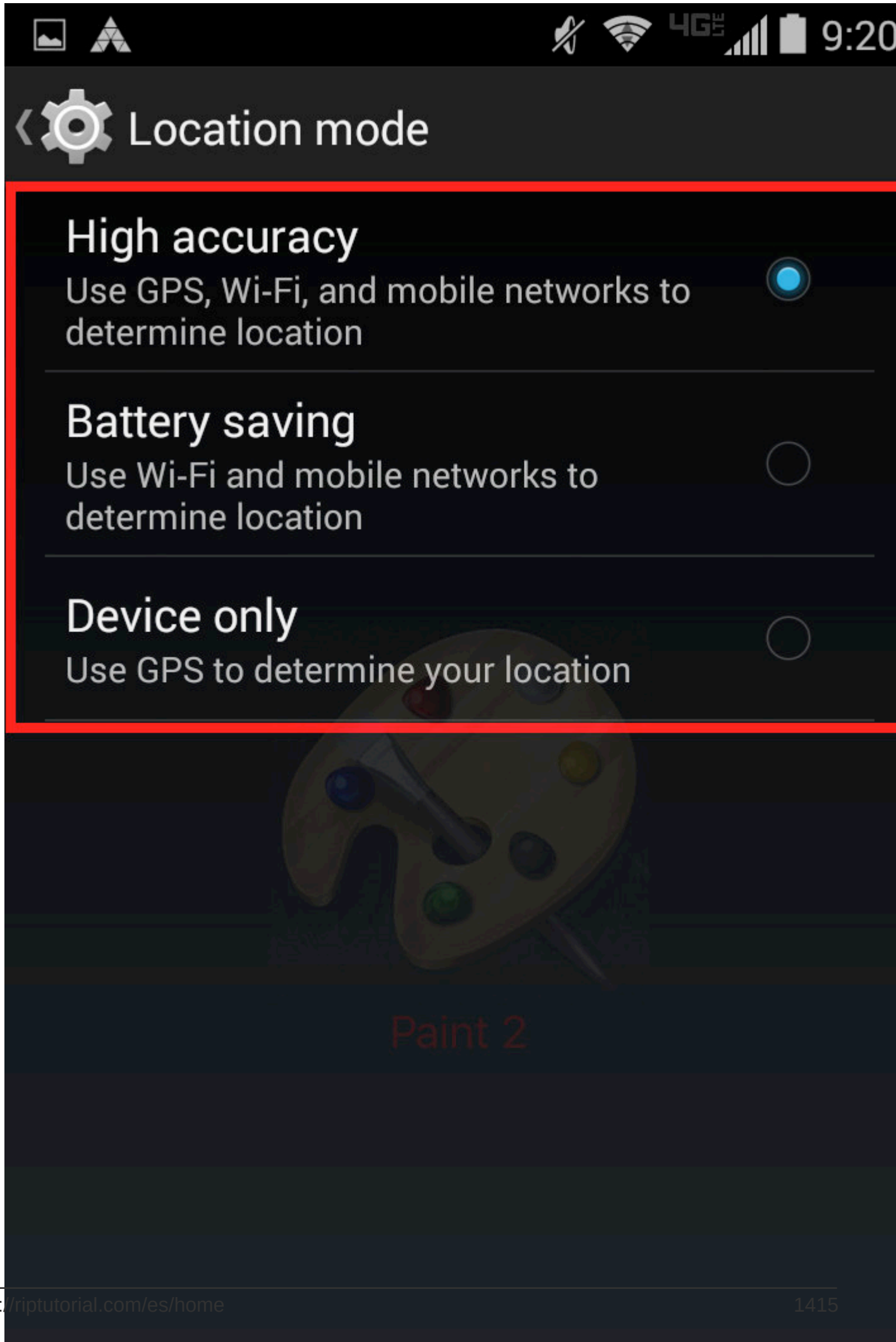
   Obviamente, asegúrate de tener los servicios de ubicación activados.


9:20
Location
ON
Is this on?
E911 only
E911 location cannot be turned off on any mobile cellular phone
Mode
High accuracy
RECENT LOCATION REQUESTS
LocationServices
Low battery use
Motorola Modem Service
Low battery use
Test_App
Low battery use
authapktest
Low battery use
LOCATION SERVICES

Si está utilizando servicios de red, ¿activó "Escanear siempre disponible"? ¿Tiene el modo de ubicación configurado en "Mejor" ("Alta precisión") o "Ahorro de batería" ("Sólo en red")?


9:19
Advanced Wi-Fi
Network notification
Notify me when an open network is available
Wi-Fi notifications
Show Wi-Fi status in notification panel
Keep Wi-Fi on during sleep
Always
Scanning always available
Let Google's location service and other apps scan for networks, even when Wi-Fi is off
Avoid poor connections
Don't use a Wi-Fi network unless it has a good Internet connection
Wi-Fi frequency band
Auto

Si está utilizando GPS, ¿activó "Mejor" ("Alta precisión") o "Solo dispositivo" en el modo de ubicación?


9:20
Location mode
High accuracy
Use GPS, Wi-Fi, and mobile networks to determine location
Battery saving
Use Wi-Fi and mobile networks to determine location
Device only
Use GPS to determine your location
Paint 2

Sí, esto está aquí dos veces. ¿ `PendingIntent` usar un `LocationListener` lugar de un `PendingIntent` , o viceversa, para asegurarse de que realmente implementó el `LocationManager` correctamente? ¿Está seguro de que la solicitud de ubicación no se está eliminando en alguna parte del ciclo de vida de la actividad o el servicio que no esperaba que ocurriera?

`PendingIntent` , o viceversa, para asegurarse de que realmente implementó el `LocationManager` correctamente? ¿Está seguro de que la solicitud de ubicación no se está eliminando en alguna parte del ciclo de vida de la actividad o el servicio que no esperaba que ocurriera?

---

6. Revisa tu entorno!

   ¿Está probando el GPS en el primer piso de un edificio en el centro de San Francisco? ¿Está probando ubicaciones de red en medio de la nada? ¿Trabaja en un búnker subterráneo secreto sin todas las señales de radio, preguntándose por qué su dispositivo no tiene ubicación? ¡Siempre revise dos veces sus alrededores cuando intente solucionar problemas de ubicación!

Podría haber muchas otras razones menos obvias por las que la ubicación no funciona, pero antes de buscar esas correcciones esotéricas, simplemente ejecute esta lista de verificación rápida.

Examples

API de ubicación fusionada

Ejemplo de uso de la actividad con LocationRequest

```
/*
 * This example is useful if you only want to receive updates in this
 * activity only, and have no use for location anywhere else.
 */
public class LocationActivity extends AppCompatActivity implements
        GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
LocationListener {

    private GoogleApiClient mGoogleApiClient;
    private LocationRequest mLocationRequest;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mGoogleApiClient = new GoogleApiClient.Builder(this)
```

```
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();

        mLocationRequest = new LocationRequest()
                .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY) //GPS quality location
points
                .setInterval(2000) //At least once every 2 seconds
                .setFastestInterval(1000); //At most once a second
    }

    @Override
    protected void onStart(){
        super.onStart();
        mGoogleApiClient.connect();
    }

    @Override
    protected void onResume(){
        super.onResume();
        //Permission check for Android 6.0+
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
            if(mGoogleApiClient.isConnected()) {
                LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
            }
        }
    }

    @Override
    protected void onPause(){
        super.onPause();
        //Permission check for Android 6.0+
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
            if(mGoogleApiClient.isConnected()) {
                LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient,
this);
            }
        }
    }

    @Override
    protected void onStop(){
        super.onStop();
        mGoogleApiClient.disconnect();
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
        }
    }

    @Override
    public void onConnectionSuspended(int i) {
```

```
        mGoogleApiClient.connect();
    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    }

    @Override
    public void onLocationChanged(Location location) {
        //Handle your location update code here
    }
}
```

Ejemplo de uso de Service w / PendingIntent y BroadcastReceiver

EjemploActividad

Lectura recomendada: LocalBroadcastManager

```
/*
 * This example is useful if you have many different classes that should be
 * receiving location updates, but want more granular control over which ones
 * listen to the updates.
 *
 * For example, this activity will stop getting updates when it is not visible, but a database
 * class with a registered local receiver will continue to receive updates, until
"stopUpdates()" is called here.
 *
 */
public class ExampleActivity extends AppCompatActivity {

    private InternalLocationReceiver mInternalLocationReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);

        //Create internal receiver object in this method only.
        mInternalLocationReceiver = new InternalLocationReceiver(this);
    }

    @Override
    protected void onResume(){
        super.onResume();

        //Register to receive updates in activity only when activity is visible
        LocalBroadcastManager.getInstance(this).registerReceiver(mInternalLocationReceiver,
new IntentFilter("googleLocation"));
    }

    @Override
    protected void onPause(){
        super.onPause();

        //Unregister to stop receiving updates in activity when it is not visible.
```

```
        //NOTE: You will still receive updates even if this activity is killed.
        LocalBroadcastManager.getInstance(this).unregisterReceiver(mInternalLocationReceiver);
    }

    //Helper method to get updates
    private void requestUpdates(){
        startService(new Intent(this, LocationService.class).putExtra("request", true));
    }

    //Helper method to stop updates
    private void stopUpdates(){
        startService(new Intent(this, LocationService.class).putExtra("remove", true));
    }

    /*
     * Internal receiver used to get location updates for this activity.
     *
     * This receiver and any receiver registered with LocalBroadcastManager does
     * not need to be registered in the Manifest.
     *
     */
    private static class InternalLocationReceiver extends BroadcastReceiver{

        private ExampleActivity mActivity;

        InternalLocationReceiver(ExampleActivity activity){
            mActivity = activity;
        }

        @Override
        public void onReceive(Context context, Intent intent) {
            final ExampleActivity activity = mActivity;
            if(activity != null) {
                LocationResult result = intent.getParcelableExtra("result");
                //Handle location update here
            }
        }
    }
}
```

Servicio de localización

NOTA: ¡No olvides registrar este servicio en el Manifiesto!

```
public class LocationService extends Service implements
        GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener {

    private GoogleApiClient mGoogleApiClient;
    private LocationRequest mLocationRequest;

    @Override
    public void onCreate(){
        super.onCreate();
        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();

        mLocationRequest = new LocationRequest()
```

```
                .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY) //GPS quality location
points
                .setInterval(2000) //At least once every 2 seconds
                .setFastestInterval(1000); //At most once a second
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        super.onStartCommand(intent, flags, startId);
        //Permission check for Android 6.0+
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
            if (intent.getBooleanExtra("request", false)) {
                if (mGoogleApiClient.isConnected()) {
                    LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, getPendingIntent());
                } else {
                    mGoogleApiClient.connect();
                }
            }
            else if(intent.getBooleanExtra("remove", false)){
                stopSelf();
            }
        }

        return START_STICKY;
    }

    @Override
    public void onDestroy(){
        super.onDestroy();
        if(mGoogleApiClient.isConnected()){
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient,
getPendingIntent());
            mGoogleApiClient.disconnect();
        }
    }

    private PendingIntent getPendingIntent(){

        //Example for IntentService
        //return PendingIntent.getService(this, 0, new Intent(this,
**YOUR_INTENT_SERVICE_CLASS_HERE**), PendingIntent.FLAG_UPDATE_CURRENT);

        //Example for BroadcastReceiver
        return PendingIntent.getBroadcast(this, 0, new Intent(this, LocationReceiver.class),
PendingIntent.FLAG_UPDATE_CURRENT);
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        //Permission check for Android 6.0+
        if(ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, getPendingIntent());
        }
    }

    @Override
    public void onConnectionSuspended(int i) {
```

```
        mGoogleApiClient.connect();
    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

UbicaciónReceptor

NOTA: ¡No olvides registrar este receptor en el Manifiesto!

```
public class LocationReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        if(LocationResult.hasResult(intent)){
            LocationResult locationResult = LocationResult.extractResult(intent);
            LocalBroadcastManager.getInstance(context).sendBroadcast(new
Intent("googleLocation").putExtra("result", locationResult));
        }
    }
}
```

Solicitando actualizaciones de ubicación usando LocationManager

Como siempre, debe asegurarse de tener los permisos necesarios.

```
public class MainActivity extends AppCompatActivity implements LocationListener{

    private LocationManager mLocationManager = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        mLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    }


    @Override
    protected void onResume() {
        super.onResume();

        try {
            mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);
        }
        catch(SecurityException e){
            // The app doesn't have the correct permissions
```

```
        }
    }


    @Override
    protected void onPause() {
        try{
            mLocationManager.removeUpdates(this);
        }
        catch (SecurityException e){
            // The app doesn't have the correct permissions
        }

        super.onPause();
    }




    @Override
    public void onLocationChanged(Location location) {
        // We received a location update!
        Log.i("onLocationChanged", location.toString());
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {

    }

    @Override
    public void onProviderEnabled(String provider) {

    }

    @Override
    public void onProviderDisabled(String provider) {

    }
}
```

Solicitar actualizaciones de ubicación en un subproceso separado utilizando LocationManager

Como siempre, debe asegurarse de tener los permisos necesarios.

```
public class MainActivity extends AppCompatActivity implements LocationListener{

    private LocationManager mLocationManager = null;
    HandlerThread mLocationHandlerThread = null;
    Looper mLocationHandlerLooper = null;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
```

```
        mLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        mLocationHandlerThread = new HandlerThread("locationHandlerThread");
    }


    @Override
    protected void onResume() {
        super.onResume();

        mLocationHandlerThread.start();
        mLocationHandlerLooper = mLocationHandlerThread.getLooper();

        try {
            mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this,
mLocationHandlerLooper);
        }
        catch(SecurityException e){
            // The app doesn't have the correct permissions
        }
    }


    @Override
    protected void onPause() {
        try{
            mLocationManager.removeUpdates(this);
        }
        catch (SecurityException e){
            // The app doesn't have the correct permissions
        }

        mLocationHandlerLooper = null;

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR2)
            mLocationHandlerThread.quitSafely();
        else
            mLocationHandlerThread.quit();

        mLocationHandlerThread = null;


        super.onPause();
    }




    @Override
    public void onLocationChanged(Location location) {
        // We received a location update on a separate thread!
        Log.i("onLocationChanged", location.toString());

        // You can verify which thread you're on by something like this:
        // Log.d("Which thread?", Thread.currentThread() == Looper.getMainLooper().getThread()
? "UI Thread" : "New thread");
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {

    }
```

```
    @Override
    public void onProviderEnabled(String provider) {

    }

    @Override
    public void onProviderDisabled(String provider) {

    }
}
```

Registrar geofence

He creado la clase de `GeoFenceObserversationService` **singleton** .

GeoFenceObserversationService.java :

```
public class GeoFenceObserversationService extends Service implements
GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
ResultCallback<Status> {

    protected static final String TAG = "GeoFenceObserversationService";
    protected GoogleApiClient mGoogleApiClient;
    protected ArrayList<Geofence> mGeofenceList;
    private boolean mGeofencesAdded;
    private SharedPreferences mSharedPreferences;
    private static GeoFenceObserversationService mInstant;
    public static GeoFenceObserversationService getInstant(){
        return mInstant;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        mInstant = this;
        mGeofenceList = new ArrayList<Geofence>();
        mSharedPreferences = getSharedPreferences(AppConstants.SHARED_PREFERENCES_NAME,
MODE_PRIVATE);
        mGeofencesAdded = mSharedPreferences.getBoolean(AppConstants.GEOFENCES_ADDED_KEY,
false);

        buildGoogleApiClient();
    }


    @Override
    public void onDestroy() {
        mGoogleApiClient.disconnect();
        super.onDestroy();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
```

```
    public int onStartCommand(Intent intent, int flags, int startId) {
        return START_STICKY;
    }

    protected void buildGoogleApiClient() {
        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
        mGoogleApiClient.connect();
    }

    @Override
    public void onConnected(Bundle connectionHint) {
    }

    @Override
    public void onConnectionFailed(ConnectionResult result) {
    }

    @Override
    public void onConnectionSuspended(int cause) {

    }

    private GeofencingRequest getGeofencingRequest() {

        GeofencingRequest.Builder builder = new GeofencingRequest.Builder();
        builder.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER);
        builder.addGeofences(mGeofenceList);
        return builder.build();
    }


    public void addGeofences() {
            if (!mGoogleApiClient.isConnected()) {
                Toast.makeText(this, getString(R.string.not_connected),
Toast.LENGTH_SHORT).show();
                return;
            }

        populateGeofenceList();
        if(!mGeofenceList.isEmpty()){
            try {
                LocationServices.GeofencingApi.addGeofences(mGoogleApiClient,
getGeofencingRequest(), getGeofencePendingIntent()).setResultCallback(this);
            } catch (SecurityException securityException) {
                securityException.printStackTrace();
            }
        }

        }

    public void removeGeofences() {
        if (!mGoogleApiClient.isConnected()) {
            Toast.makeText(this, getString(R.string.not_connected),
Toast.LENGTH_SHORT).show();
            return;
        }
        try {
```

```
            LocationServices.GeofencingApi.removeGeofences(mGoogleApiClient,getGeofencePending
Intent()).setResultCallback(this);
        } catch (SecurityException securityException) {
            securityException.printStackTrace();
        }
    }


    public void onResult(Status status) {

        if (status.isSuccess()) {
            mGeofencesAdded = !mGeofencesAdded;
            SharedPreferences.Editor editor = mSharedPreferences.edit();
            editor.putBoolean(AppConstants.GEOFENCES_ADDED_KEY, mGeofencesAdded);
            editor.apply();
        } else {
            String errorMessage = AppConstants.getErrorString(this,status.getStatusCode());
            Log.i("Geofence", errorMessage);
        }
    }

    private PendingIntent getGeofencePendingIntent() {
        Intent intent = new Intent(this, GeofenceTransitionsIntentService.class);
        return PendingIntent.getService(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
    }

    private void populateGeofenceList() {
        mGeofenceList.clear();
        List<GeoFencingResponce> geoFenceList = getGeofencesList;
        if(geoFenceList!=null&&!geoFenceList.isEmpty()){
            for (GeoFencingResponce obj : geoFenceList){
                mGeofenceList.add(obj.getGeofence());
                Log.i(TAG,"Registered Geofences : " + obj.Id+"-"+obj.Name+"-"+obj.Lattitude+"-
"+obj.Longitude);
            }
        }
    }
}
```

AppConstant :

```
   public static final String SHARED_PREFERENCES_NAME = PACKAGE_NAME +
".SHARED_PREFERENCES_NAME";
    public static final String GEOFENCES_ADDED_KEY = PACKAGE_NAME + ".GEOFENCES_ADDED_KEY";
    public static final String DETECTED_GEOFENCES = "detected_geofences";
    public static final String DETECTED_BEACONS = "detected_beacons";

    public static String getErrorString(Context context, int errorCode) {
        Resources mResources = context.getResources();
        switch (errorCode) {
            case GeofenceStatusCodes.GEOFENCE_NOT_AVAILABLE:
                return mResources.getString(R.string.geofence_not_available);
            case GeofenceStatusCodes.GEOFENCE_TOO_MANY_GEOFENCES:
                return mResources.getString(R.string.geofence_too_many_geofences);
            case GeofenceStatusCodes.GEOFENCE_TOO_MANY_PENDING_INTENTS:
                return mResources.getString(R.string.geofence_too_many_pending_intents);
            default:
                return mResources.getString(R.string.unknown_geofence_error);
        }
    }
```

¿Dónde empecé servicio? De la clase de aplicación

- `startService(new Intent(getApplicationContext(),GeoFenceObserversationService.class));`

¿Cómo me registré Geofences?

- `GeoFenceObserversationService.getInstant().addGeofences();`

Obtener la dirección de la ubicación utilizando Geocoder

Después de obtener el objeto de `Location` de `FusedAPI` , puede adquirir fácilmente la información de `Address` de ese objeto.

```
private Address getCountryInfo(Location location) {
    Address address = null;
    Geocoder geocoder = new Geocoder(getActivity(), Locale.getDefault());
    String errorMessage;
    List<Address> addresses = null;
    try {
        addresses = geocoder.getFromLocation(
                location.getLatitude(),
                location.getLongitude(),
                // In this sample, get just a single address.
                1);
    } catch (IOException ioException) {
        // Catch network or other I/O problems.
        errorMessage = "IOException>>" + ioException.getMessage();
    } catch (IllegalArgumentException illegalArgumentException) {
        // Catch invalid latitude or longitude values.
        errorMessage = "IllegalArgumentException>>" + illegalArgumentException.getMessage();
    }
    if (addresses != null && !addresses.isEmpty()) {
        address = addresses.get(0);
    }
    return country;
}
```

Obteniendo actualizaciones de ubicación en un BroadcastReceiver

Primero cree una clase BroadcastReceiver para manejar las actualizaciones de ubicación entrantes:

```
public class LocationReceiver extends BroadcastReceiver implements Constants {

    @Override
     public void onReceive(Context context, Intent intent) {

        if (LocationResult.hasResult(intent)) {
            LocationResult locationResult = LocationResult.extractResult(intent);
            Location location = locationResult.getLastLocation();
            if (location != null) {
                // Do something with your location
            } else {
                Log.d(LocationReceiver.class.getSimpleName(), "*** location object is null
***");
            }
```

```
        }
    }
}
```

Luego, cuando se conecte a GoogleApiClient en la devolución de llamada onConnected:

```
@Override
public void onConnected(Bundle connectionHint) {
    Intent backgroundIntent = new Intent(this, LocationReceiver.class);
    mBackgroundPendingIntent = backgroundPendingIntent.getBroadcast(getApplicationContext(),
LOCATION_REUEST_CODE, backgroundIntent, PendingIntent.FLAG_CANCEL_CURRENT);
    mFusedLocationProviderApi.requestLocationUpdates(mLocationClient, mLocationRequest,
backgroundPendingIntent);
}
```

No olvide eliminar el intento de actualización de la ubicación en la devolución de llamada del ciclo de vida apropiado:

```
@Override
public void onDestroy() {
    if (servicesAvailable && mLocationClient != null) {
        if (mLocationClient.isConnected()) {
            fusedLocationProviderApi.removeLocationUpdates(mLocationClient,
backgroundPendingIntent);
            // Destroy the current location client
            mLocationClient = null;
        } else {
            mLocationClient.unregisterConnectionCallbacks(this);
            mLocationClient = null;
        }
    }
    super.onDestroy();
}
```

Lea Ubicación en línea: https://riptutorial.com/es/android/topic/1837/ubicacion

Capítulo 248: Una forma rápida de configurar Retrolambda en un proyecto de Android.

Introducción

Retrolambda es una biblioteca que permite utilizar expresiones lambda de Java 8, referencias de métodos y declaraciones de prueba con recursos en Java 7, 6 o 5.

El complemento Retrolambda de Gradle permite integrar Retrolambda en una construcción basada en Gradle. Esto permite, por ejemplo, utilizar estas construcciones en una aplicación de Android, ya que el desarrollo estándar de Android actualmente no es compatible con Java 8.

Examples

Configuración y ejemplo de uso:

Pasos de configuración:

1. Descarga e instala jdk8.
2. Agregue lo siguiente a la construcción principal de su proyecto.

```
buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
        classpath 'me.tatarka:gradle-retrolambda:3.2.3'
    }
}
```

3. Ahora agregue esto al build.gradle de su módulo de aplicación

```
apply plugin: 'com.android.application' // or apply plugin: 'java'
apply plugin: 'me.tatarka.retrolambda'
```

4. Agregue estas líneas al build.gradle de su módulo de aplicación para informar al IDE del nivel de idioma:

```
android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

Ejemplo:

Así que cosas como esta:

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        log("Clicked");
    }
});
```

Conviértete en esto:

```
button.setOnClickListener(v -> log("Clicked"));
```

Lea Una forma rápida de configurar Retrolambda en un proyecto de Android. en línea: https://riptutorial.com/es/android/topic/8822/una-forma-rapida-de-configurar-retrolambda-en-un-proyecto-de-android-

Capítulo 249: URL de devolución de llamada

Examples

Ejemplo de URL de devolución de llamada con Instagram OAuth

Uno de los casos de uso de las *URL* de *devolución* de *llamada* es OAuth. Hagamos esto con un inicio de sesión en Instagram: si el usuario ingresa sus credenciales y hace clic en el botón *Iniciar sesión* , Instagram validará las credenciales y devolverá un `access_token` . Necesitamos ese `access_token` en nuestra aplicación.

Para que nuestra aplicación pueda escuchar dichos enlaces, debemos agregar una URL de devolución de llamada a nuestra `Activity` . Podemos hacer esto agregando un `<intent-filter/>` a nuestra `Activity` , que reaccionará a esa URL de devolución de llamada. Supongamos que nuestra URL de devolución de llamada es `appSchema://appName.com` . Luego, debe agregar las siguientes líneas a su `Activity` deseada en el archivo *Manifest.xml* :

```
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.BROWSABLE"/>
<data android:host="appName.com" android:scheme="appSchema"/>
```

Explicación de las líneas anteriores:

- `<category android:name="android.intent.category.BROWSABLE"/>` hace que la actividad de destino se inicie mediante un navegador web para mostrar los datos a los que se hace referencia en un enlace.
- `<data android:host="appName.com" android:scheme="appSchema"/>` especifica nuestro esquema y host de nuestra URL de devolución de llamada.
- En conjunto, estas líneas harán que la `Activity` específica se abra cada vez que se llame a la URL de devolución de llamada en un navegador.

Ahora, para obtener el contenido de la URL en su `Activity` , debe anular el método `onResume()` siguiente manera:

```
@Override
public void onResume() {
    // The following line will return "appSchema://appName.com".
    String CALLBACK_URL = getResources().getString(R.string.insta_callback);
    Uri uri = getIntent().getData();
    if (uri != null && uri.toString().startsWith(CALLBACK_URL)) {
        String access_token = uri.getQueryParameter("access_token");
    }
    // Perform other operations here.
}
```

Ahora ha recuperado el `access_token` de Instagram, que se usa en varios puntos finales API de Instagram.

Lea URL de devolución de llamada en línea: https://riptutorial.com/es/android/topic/4790/url-de-devolucion-de-llamada

Capítulo 250: Utilidades de tiempo

Examples

Convertir formato de fecha en milisegundos

Para convertir su fecha en formato dd / MM / aaaa en milisegundos, llame a esta función con datos como String

```
public long getMilliFromDate(String dateFormat) {
    Date date = new Date();
    SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
    try {
        date = formatter.parse(dateFormat);
    } catch (ParseException e) {
        e.printStackTrace();
    }
    System.out.println("Today is " + date);
    return date.getTime();
}
```

Este método convierte los milisegundos en la fecha del formato de sello de tiempo:

```
public String getTimeStamp(long timeinMillies) {
    String date = null;
    SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"); // modify format
    date = formatter.format(new Date(timeinMillies));
    System.out.println("Today is " + date);

    return date;
}
```

Este método convertirá determinados días, meses y años en milisegundos. Será muy ayudar al utilizar `Timpicker` o `Datepicker`

```
public static long getTimeInMillis(int day, int month, int year) {
    Calendar calendar = Calendar.getInstance();
    calendar.set(year, month, day);
    return calendar.getTimeInMillis();
}
```

Devolverá milisegundos desde la fecha.

```
public static String getNormalDate(long timeInMillies) {
    String date = null;
    SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
    date = formatter.format(timeInMillies);
    System.out.println("Today is " + date);
    return date;
}
```

Volverá la fecha actual

```
public static String getCurrentDate() {
    Calendar c = Calendar.getInstance();
    System.out.println("Current time => " + c.getTime());
    SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy");
    String formattedDate = df.format(c.getTime());
    return formattedDate;
}
```

Nota: Java proporciona números de compatibilidad de formato de fecha Patrón de fecha

Para comprobar dentro de un plazo

Este ejemplo ayudará a verificar que el tiempo dado esté dentro de un período o no.

Para comprobar que la hora es hoy, podemos usar la clase ***DateUtils.***

```
boolean isToday = DateUtils.isToday(timeInMillis);
```

Para comprobar el tiempo es dentro de una semana,

```
private static boolean isWithinWeek(final long millis) {
    return System.currentTimeMillis() - millis <= (DateUtils.WEEK_IN_MILLIS -
DateUtils.DAY_IN_MILLIS);
}
```

Para comprobar el tiempo es dentro de un año,

```
private static boolean isWithinYear(final long millis) {
    return System.currentTimeMillis() - millis <= DateUtils.YEAR_IN_MILLIS;
}
```

Para verificar la hora está dentro de un número de días del día incluyendo hoy,

```
public static boolean isWithinDay(long timeInMillis, int day) {
    long diff = System.currentTimeMillis() - timeInMillis;

    float dayCount = (float) (diff / DateUtils.DAY_IN_MILLIS);

    return dayCount < day;
}
```

Nota: DateUtils es **android.text.format.DateUtils**

GetCurrentRealTime

Esto calcula la hora actual del dispositivo y agrega / resta la diferencia entre la hora real y la del dispositivo

```
public static Calendar getCurrentRealTime() {
```

```
    long bootTime = networkTime - SystemClock.elapsedRealtime();
    Calendar calInstance = Calendar.getInstance();
    calInstance.setTimeZone(getUTCTimeZone());
    long currentDeviceTime = bootTime + SystemClock.elapsedRealtime();
    calInstance.setTimeInMillis(currentDeviceTime);
    return calInstance;
}
```

obtener zona horaria UTC.

```
public static TimeZone getUTCTimeZone() {
    return TimeZone.getTimeZone("GMT");
}
```

Lea Utilidades de tiempo en línea: https://riptutorial.com/es/android/topic/7138/utilidades-de-tiempo

Capítulo 251: Validación de correo electrónico

Examples

Validación de la dirección de correo electrónico

Agregue el siguiente método para verificar si una dirección de correo electrónico es válida o no:

```
private boolean isValidEmailId(String email){
  return Pattern.compile("^(([\\w-]+\\.)+[\\w-]+|([a-zA-Z]{1}|[\\w-]{2,}))@"
              + "((([0-1]?[0-9]{1,2}|25[0-5]|2[0-4][0-9])\\.([0-1]?"
              + "[0-9]{1,2}|25[0-5]|2[0-4][0-9])\\."
              + "([0-1]?[0-9]{1,2}|25[0-5]|2[0-4][0-9])\\.([0-1]?"
              + "[0-9]{1,2}|25[0-5]|2[0-4][0-9])){1}|"
              + "([a-zA-Z]+[\\w-]+\\.)+[a-zA-Z]{2,4})$").matcher(email).matches();
}
```

El método anterior se puede verificar fácilmente convirtiendo el texto de un widget `EditText` en una `String` :

```
if(isValidEmailId(edtEmailId.getText().toString().trim())){
  Toast.makeText(getApplicationContext(), "Valid Email Address.", Toast.LENGTH_SHORT).show();
}else{
  Toast.makeText(getApplicationContext(), "InValid Email Address.",
Toast.LENGTH_SHORT).show();
}
```

Validación de la dirección de correo electrónico con el uso de patrones

```
  if (Patterns.EMAIL_ADDRESS.matcher(email).matches()){
       Log.i("EmailCheck","It is valid");
  }
```

Lea Validación de correo electrónico en línea:
https://riptutorial.com/es/android/topic/5605/validacion-de-correo-electronico

Capítulo 252: VectorDrawable y AnimatedVectorDrawable

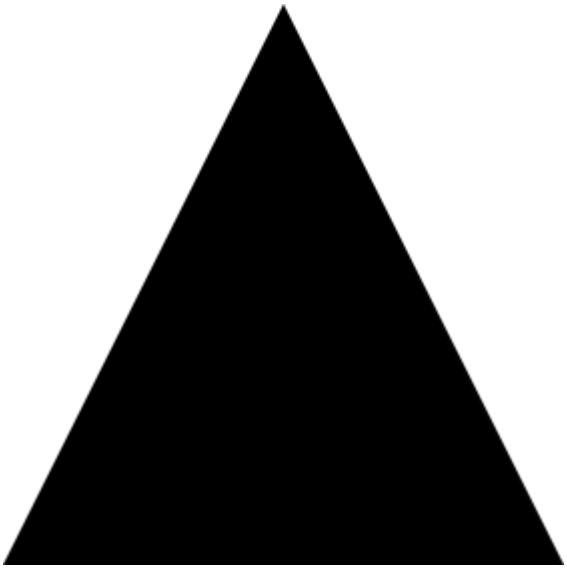
Examples

Básico VectorDrawable

Un `VectorDrawable` debe constar de al menos una etiqueta `<path>` define una forma

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">
    <path
        android:fillColor="#FF000000"
        android:pathData="M0,24 l12,-24 l12,24 z"/>
</vector>
```

Esto produciría un triángulo negro:



Utilizando

Un `<clip-path>` define una forma que actúa como una ventana, y solo permite que partes de un `<path>` muestren si están dentro de la forma `<clip-path>` y eliminen el resto.

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">
    <clip-path
```

```
        android:name="square clip path"
        android:pathData="M6,6 h12 v12 h-12 z"/>
    <path
        android:name="triangle"
        android:fillColor="#FF000000"
        android:pathData="M0,24 l12,-24 l12,24 z"/>

</vector>
```

En este caso, `<path>` produce un triángulo negro, pero `<clip-path>` define una forma cuadrada más pequeña, y solo permite que parte del triángulo se muestre a través de:



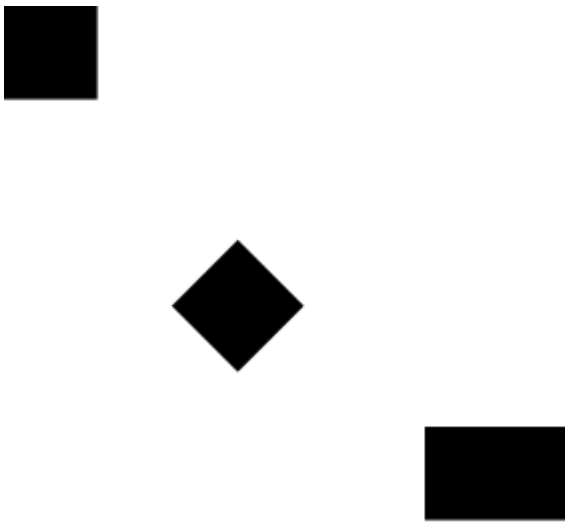
etiquetas

Una etiqueta `<group>` permite ajustar la escala, la rotación y la posición de uno o más elementos de un `VectorDrawable` :

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
android:width="24dp"
android:height="24dp"
android:viewportWidth="24.0"
android:viewportHeight="24.0">
    <path
        android:pathData="M0,0 h4 v4 h-4 z"
        android:fillColor="#FF000000"/>

    <group
        android:name="middle square group"
        android:translateX="10"
        android:translateY="10"
        android:rotation="45">
        <path
            android:pathData="M0,0 h4 v4 h-4 z"
            android:fillColor="#FF000000"/>
    </group>

    <group
        android:name="last square group"
        android:translateX="18"
        android:translateY="18"
```

```
        android:scaleX="1.5">
        <path
            android:pathData="M0,0 h4 v4 h-4 z"
            android:fillColor="#FF000000"/>
    </group>
</vector>
```

El código de ejemplo anterior contiene tres etiquetas `<path>` idénticas, y todas describen cuadrados negros. El primer cuadrado no está ajustado. El segundo cuadrado está envuelto en una etiqueta `<group>` que lo mueve y lo gira en 45 °. El tercer cuadrado está envuelto en una etiqueta `<group>` que lo mueve y lo estira horizontalmente en un 50%. El resultado es el siguiente:



Una etiqueta `<group>` puede contener múltiples etiquetas `<path>` y `<clip-path>` . Incluso puede contener otro `<group>` .

AnimatedVectorDrawable básico

Un `AnimatedVectorDrawable` requiere al menos 3 componentes:

- Un `VectorDrawable` que será manipulado.
- Un `objectAnimator` que define qué propiedad cambiar y cómo
- El propio `AnimatedVectorDrawable` que conecta el `objectAnimator` al `VectorDrawable` para crear la animación.

Lo siguiente crea un triángulo que cambia su color de negro a rojo.

The `VectorDrawable` , nombre de archivo: `triangle_vector_drawable.xml`

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">

    <path
        android:name="triangle"
        android:fillColor="@android:color/black"
```

```
        android:pathData="M0,24 l12,-24 l12,24 z"/>

</vector>
```

El `objectAnimator` , nombre de archivo: `color_change_animator.xml`

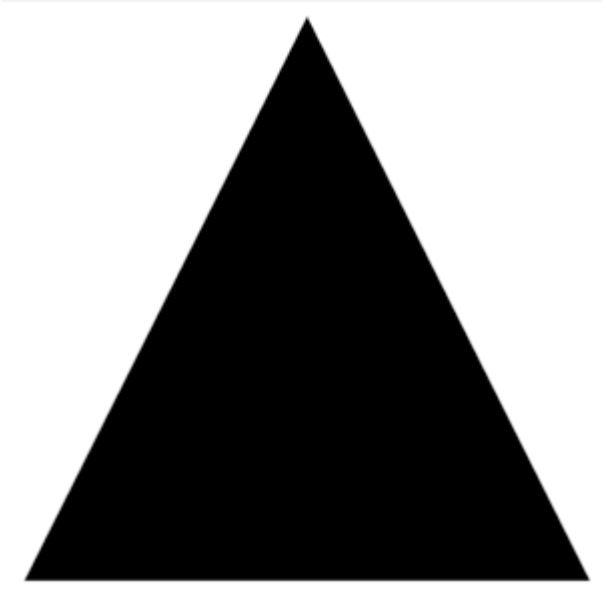
```
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:propertyName="fillColor"
    android:duration="2000"
    android:repeatCount="infinite"
    android:valueFrom="@android:color/black"
    android:valueTo="@android:color/holo_red_light"/>
```

El `AnimatedVectorDrawable` , nombre de archivo: `triangle_animated_vector.xml`

```
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/triangle_vector_drawable">

    <target
        android:animation="@animator/color_change_animator"
        android:name="triangle"/>

</animated-vector>
```

Tenga en cuenta que `<target>` especifica `android:name="triangle"` que coincide con `<path>` en `VectorDrawable` . Un `VectorDrawable` puede contener múltiples elementos y la propiedad `android:name` se usa para definir a qué elemento se dirige.

Resultado:



Utilizando trazos

El uso del trazo SVG facilita la creación de un Vector dibujable con una longitud de trazo unificada, según las pautas de Diseño de materiales :

> Los pesos de trazo consistentes son clave para unificar la familia de iconos del sistema en general. Mantenga un ancho de 2dp para todas las instancias de trazo, incluidas las curvas, los ángulos y los trazos interiores y exteriores.

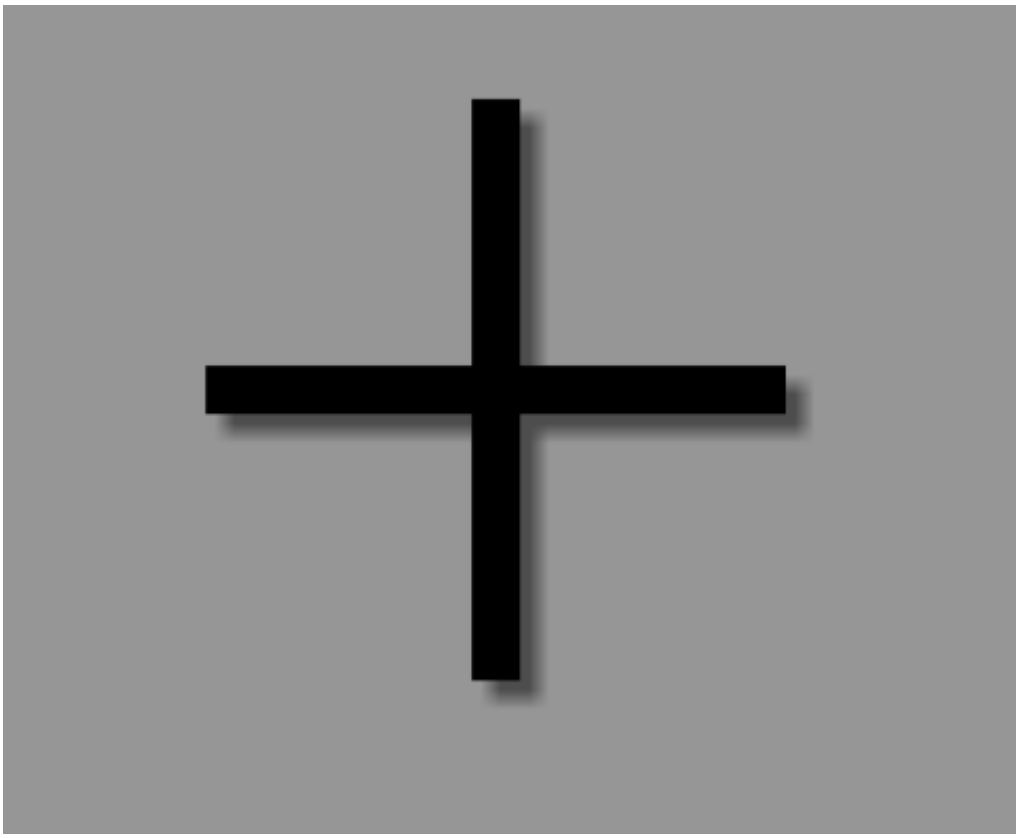
Entonces, por ejemplo, esta es la forma en que crearía un signo "más" utilizando los trazos:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
android:width="24dp"
android:height="24dp"
android:viewportHeight="24.0"
android:viewportWidth="24.0">
    <path
        android:fillColor="#FF000000"
        android:strokeColor="#F000"
        android:strokeWidth="2"
        android:pathData="M12,0 V24 M0,12 H24" />
</vector>
```

- `strokeColor` define el color del trazo.
- `strokeWidth` define el ancho (en dp) del golpe (2dp en este caso, como lo sugieren las directrices).
- `pathData` es donde describimos nuestra imagen SVG:
- `M12,0` mueve el "cursor" a la posición 12,0
- `V24` crea una línea vertical a la posición 12, 24

etc., consulte la documentación de SVG y este útil tutorial "Ruta SVG" de w3schools para obtener más información sobre los comandos de ruta específicos.

Como resultado, obtuvimos este signo más sencillo:



Esto es **especialmente útil** para crear un `AnimatedVectorDrawable` , ya que ahora está operando con un solo trazo con una longitud unificada, en lugar de una ruta por lo demás complicada.

Compatibilidad de vectores a través de AppCompat

Algunos requisitos previos en el `build.gradle` para que los vectores funcionen hasta API 7 para VectorDrawables y API 13 para AnimatedVectorDrawables (con algunas advertencias actualmente):

```
//Build Tools has to be 24+
buildToolsVersion '24.0.0'

defaultConfig {
    vectorDrawables.useSupportLibrary = true
    generatedDensities = []
    aaptOptions {
        additionalParameters "--no-version-vectors"
    }
}

dependencies {
    compile 'com.android.support:appcompat-v7:24.1.1'
}
```

En su `layout.xml` :

```
<ImageView
    android:id="@+id/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    appCompat:src="@drawable/vector_drawable"
    android:contentDescription="@null" />
```

Lea VectorDrawable y AnimatedVectorDrawable en línea:
https://riptutorial.com/es/android/topic/1627/vectordrawable-y-animatedvectordrawable

Capítulo 253: Versiones de android

Observaciones

Nombre	versión de Android	Fecha de lanzamiento	Nivel de API	Build.VERSION_CODES
Pastel de ángel (alfa)	1.0	23 de septiembre de 2008	1	BASE
Battenberg (Beta)	1.1	9 de febrero de 2009	2	BASE_1_1
Magdalena	1.5	30 de abril de 2009	3	CUPCAKE
Rosquilla	1.6	15 de septiembre de 2009	4	ROSQUILLA
Eclair	2.0	26 de octubre de 2009	5	ECLAIR
	2.0.1	3 de diciembre de 2009	6	ECLAIR_0_1
	2.1	12 de enero de 2010	7	ECLAIR_MR1
Froyo	2.2	20 de mayo de 2010	8	FROYO
Pan de jengibre	2.3	6 de diciembre de 2010	9	PAN DE JENGIBRE
	2.3.3	9 de febrero de 2011	10	GINGERBREAD_MR1
Panal	3.0	22 de febrero de 2011	11	PANAL
	3.1	10 de mayo de 2011	12	HONEYCOMB_MR2
	3.2	15 de julio de 2011	13	HONEYCOMB_MR1

Nombre	versión de Android	Fecha de lanzamiento	Nivel de API	Build.VERSION_CODES
Sandwich De Helado	4.0	19 de octubre de 2011	14	ICE_CREAM_SANDWICH
	4.0.3	16 de diciembre de 2011	15	ICE_CREAM_SANDWICH_MR1
Frijol de jalea	4.1	9 de julio de 2012	dieciséis	FRIJOL DE JALEA
	4.2	13 de noviembre de 2012	17	JELLY_BEAN_MR1
	4.3	24 de julio de 2013	18	JELLY_BEAN_MR2
Kit Kat	4.4	31 de octubre de 2013	19	KIT KAT
		25 de julio de 2014	20	KITKAT_WATCH
Pirulí	5.0	17 de octubre de 2014	21	PIRULÍ
	5.1	9 de marzo de 2015	22	LOLLIPOP_MR1
Malvavisco	6.0	5 de octubre de 2015	23	METRO
Turrón	7.0	22 de agosto de 2016	24	norte
	7.1.1	5 de diciembre de 2016	25	N_MR1

Examples

Comprobación de la versión de Android en el dispositivo en tiempo de ejecución

`Build.VERSION_CODES` es una enumeración de los códigos de versión SDK conocidos actualmente.

Para ejecutar condicionalmente el código basado en la versión de Android del dispositivo, use la

anotación `TargetApi` para evitar errores de pelusa y verifique la versión de compilación antes de ejecutar el código específico para el nivel de API.

Este es un ejemplo de cómo usar una clase que se introdujo en API-23, en un proyecto que admite niveles de API inferiores a 23:

```
@Override
@TargetApi(23)
public void onResume() {
    super.onResume();
    if (android.os.Build.VERSION.SDK_INT <= Build.VERSION_CODES.M) {
        //run Marshmallow code
        FingerprintManager fingerprintManager =
this.getSystemService(FingerprintManager.class);
        //.....................
    }
}
```

Lea Versiones de android en línea: https://riptutorial.com/es/android/topic/3264/versiones-de-android

Capítulo 254: Versiones de Project SDK

Introducción

Una aplicación de Android debe ejecutarse en todo tipo de dispositivos. Cada dispositivo puede tener una versión diferente en Android ejecutándose en él.

Ahora, es posible que cada versión de Android no sea compatible con todas las funciones que su aplicación requiere, por lo que al crear una aplicación, debe tener en cuenta la versión mínima y máxima de Android.

Parámetros

Parámetro	Detalles
Versión SDK	La versión del SDK para cada campo es el entero del nivel de API del SDK de la versión de Android. Por ejemplo, Froyo (Android 2.2) corresponde al nivel de API 8. Estos enteros también se definen en `Build.VERSION_CODES` .

Observaciones

Hay cuatro versiones relevantes de SDK en cada proyecto:

- `targetSdkVersion` es la última versión de Android con la que has probado.

  El marco utilizará `targetSdkVersion` para determinar cuándo habilitar ciertos comportamientos de compatibilidad. Por ejemplo, la API de nivel 23 o superior le dará acceso al modelo de permisos de tiempo de ejecución .

- `minSdkVersion` es la versión mínima de Android que admite su aplicación. Los usuarios que ejecuten cualquier versión de Android anterior a esta versión no podrán instalar su aplicación o verla en Play Store.

- `maxSdkVersion` es la versión máxima de Android que admite su aplicación. Los usuarios que ejecuten cualquier versión de Android más nueva que esta versión no podrán instalar su aplicación o verla en Play Store. Por lo general, no se debe usar, ya que la mayoría de las aplicaciones funcionarán en versiones más recientes de Android sin ningún esfuerzo adicional.

- `compileSdkVersion` es la versión del SDK de Android con la que se compilará su aplicación. En general, debería ser la última versión de Android que se haya lanzado públicamente. Esto define a qué API puede acceder al escribir su código. No puede llamar a los métodos introducidos en el nivel de API 23 si su `compileSdkVersion` se establece en 22 o inferior.

Examples

Definir versiones de proyecto SDK

En su archivo `build.gradle` del módulo principal (**aplicación**), defina su número de versión mínimo y objetivo.

```
android {
    //the version of sdk source used to compile your project
    compileSdkVersion 23

    defaultConfig {
        //the minimum sdk version required by device to run your app
        minSdkVersion 19
        //you normally don't need to set max sdk limit so that your app can support future
versions of android without updating app
        //maxSdkVersion 23
        //
        //the latest sdk version of android on which you are targeting(building and testing)
your app, it should be same as compileSdkVersion
        targetSdkVersion 23
    }
}
```

Lea Versiones de Project SDK en línea: https://riptutorial.com/es/android/topic/162/versiones-de-project-sdk

Capítulo 255: Vibración

Examples

Empezando con la vibración

Conceder permiso de vibración

Antes de comenzar a implementar el código, debe agregar permiso en el manifiesto de Android:

```
<uses-permission android:name="android.permission.VIBRATE"/>
```

Biblioteca de vibraciones de importación

```
import android.os.Vibrator;
```

Obtener instancia de vibrador de contexto

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

Comprobar dispositivo tiene vibrador

```
void boolean isHaveVibrate(){
    if (vibrator.hasVibrator()) {
        return true;
    }
    return false;
}
```

Vibrar indefinidamente

usando el *patrón vibrar (largo [], repetición int)*

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);

// Start time delay
// Vibrate for 500 milliseconds
// Sleep for 1000 milliseconds
long[] pattern = {0, 500, 1000};

// 0 meaning is repeat indefinitely
vibrator.vibrate(pattern, 0);
```

Patrones de vibracion

Puede crear patrones de vibración pasando una serie de largos, cada uno de los cuales representa una duración en milisegundos. El primer número es el tiempo de retardo de inicio. Cada entrada de matriz luego alterna entre vibrar, dormir, vibrar, dormir, etc.

El siguiente ejemplo demuestra este patrón:

- vibra 100 milisegundos y duerme 1000 milisegundos
- vibra 200 milisegundos y duerme 2000 milisegundos

```
long[] pattern = {0, 100, 1000, 200, 2000};
```

Para hacer que el patrón se repita, pase el índice a la matriz de patrones en la que se iniciará la repetición, o `-1` para deshabilitar la repetición.

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(pattern, -1); // does not repeat
vibrator.vibrate(pattern,  0); // repeats forever
```

Dejar de vibrar

Si quieres dejar de vibrar por favor llama

```
vibrator.cancel();
```

Vibrar por una vez

utilizando el *vibrar (milisegundos largos)*

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(500);
```

Lea Vibración en línea: https://riptutorial.com/es/android/topic/3359/vibracion

Capítulo 256: VideoView

Examples

VideoView Crear

Encuentra VideoView en Actividad y agrega video en él.

```
VideoView videoView = (VideoView) .findViewById(R.id.videoView);
videoView.setVideoPath(pathToVideo);
```

Comience a reproducir el video.

```
videoView.start();
```

Definir VideoView en el archivo de diseño XML.

```
<VideoView
    android:id="@+id/videoView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center" />
```

Reproducir video desde la URL con el uso de VideoView

```
 videoView.setVideoURI(Uri.parse("http://example.com/examplevideo.mp4"));
 videoView.requestFocus();

 videoView.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
      @Override
      public void onCompletion(MediaPlayer mediaPlayer) {
      }
 });

 videoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
     @Override
     public void onPrepared(MediaPlayer mediaPlayer) {
          videoView.start();
          mediaPlayer.setOnVideoSizeChangedListener(new
MediaPlayer.OnVideoSizeChangedListener() {
                @Override
                public void onVideoSizeChanged(MediaPlayer mp, int width, int height) {
                      MediaController mediaController = new
MediaController(ActivityName.this);
                      videoView.setMediaController(mediaController);
                      mediaController.setAnchorView(videoView);
                }
          });
     }
 });

 videoView.setOnErrorListener(new MediaPlayer.OnErrorListener() {
```

```
        @Override
        public boolean onError(MediaPlayer mediaPlayer, int i, int i1) {
             return false;
        }
});
```

Lea VideoView en línea: https://riptutorial.com/es/android/topic/8962/videoview

Capítulo 257: VideoView optimizado

Introducción

La reproducción de un video usando un `VideoView` que extiende `SurfaceView` dentro de una fila de un `ListView` parece funcionar al principio, hasta que el usuario intenta desplazarse por la lista. Tan pronto como la lista comienza a desplazarse, el video se vuelve negro (a veces se muestra en blanco). Sigue reproduciéndose en segundo plano, pero ya no puedes verlo porque muestra el resto del video como una caja negra. Con el Optimizado VideoView personalizado, los videos se reproducirán en el desplazamiento en el `ListView` al igual que nuestro Instagram, Facebook, Twitter.

Examples

VideoView optimizado en ListView

Este es el `VideoView` personalizado que necesita para tenerlo en su paquete.

Diseño de VideoView personalizado:

```
<your.packagename.VideoView
    android:id="@+id/video_view"
    android:layout_width="300dp"
    android:layout_height="300dp" />
```

Código para `VideoView` optimizado personalizado:

```
package your.package.com.whateveritis;

import android.content.Context;
import android.content.Intent;
import android.graphics.SurfaceTexture;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.media.MediaPlayer.OnErrorListener;
import android.media.MediaPlayer.OnInfoListener;
import android.net.Uri;
import android.util.AttributeSet;
import android.util.Log;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.Surface;
import android.view.TextureView;
import android.view.View;
import android.widget.MediaController;
import android.widget.MediaController.MediaPlayerControl;

import java.io.IOException;
```

```
/**
 * VideoView is used to play video, just like
 * {@link android.widget.VideoView VideoView}. We define a custom view, because
 * we could not use {@link android.widget.VideoView VideoView} in ListView. <br/>
 * VideoViews inside ScrollViews do not scroll properly. Even if you use the
 * workaround to set the background color, the MediaController does not scroll
 * along with the VideoView. Also, the scrolling video looks horrendous with the
 * workaround, lots of flickering.
 *
 * @author leo
 */
public class VideoView extends TextureView implements MediaPlayerControl {

    private static final String TAG = "tag";

    // all possible internal states
    private static final int STATE_ERROR = -1;
    private static final int STATE_IDLE = 0;
    private static final int STATE_PREPARING = 1;
    private static final int STATE_PREPARED = 2;
    private static final int STATE_PLAYING = 3;
    private static final int STATE_PAUSED = 4;
    private static final int STATE_PLAYBACK_COMPLETED = 5;

    // currentState is a VideoView object's current state.
    // targetState is the state that a method caller intends to reach.
    // For instance, regardless the VideoView object's current state,
    // calling pause() intends to bring the object to a target state
    // of STATE_PAUSED.
    private int mCurrentState = STATE_IDLE;
    private int mTargetState = STATE_IDLE;

    // Stuff we need for playing and showing a video
    private MediaPlayer mMediaPlayer;
    private int mVideoWidth;
    private int mVideoHeight;
    private int mSurfaceWidth;
    private int mSurfaceHeight;
    private SurfaceTexture mSurfaceTexture;
    private Surface mSurface;
    private MediaController mMediaController;
    private MediaPlayer.OnCompletionListener mOnCompletionListener;
    private MediaPlayer.OnPreparedListener mOnPreparedListener;

    private MediaPlayer.OnErrorListener mOnErrorListener;
    private MediaPlayer.OnInfoListener mOnInfoListener;

    private int mSeekWhenPrepared; // recording the seek position while
    // preparing
    private int mCurrentBufferPercentage;
    private int mAudioSession;
    private Uri mUri;

    private Context mContext;

    public VideoView(final Context context) {
        super(context);
        mContext = context;
        initVideoView();
    }
```

```
public VideoView(final Context context, final AttributeSet attrs) {
    super(context, attrs);
    mContext = context;
    initVideoView();
}

public VideoView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
    mContext = context;
    initVideoView();
}

public void initVideoView() {
    mVideoHeight = 0;
    mVideoWidth = 0;
    setFocusable(false);
    setSurfaceTextureListener(mSurfaceTextureListener);
}

public int resolveAdjustedSize(int desiredSize, int measureSpec) {
    int result = desiredSize;
    int specMode = MeasureSpec.getMode(measureSpec);
    int specSize = MeasureSpec.getSize(measureSpec);

    switch (specMode) {
        case MeasureSpec.UNSPECIFIED:
        /*
         * Parent says we can be as big as we want. Just don't be larger
         * than max size imposed on ourselves.
         */
            result = desiredSize;
            break;

        case MeasureSpec.AT_MOST:
        /*
         * Parent says we can be as big as we want, up to specSize. Don't be
         * larger than specSize, and don't be larger than the max size
         * imposed on ourselves.
         */
            result = Math.min(desiredSize, specSize);
            break;

        case MeasureSpec.EXACTLY:
            // No choice. Do what we are told.
            result = specSize;
            break;
    }
    return result;
}

public void setVideoPath(String path) {
    Log.d(TAG, "Setting video path to: " + path);
    setVideoURI(Uri.parse(path));
}

public void setVideoURI(Uri _videoURI) {
    mUri = _videoURI;
    mSeekWhenPrepared = 0;
    requestLayout();
    invalidate();
    openVideo();
```

```
    }

    public Uri getUri() {
        return mUri;
    }

    public void setSurfaceTexture(SurfaceTexture _surfaceTexture) {
        mSurfaceTexture = _surfaceTexture;
    }

    public void openVideo() {
        if ((mUri == null) || (mSurfaceTexture == null)) {
            Log.d(TAG, "Cannot open video, uri or surface texture is null.");
            return;
        }
        // Tell the music playback service to pause
        // TODO: these constants need to be published somewhere in the
        // framework.
        Intent i = new Intent("com.android.music.musicservicecommand");
        i.putExtra("command", "pause");
        mContext.sendBroadcast(i);
        release(false);
        try {
            mSurface = new Surface(mSurfaceTexture);
            mMediaPlayer = new MediaPlayer();
            if (mAudioSession != 0) {
                mMediaPlayer.setAudioSessionId(mAudioSession);
            } else {
                mAudioSession = mMediaPlayer.getAudioSessionId();
            }

            mMediaPlayer.setOnBufferingUpdateListener(mBufferingUpdateListener);
            mMediaPlayer.setOnCompletionListener(mCompleteListener);
            mMediaPlayer.setOnPreparedListener(mPreparedListener);
            mMediaPlayer.setOnErrorListener(mErrorListener);
            mMediaPlayer.setOnInfoListener(mOnInfoListener);
            mMediaPlayer.setOnVideoSizeChangedListener(mVideoSizeChangedListener);

            mMediaPlayer.setSurface(mSurface);
            mCurrentBufferPercentage = 0;
            mMediaPlayer.setDataSource(mContext, mUri);

            mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
            mMediaPlayer.setScreenOnWhilePlaying(true);

            mMediaPlayer.prepareAsync();
            mCurrentState = STATE_PREPARING;
        } catch (IllegalStateException e) {
            mCurrentState = STATE_ERROR;
            mTargetState = STATE_ERROR;
            String msg = (e.getMessage()==null)?"":e.getMessage();
            Log.i("",msg);  // TODO auto-generated catch block
        } catch (IOException e) {
            mCurrentState = STATE_ERROR;
            mTargetState = STATE_ERROR;
            String msg = (e.getMessage()==null)?"":e.getMessage();
            Log.i("",msg);  // TODO auto-generated catch block
        }
    }

    public void stopPlayback() {
```

```
        if (mMediaPlayer != null) {
            mMediaPlayer.stop();
            mMediaPlayer.release();
            mMediaPlayer = null;
            if (null != mMediaControllListener) {
                mMediaControllListener.onStop();
            }
        }
    }

    public void setMediaController(MediaController controller) {
        if (mMediaController != null) {
            mMediaController.hide();
        }
        mMediaController = controller;
        attachMediaController();
    }

    private void attachMediaController() {
        if (mMediaPlayer != null && mMediaController != null) {
            mMediaController.setMediaPlayer(this);
            View anchorView = this.getParent() instanceof View ? (View) this.getParent() :
this;
            mMediaController.setAnchorView(anchorView);
            mMediaController.setEnabled(isInPlaybackState());
        }
    }

    private void release(boolean cleartargetstate) {
        Log.d(TAG, "Releasing media player.");
        if (mMediaPlayer != null) {
            mMediaPlayer.reset();
            mMediaPlayer.release();
            mMediaPlayer = null;
            mCurrentState = STATE_IDLE;
            if (cleartargetstate) {
                mTargetState = STATE_IDLE;
            }
        } else {
            Log.d(TAG, "Media player was null, did not release.");
        }
    }

    @Override
    protected void onMeasure(final int widthMeasureSpec, final int heightMeasureSpec) {
        // Will resize the view if the video dimensions have been found.
        // video dimensions are found after onPrepared has been called by
        // MediaPlayer
        int width = getDefaultSize(mVideoWidth, widthMeasureSpec);
        int height = getDefaultSize(mVideoHeight, heightMeasureSpec);
        if ((mVideoWidth > 0) && (mVideoHeight > 0)) {
            if ((mVideoWidth * height) > (width * mVideoHeight)) {
                Log.d(TAG, "Video too tall, change size.");
                height = (width * mVideoHeight) / mVideoWidth;
            } else if ((mVideoWidth * height) < (width * mVideoHeight)) {
                Log.d(TAG, "Video too wide, change size.");
                width = (height * mVideoWidth) / mVideoHeight;
            } else {
                Log.d(TAG, "Aspect ratio is correct.");
            }
        }
```

```
        setMeasuredDimension(width, height);
    }

    @Override
    public boolean onTouchEvent(MotionEvent ev) {
        if (isInPlaybackState() && mMediaController != null) {
            toggleMediaControlsVisiblity();
        }
        return false;
    }

    @Override
    public boolean onTrackballEvent(MotionEvent ev) {
        if (isInPlaybackState() && mMediaController != null) {
            toggleMediaControlsVisiblity();
        }
        return false;
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        boolean isKeyCodeSupported = keyCode != KeyEvent.KEYCODE_BACK && keyCode !=
KeyEvent.KEYCODE_VOLUME_UP && keyCode != KeyEvent.KEYCODE_VOLUME_DOWN
                && keyCode != KeyEvent.KEYCODE_VOLUME_MUTE && keyCode != KeyEvent.KEYCODE_MENU
&& keyCode != KeyEvent.KEYCODE_CALL
                && keyCode != KeyEvent.KEYCODE_ENDCALL;
        if (isInPlaybackState() && isKeyCodeSupported && mMediaController != null) {
            if (keyCode == KeyEvent.KEYCODE_HEADSETHOOK || keyCode ==
KeyEvent.KEYCODE_MEDIA_PLAY_PAUSE) {
                if (mMediaPlayer.isPlaying()) {
                    pause();
                    mMediaController.show();
                } else {
                    start();
                    mMediaController.hide();
                }
                return true;
            } else if (keyCode == KeyEvent.KEYCODE_MEDIA_PLAY) {
                if (!mMediaPlayer.isPlaying()) {
                    start();
                    mMediaController.hide();
                }
                return true;
            } else if (keyCode == KeyEvent.KEYCODE_MEDIA_STOP || keyCode ==
KeyEvent.KEYCODE_MEDIA_PAUSE) {
                if (mMediaPlayer.isPlaying()) {
                    pause();
                    mMediaController.show();
                }
                return true;
            } else {
                toggleMediaControlsVisiblity();
            }
        }

        return super.onKeyDown(keyCode, event);
    }

    private void toggleMediaControlsVisiblity() {
        if (mMediaController.isShowing()) {
            mMediaController.hide();
```

```
        } else {
            mMediaController.show();
        }
    }

    public void start() {
        // This can potentially be called at several points, it will go through
        // when all conditions are ready
        // 1. When setting the video URI
        // 2. When the surface becomes available
        // 3. From the activity
        if (isInPlaybackState()) {
            mMediaPlayer.start();
            mCurrentState = STATE_PLAYING;
            if (null != mMediaControllListener) {
                mMediaControllListener.onStart();
            }
        } else {
            Log.d(TAG, "Could not start. Current state " + mCurrentState);
        }
        mTargetState = STATE_PLAYING;
    }

    public void pause() {
        if (isInPlaybackState()) {
            if (mMediaPlayer.isPlaying()) {
                mMediaPlayer.pause();
                mCurrentState = STATE_PAUSED;
                if (null != mMediaControllListener) {
                    mMediaControllListener.onPause();
                }
            }
        }
        mTargetState = STATE_PAUSED;
    }

    public void suspend() {
        release(false);
    }

    public void resume() {
        openVideo();
    }

    @Override
    public int getDuration() {
        if (isInPlaybackState()) {
            return mMediaPlayer.getDuration();
        }

        return -1;
    }

    @Override
    public int getCurrentPosition() {
        if (isInPlaybackState()) {
            return mMediaPlayer.getCurrentPosition();
        }
        return 0;
    }
```

```
    @Override
    public void seekTo(int msec) {
        if (isInPlaybackState()) {
            mMediaPlayer.seekTo(msec);
            mSeekWhenPrepared = 0;
        } else {
            mSeekWhenPrepared = msec;
        }
    }

    @Override
    public boolean isPlaying() {
        return isInPlaybackState() && mMediaPlayer.isPlaying();
    }

    @Override
    public int getBufferPercentage() {
        if (mMediaPlayer != null) {
            return mCurrentBufferPercentage;
        }
        return 0;
    }

    private boolean isInPlaybackState() {
        return ((mMediaPlayer != null) && (mCurrentState != STATE_ERROR) && (mCurrentState !=
STATE_IDLE) && (mCurrentState != STATE_PREPARING));
    }

    @Override
    public boolean canPause() {
        return false;
    }

    @Override
    public boolean canSeekBackward() {
        return false;
    }

    @Override
    public boolean canSeekForward() {
        return false;
    }

    @Override
    public int getAudioSessionId() {
        if (mAudioSession == 0) {
            MediaPlayer foo = new MediaPlayer();
            mAudioSession = foo.getAudioSessionId();
            foo.release();
        }
        return mAudioSession;
    }

    // Listeners
    private MediaPlayer.OnBufferingUpdateListener mBufferingUpdateListener = new
MediaPlayer.OnBufferingUpdateListener() {
        @Override
        public void onBufferingUpdate(final MediaPlayer mp, final int percent) {
            mCurrentBufferPercentage = percent;
        }
    };
```

```
    private MediaPlayer.OnCompletionListener mCompleteListener = new
MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(final MediaPlayer mp) {
            mCurrentState = STATE_PLAYBACK_COMPLETED;
            mTargetState = STATE_PLAYBACK_COMPLETED;
            mSurface.release();

            if (mMediaController != null) {
                mMediaController.hide();
            }

            if (mOnCompletionListener != null) {
                mOnCompletionListener.onCompletion(mp);
            }

            if (mMediaControllListener != null) {
                mMediaControllListener.onComplete();
            }
        }
    };

    private MediaPlayer.OnPreparedListener mPreparedListener = new
MediaPlayer.OnPreparedListener() {
        @Override
        public void onPrepared(final MediaPlayer mp) {
            mCurrentState = STATE_PREPARED;

            mMediaController = new MediaController(getContext());

            if (mOnPreparedListener != null) {
                mOnPreparedListener.onPrepared(mMediaPlayer);
            }
            if (mMediaController != null) {
                mMediaController.setEnabled(true);
                //mMediaController.setAnchorView(getRootView());
            }

            mVideoWidth = mp.getVideoWidth();
            mVideoHeight = mp.getVideoHeight();

            int seekToPosition = mSeekWhenPrepared; // mSeekWhenPrepared may be
            // changed after seekTo()
            // call
            if (seekToPosition != 0) {
                seekTo(seekToPosition);
            }

            requestLayout();
            invalidate();
            if ((mVideoWidth != 0) && (mVideoHeight != 0)) {
                if (mTargetState == STATE_PLAYING) {
                    mMediaPlayer.start();
                    if (null != mMediaControllListener) {
                        mMediaControllListener.onStart();
                    }
                }
            } else {
                if (mTargetState == STATE_PLAYING) {
                    mMediaPlayer.start();
```

```
                    if (null != mMediaControllListener) {
                        mMediaControllListener.onStart();
                    }
                }
            }
        }
    };

    private MediaPlayer.OnVideoSizeChangedListener mVideoSizeChangedListener = new
MediaPlayer.OnVideoSizeChangedListener() {
        @Override
        public void onVideoSizeChanged(final MediaPlayer mp, final int width, final int
height) {
            mVideoWidth = mp.getVideoWidth();
            mVideoHeight = mp.getVideoHeight();
            if (mVideoWidth != 0 && mVideoHeight != 0) {
                requestLayout();
            }
        }
    };

    private MediaPlayer.OnErrorListener mErrorListener = new MediaPlayer.OnErrorListener() {
        @Override
        public boolean onError(final MediaPlayer mp, final int what, final int extra) {
            Log.d(TAG, "Error: " + what + "," + extra);
            mCurrentState = STATE_ERROR;
            mTargetState = STATE_ERROR;

            if (mMediaController != null) {
                mMediaController.hide();
            }

            /* If an error handler has been supplied, use it and finish. */
            if (mOnErrorListener != null) {
                if (mOnErrorListener.onError(mMediaPlayer, what, extra)) {
                    return true;
                }
            }

            /*
             * Otherwise, pop up an error dialog so the user knows that
             * something bad has happened. Only try and pop up the dialog if
             * we're attached to a window. When we're going away and no longer
             * have a window, don't bother showing the user an error.
             */
            if (getWindowToken() != null) {

//                new AlertDialog.Builder(mContext).setMessage("Error: " + what + "," +
extra).setPositiveButton("OK", new DialogInterface.OnClickListener() {
//                    public void onClick(DialogInterface dialog, int whichButton) {
//                        /*
//                         * If we get here, there is no onError listener, so at
//                         * least inform them that the video is over.
//                         */
//                        if (mOnCompletionListener != null) {
//                            mOnCompletionListener.onCompletion(mMediaPlayer);
//                        }
//                    }
//                }).setCancelable(false).show();
            }
            return true;
```

```
        }
    };

    SurfaceTextureListener mSurfaceTextureListener = new SurfaceTextureListener() {
        @Override
        public void onSurfaceTextureAvailable(final SurfaceTexture surface, final int width,
final int height) {
            Log.d(TAG, "onSurfaceTextureAvailable.");
            mSurfaceTexture = surface;
            openVideo();
        }

        @Override
        public void onSurfaceTextureSizeChanged(final SurfaceTexture surface, final int width,
final int height) {
            Log.d(TAG, "onSurfaceTextureSizeChanged: " + width + '/' + height);
            mSurfaceWidth = width;
            mSurfaceHeight = height;
            boolean isValidState = (mTargetState == STATE_PLAYING);
            boolean hasValidSize = (mVideoWidth == width && mVideoHeight == height);
            if (mMediaPlayer != null && isValidState && hasValidSize) {
                if (mSeekWhenPrepared != 0) {
                    seekTo(mSeekWhenPrepared);
                }
                start();
            }
        }

        @Override
        public boolean onSurfaceTextureDestroyed(final SurfaceTexture surface) {

            mSurface = null;
            if (mMediaController != null)
                mMediaController.hide();
            release(true);
            return true;
        }

        @Override
        public void onSurfaceTextureUpdated(final SurfaceTexture surface) {

        }
    };

    /**
     * Register a callback to be invoked when the media file is loaded and ready
     * to go.
     *
     * @param l The callback that will be run
     */
    public void setOnPreparedListener(MediaPlayer.OnPreparedListener l) {
        mOnPreparedListener = l;
    }

    /**
     * Register a callback to be invoked when the end of a media file has been
     * reached during playback.
     *
     * @param l The callback that will be run
     */
    public void setOnCompletionListener(OnCompletionListener l) {
```

```
        mOnCompletionListener = l;
    }

    /**
     * Register a callback to be invoked when an error occurs during playback or
     * setup. If no listener is specified, or if the listener returned false,
     * VideoView will inform the user of any errors.
     *
     * @param l The callback that will be run
     */
    public void setOnErrorListener(OnErrorListener l) {
        mOnErrorListener = l;
    }

    /**
     * Register a callback to be invoked when an informational event occurs
     * during playback or setup.
     *
     * @param l The callback that will be run
     */
    public void setOnInfoListener(OnInfoListener l) {
        mOnInfoListener = l;
    }

    public static interface MediaControllListener {
        public void onStart();

        public void onPause();

        public void onStop();

        public void onComplete();
    }

    MediaControllListener mMediaControllListener;

    public void setMediaControllListener(MediaControllListener mediaControllListener) {
        mMediaControllListener = mediaControllListener;
    }


    @Override
    public void setVisibility(int visibility) {
        System.out.println("setVisibility: " + visibility);
        super.setVisibility(visibility);
    }
}
```

Ayuda de este repositorio de gitub . Aunque tiene algunos problemas, tal como estaba escrito hace 3 años, logré solucionarlos por mi cuenta como se indicó anteriormente.

Lea VideoView optimizado en línea: https://riptutorial.com/es/android/topic/10638/videoview-optimizado

Capítulo 258: ViewFlipper

Introducción

Un `ViewFlipper` es un `ViewAnimator` que cambia entre dos o más vistas que se le han agregado. Solo se muestra un niño a la vez. Si se solicita, `ViewFlipper` puede `ViewFlipper` automáticamente entre cada niño en un intervalo regular.

Examples

ViewFlipper con imagen deslizante

Archivo XML:

```
<ViewFlipper
    android:id="@+id/viewflip"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:layout_weight="1"
    />
```

Código java:

```
public class BlankFragment extends Fragment{
    ViewFlipper viewFlipper;
    FragmentManager fragmentManager;
    int gallery_grid_Images[] = {drawable.image1, drawable.image2, drawable.image3,
            drawable.image1, drawable.image2, drawable.image3, drawable.image1,
            drawable.image2, drawable.image3, drawable.image1
    };

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState){
        View rootView = inflater.inflate(fragment_blank, container, false);
        viewFlipper = (ViewFlipper)rootView.findViewById(R.id.viewflip);
        for(int i=0; i<gallery_grid_Images.length; i++){
            // This will create dynamic image views and add them to the ViewFlipper.
            setFlipperImage(gallery_grid_Images[i]);
        }
        return rootView;
    }

    private void setFlipperImage(int res) {
        Log.i("Set Filpper Called", res+"");
        ImageView image = new ImageView(getContext());
        image.setBackgroundResource(res);
        viewFlipper.addView(image);
        viewFlipper.setFlipInterval(1000);
        viewFlipper.setAutoStart(true);
    }
}
```

Lea ViewFlipper en línea: https://riptutorial.com/es/android/topic/9032/viewflipper

Capítulo 259: ViewPager

Introducción

ViewPager es un administrador de diseño que permite al usuario voltear hacia la izquierda y hacia la derecha a través de las páginas de datos. Se usa con más frecuencia junto con Fragmento, que es una forma conveniente de suministrar y administrar el ciclo de vida de cada página.

Observaciones

Una cosa importante a tener en cuenta sobre el uso de ViewPager es que hay dos versiones diferentes de `FragmentPagerAdapter` y `FragmentStatePagerAdapter` .

Si usa `android.app.Fragment` Fragmentos nativos con FragmentPagerAdapter o FragmentStatePagerAdapter, debe usar las versiones de la biblioteca de soporte del adaptador v13, es decir, `android.support.v13.app.FragmentStatePagerAdapter` .

Si está utilizando `android.support.v4.app.Fragment` support library Fragments con un FragmentPagerAdapter o FragmentStatePagerAdapter, debe usar las versiones de la biblioteca de soporte v4 del adaptador, es decir, `android.support.v4.app.FragmentStatePagerAdapter` .

Examples

Uso básico de ViewPager con fragmentos.

Un `ViewPager` permite mostrar múltiples fragmentos en una actividad que se puede navegar girando hacia la izquierda o hacia la derecha. Un `ViewPager` debe ser alimentado de Vistas o Fragmentos usando un `PagerAdapter` .

Sin embargo, existen otras dos implementaciones específicas que le resultarán más útiles en caso de utilizar Fragments que son `FragmentPagerAdapter` y `FragmentStatePagerAdapter` . Cuando se debe crear una instancia de un Fragmento por primera vez, se `getItem(position)` para cada posición que se necesite instanciar. El método `getCount()` devolverá el número total de páginas para que `ViewPager` sepa cuántos Fragmentos deben mostrarse.

Tanto `FragmentPagerAdapter` como `FragmentStatePagerAdapter` mantienen un caché de los fragmentos que el `ViewPager` deberá mostrar. Por defecto, el `ViewPager` intentará almacenar un máximo de 3 Fragmentos que corresponden al Fragmento actualmente visible, y los que están al lado de la derecha y la izquierda. Además, `FragmentStatePagerAdapter` mantendrá el estado de cada uno de sus fragmentos.

Tenga en cuenta que ambas implementaciones asumen que sus fragmentos mantendrán sus posiciones, por lo que si mantiene una lista de los fragmentos en lugar de tener un número estático de ellos, como puede ver en el método `getItem()` , deberá crear una subclase de `PagerAdapter` y anular al menos los métodos `instantiateItem()` , `destroyItem()` y `getItemPosition()` .

Solo agregue un ViewPager en su diseño como se describe en el ejemplo básico :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>
    <android.support.v4.view.ViewPager
        android:id="@+id/vpPager">
    </android.support.v4.view.ViewPager>
</LinearLayout>
```

Luego, defina el adaptador que determinará cuántas páginas existen y qué fragmento mostrar para cada página del adaptador.

```
public class MyViewPagerActivity extends AppCompatActivity {
    private static final String TAG = MyViewPagerActivity.class.getName();

    private MyPagerAdapter mFragmentAdapter;
    private ViewPager mViewPager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.myActivityLayout);

        //Apply the Adapter
        mFragmentAdapter = new MyPagerAdapter(getSupportFragmentManager());
        mViewPager = (ViewPager) findViewById(R.id.view_pager);
        mViewPager.setAdapter(mFragmentAdapter);
    }

    private class MyPagerAdapter extends FragmentPagerAdapter{

        public MyPagerAdapter(FragmentManager supportFragmentManager) {
            super(supportFragmentManager);
        }

        // Returns the fragment to display for that page
        @Override
        public Fragment getItem(int position) {
            switch(position) {
                case 0:
                    return new Fragment1();

                case 1:
                    return new Fragment2();

                case 2:
                    return new Fragment3();

                default:
                    return null;
            }
        }

        // Returns total number of pages
        @Override
        public int getCount() {
            return 3;
        }
```

```
    }
}
```

3.2.x

Si está utilizando `android.app.Fragment` , debe agregar esta dependencia:

```
compile 'com.android.support:support-v13:25.3.1'
```

Si está utilizando `android.support.v4.app.Fragment` , debe agregar esta dependencia:

```
compile 'com.android.support:support-fragment:25.3.1'
```

ViewPager con TabLayout

Se puede utilizar un `TabLayout` para una navegación más fácil.
Puede configurar las pestañas para cada fragmento en su adaptador usando el método `TabLayout.newTab()` pero hay otro método más conveniente y más fácil para esta tarea que es `TabLayout.setupWithViewPager()` .

Este método se sincronizará creando y eliminando pestañas de acuerdo con el contenido del adaptador asociado con su `ViewPager` cada vez que lo llame.
Además, establecerá una devolución de llamada por lo que cada vez que el usuario pase la página, se seleccionará la pestaña correspondiente.

Solo define un diseño

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>

    <android.support.design.widget.TabLayout
        android:id="@+id/tabs"
        app:tabMode="scrollable" />

    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager"
        android:layout_width="match_parent"
        android:layout_height="0px"
        android:layout_weight="1" />

</LinearLayout>
```

Luego implemente el `FragmentPagerAdapter` y aplíquelo al `ViewPager` :

```
public class MyViewPagerActivity extends AppCompatActivity {
    private static final String TAG = MyViewPagerActivity.class.getName();

    private MyPagerAdapter mFragmentAdapter;
    private ViewPager mViewPager;
    private TabLayout mTabLayout;

    @Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.myActivityLayout);

    // Get the ViewPager and apply the PagerAdapter
    mFragmentAdapter = new MyPagerAdapter(getSupportFragmentManager());
    mViewPager = (ViewPager) findViewById(R.id.view_pager);
    mViewPager.setAdapter(mFragmentAdapter);

    // link the tabLayout and the viewpager together
    mTabLayout = (TabLayout) findViewById(R.id.tab_layout);
    mTabLayout.setupWithViewPager(mViewPager);
}

private class MyPagerAdapter extends FragmentPagerAdapter{

    public MyPagerAdapter(FragmentManager supportFragmentManager) {
        super(supportFragmentManager);
    }

     // Returns the fragment to display for that page
    @Override
    public Fragment getItem(int position) {
        switch(position) {
            case 0:
                return new Fragment1();

            case 1:
                return new Fragment2();

            case 2:
                return new Fragment3();

            default:
                return null;
        }
    }

    // Will be displayed as the tab's label
    @Override
    public CharSequence getPageTitle(int position) {
        switch(position) {
            case 0:
                return "Fragment 1 title";

            case 1:
                return "Fragment 2 title";

            case 2:
                return "Fragment 3 title";

            default:
                return null;
        }
    }

    // Returns total number of pages
    @Override
    public int getCount() {
        return 3;
    }
```

```
    }
}
```

ViewPager con PreferenceFragment

Hasta hace poco, el uso de `android.support.v4.app.FragmentPagerAdapter` impediría el uso de `PreferenceFragment` como uno de los fragmentos utilizados en FragmentPagerAdapter.

Las últimas versiones de la biblioteca de soporte v7 ahora incluyen la clase `PreferenceFragmentCompat` , que funcionará con un ViewPager y la versión v4 de FragmentPagerAdapter.

Fragmento de ejemplo que extiende `PreferenceFragmentCompat` :

```
import android.os.Bundle;
import android.support.v7.preference.PreferenceFragmentCompat;
import android.view.View;

public class MySettingsPrefFragment extends PreferenceFragmentCompat {

    public MySettingsPrefFragment() {
        // Required empty public constructor
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.fragment_settings_pref);
    }

    @Override
    public void onCreatePreferences(Bundle bundle, String s) {

    }
}
```

Ahora puede usar este fragmento en una subclase `android.support.v4.app.FragmentPagerAdapter` :

```
private class PagerAdapterWithSettings extends FragmentPagerAdapter {

    public PagerAdapterWithSettings(FragmentManager supportFragmentManager) {
        super(supportFragmentManager);
    }

    @Override
    public Fragment getItem(int position) {
        switch(position) {
            case 0:
                return new FragmentOne();

            case 1:
                return new FragmentTwo();

            case 2:
                return new MySettingsPrefFragment();
```

```
            default:
                return null;
        }
    }

    // .......

}
```

Agregar un ViewPager

Asegúrese de que la siguiente dependencia se agregue al archivo `build.gradle` su aplicación en las dependencias:

```
compile 'com.android.support:support-core-ui:25.3.0'
```

Luego agregue el `ViewPager` a su diseño de actividad:

```
<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
```

Luego define tu `PagerAdapter` :

```
public class MyPagerAdapter extends PagerAdapter {

    private Context mContext;

    public CustomPagerAdapter(Context context) {
        mContext = context;
    }

    @Override
    public Object instantiateItem(ViewGroup collection, int position) {

        // Create the page for the given position. For example:
        LayoutInflater inflater = LayoutInflater.from(mContext);
        ViewGroup layout = (ViewGroup) inflater.inflate(R.layout.xxxx, collection, false);
        collection.addView(layout);
        return layout;
    }

    @Override
    public void destroyItem(ViewGroup collection, int position, Object view) {
        // Remove a page for the given position. For example:
        collection.removeView((View) view);
    }

    @Override
    public int getCount() {
        //Return the number of views available.
        return numberOfPages;
    }
```

```
    @Override
    public boolean isViewFromObject(View view, Object object) {
        // Determines whether a page View is associated with a specific key object
        // as returned by instantiateItem(ViewGroup, int). For example:
        return view == object;
    }
}
```

Finalmente configure el `ViewPager` en su actividad:

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);
        viewPager.setAdapter(new MyPagerAdapter(this));
    }
}
```

ViewPager con un indicador de puntos



Todo lo que necesitamos es: ViewPager , TabLayout y 2 dibujables para puntos seleccionados y predeterminados.

En primer lugar, debemos agregar `TabLayout` a nuestro diseño de pantalla y conectarlo con `ViewPager` . Podemos hacer esto de dos maneras:

TabLayout anidado en ViewPager

```
<android.support.v4.view.ViewPager
    android:id="@+id/photos_viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.design.widget.TabLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</android.support.v4.view.ViewPager>
```

> En este caso, `TabLayout` se conectará automáticamente con `ViewPager` , pero `TabLayout` estará al lado de `ViewPager` , no sobre él.

TabLayout separado

```
<android.support.v4.view.ViewPager
    android:id="@+id/photos_viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>

<android.support.design.widget.TabLayout
    android:id="@+id/tab_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

> En este caso, podemos poner `TabLayout` cualquier lugar, pero tenemos que conectar `TabLayout` con `ViewPager` programáticamente

```
ViewPager pager = (ViewPager) view.findViewById(R.id.photos_viewpager);
PagerAdapter adapter = new PhotosAdapter(getChildFragmentManager(), photosUrl);
pager.setAdapter(adapter);

TabLayout tabLayout = (TabLayout) view.findViewById(R.id.tab_layout);
tabLayout.setupWithViewPager(pager, true);
```

Una vez que creamos nuestro diseño, tenemos que preparar nuestros puntos. Entonces creamos tres archivos: `selected_dot.xml` , `default_dot.xml` y `tab_selector.xml` .

---

selected_dot.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape
            android:innerRadius="0dp"
            android:shape="ring"
            android:thickness="8dp"
            android:useLevel="false">
            <solid android:color="@color/colorAccent"/>
        </shape>
    </item>
```

```
</layer-list>
```

---

default_dot.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape
            android:innerRadius="0dp"
            android:shape="ring"
            android:thickness="8dp"
            android:useLevel="false">
            <solid android:color="@android:color/darker_gray"/>
        </shape>
    </item>
</layer-list>
```

---

tab_selector.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:drawable="@drawable/selected_dot"
          android:state_selected="true"/>

    <item android:drawable="@drawable/default_dot"/>
</selector>
```

Ahora necesitamos agregar solo 3 líneas de código a `TabLayout` en nuestro diseño xml y listo.

```
app:tabBackground="@drawable/tab_selector"
app:tabGravity="center"
app:tabIndicatorHeight="0dp"
```

Configurar OnPageChangeListener

Si necesita escuchar los cambios en la página seleccionada, puede implementar el `ViewPager.OnPageChangeListener` escucha `ViewPager.OnPageChangeListener` en el ViewPager:

```
viewPager.addOnPageChangeListener(new OnPageChangeListener() {

    // This method will be invoked when a new page becomes selected. Animation is not
necessarily complete.
    @Override
    public void onPageSelected(int position) {
         // Your code
    }

    // This method will be invoked when the current page is scrolled, either as part of
    // a programmatically initiated smooth scroll or a user initiated touch scroll.
```

```
    @Override
    public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {
         // Your code
    }

    // Called when the scroll state changes. Useful for discovering when the user begins
    // dragging, when the pager is automatically settling to the current page,
    // or when it is fully stopped/idle.
    @Override
    public void onPageScrollStateChanged(int state) {
         // Your code
    }
});
```

Lea ViewPager en línea: https://riptutorial.com/es/android/topic/692/viewpager

Capítulo 260: Vista de la lista

Introducción

ListView es un grupo de vista que agrupa varios elementos de una fuente de datos como una matriz o base de datos y los muestra en una lista con capacidad de desplazamiento. Los datos se enlazan con listview usando una clase de adaptador.

Observaciones

`ListView` es un grupo de vistas que muestra una lista de elementos desplazables.
Los elementos de la lista se insertan automáticamente en la lista mediante un `Adapter` que extrae el contenido de una fuente, como una matriz o una consulta de base de datos, y convierte el resultado de cada elemento en una vista que se coloca en la lista.

Cuando el contenido de su diseño es dinámico o no está predeterminado, puede utilizar un diseño que subclases `AdapterView` para rellenar el diseño con vistas en tiempo de ejecución. Una subclase de la clase `AdapterView` usa un `Adapter` para enlazar datos a su diseño.

Antes de usar el `ListView` , también debe verificar los ejemplos de `RecyclerView` .

Examples

Filtrado con CursorAdapter

```
// Get the reference to your ListView
ListView listResults = (ListView) findViewById(R.id.listResults);

// Set its adapter
listResults.setAdapter(adapter);

// Enable filtering in ListView
listResults.setTextFilterEnabled(true);

// Prepare your adapter for filtering
adapter.setFilterQueryProvider(new FilterQueryProvider() {
    @Override
    public Cursor runQuery(CharSequence constraint) {

        // in real life, do something more secure than concatenation
        // but it will depend on your schema
        // This is the query that will run on filtering
        String query = "SELECT _ID as _id, name FROM MYTABLE "
                       + "where name like '%" + constraint + "%' "
                       + "ORDER BY NAME ASC";
        return db.rawQuery(query, null);
     }
});
```

Digamos que su consulta se ejecutará cada vez que el usuario `EditText` un `EditText` :

```
    EditText queryText = (EditText) findViewById(R.id.textQuery);
    queryText.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(final CharSequence s, final int start, final int count,
final int after) {

        }

        @Override
        public void onTextChanged(final CharSequence s, final int start, final int before,
final int count) {
            // This is the filter in action
            adapter.getFilter().filter(s.toString());
            // Don't forget to notify the adapter
            adapter.notifyDataSetChanged();
        }

        @Override
        public void afterTextChanged(final Editable s) {

        }
    });
```

ArrayAdapter personalizado

De forma predeterminada, la clase ArrayAdapter crea una vista para cada elemento de la matriz llamando a `toString()` en cada elemento y colocando el contenido en un TextView.

Para crear una vista compleja para cada elemento (por ejemplo, si desea un ImageView para cada elemento de la matriz), extienda la clase ArrayAdapter y anule el método `getView()` para devolver el tipo de vista que desea para cada elemento.

Por ejemplo:

```
public class MyAdapter extends ArrayAdapter<YourClassData>{

    private LayoutInflater inflater;

    public MyAdapter (Context context, List<YourClassData> data){
        super(context, 0, data);
        inflater = LayoutInflater.from(context);
    }

    @Override
    public long getItemId(int position)
    {
        //It is just an example
        YourClassData data = (YourClassData) getItem(position);
        return data.ID;
    }

    @Override
    public View getView(int position, View view, ViewGroup parent)
    {
        ViewHolder viewHolder;
```

```
        if (view == null) {
            view = inflater.inflate(R.layout.custom_row_layout_design, null);
            // Do some initialization

            //Retrieve the view on the item layout and set the value.
            viewHolder = new ViewHolder(view);
            view.setTag(viewHolder);
         }
         else {
             viewHolder = (ViewHolder) view.getTag();
         }

        //Retrieve your object
        YourClassData data = (YourClassData) getItem(position);

        viewHolder.txt.setTypeface(m_Font);
        viewHolder.txt.setText(data.text);
        viewHolder.img.setImageBitmap(BitmapFactory.decodeFile(data.imageAddr));

        return view;

    }

    private class ViewHolder
    {
         private final TextView txt;
         private final ImageView img;

         private ViewHolder(View view)
         {
             txt = (TextView) view.findViewById(R.id.txt);
             img = (ImageView) view.findViewById(R.id.img);
         }
    }
}
```

Un ListView básico con un ArrayAdapter

De forma predeterminada, `ArrayAdapter` crea una vista para cada elemento de la matriz llamando a `toString()` en cada elemento y colocando el contenido en un `TextView` .

Ejemplo:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, myStringArray);
```

donde `android.R.layout.simple_list_item_1` es el diseño que contiene un `TextView` para cada cadena en la matriz.

Luego simplemente llame a `setAdapter()` en su `ListView` :

```
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

Para usar algo que no sea TextViews para la visualización de matriz, por ejemplo, ImageViews, o

para que algunos de los datos, además de `toString()` , llenen las vistas, anule `getView(int, View, ViewGroup)` para devolver el tipo de vista que desea. Mira este ejemplo .

Lea Vista de la lista en línea: https://riptutorial.com/es/android/topic/4226/vista-de-la-lista

Capítulo 261: Vista de texto

Introducción

Todo lo relacionado con la personalización de TextView en Android SDK.

Sintaxis

- TextView (contexto de contexto)
- (TextView) findViewById (int id)
- void setText (int resid)
- void setText (CharSequence text) // Puedes usar String como argumento

Observaciones

Intenta usarlo en diseño xml o programáticamente.

Examples

Textview con diferentes tamaños de textos

Puede archivar diferentes tamaños de texto dentro de una vista de texto con un intervalo

```
TextView textView = (TextView) findViewById(R.id.textView);
Spannable span = new SpannableString(textView.getText());
span.setSpan(new RelativeSizeSpan(0.8f), start, end, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
textView.setText(span)
```

Personalización de TextView

```
public class CustomTextView extends TextView {

    private float strokeWidth;
    private Integer strokeColor;
    private Paint.Join strokeJoin;
    private float strokeMiter;


    public CustomTextView(Context context) {
        super(context);
        init(null);
    }

    public CustomTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(attrs);
    }
```

```
    public CustomTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init(attrs);
    }

    public void init(AttributeSet attrs) {

        if (attrs != null) {
            TypedArray a = getContext().obtainStyledAttributes(attrs,
R.styleable.CustomTextView);

            if (a.hasValue(R.styleable.CustomTextView_strokeColor)) {
                float strokeWidth =
a.getDimensionPixelSize(R.styleable.CustomTextView_strokeWidth, 1);
                int strokeColor = a.getColor(R.styleable.CustomTextView_strokeColor,
0xff000000);
                float strokeMiter =
a.getDimensionPixelSize(R.styleable.CustomTextView_strokeMiter, 10);
                Paint.Join strokeJoin = null;
                switch (a.getInt(R.styleable.CustomTextView_strokeJoinStyle, 0)) {
                    case (0):
                        strokeJoin = Paint.Join.MITER;
                        break;
                    case (1):
                        strokeJoin = Paint.Join.BEVEL;
                        break;
                    case (2):
                        strokeJoin = Paint.Join.ROUND;
                        break;
                }
                this.setStroke(strokeWidth, strokeColor, strokeJoin, strokeMiter);
            }
        }
    }

    public void setStroke(float width, int color, Paint.Join join, float miter) {
        strokeWidth = width;
        strokeColor = color;
        strokeJoin = join;
        strokeMiter = miter;
    }

    @Override
    public void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        int restoreColor = this.getCurrentTextColor();
        if (strokeColor != null) {
            TextPaint paint = this.getPaint();
            paint.setStyle(Paint.Style.STROKE);
            paint.setStrokeJoin(strokeJoin);
            paint.setStrokeMiter(strokeMiter);
            this.setTextColor(strokeColor);
            paint.setStrokeWidth(strokeWidth);
            super.onDraw(canvas);
            paint.setStyle(Paint.Style.FILL);
            this.setTextColor(restoreColor);
        }
    }
}
```

Uso:

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        CustomTextView customTextView = (CustomTextView) findViewById(R.id.pager_title);
    }
}
```

Diseño:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@mipmap/background">


    <pk.sohail.gallerytest.activity.CustomTextView
        android:id="@+id/pager_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:gravity="center"
        android:text="@string/txt_title_photo_gallery"
        android:textColor="@color/white"
        android:textSize="30dp"
        android:textStyle="bold"
        app:outerShadowRadius="10dp"
        app:strokeColor="@color/title_text_color"
        app:strokeJoinStyle="miter"
        app:strokeWidth="2dp" />

</RelativeLayout>
```

attars:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <declare-styleable name="CustomTextView">

        <attr name="outerShadowRadius" format="dimension" />
        <attr name="strokeWidth" format="dimension" />
        <attr name="strokeMiter" format="dimension" />
        <attr name="strokeColor" format="color" />
        <attr name="strokeJoinStyle">
            <enum name="miter" value="0" />
            <enum name="bevel" value="1" />
            <enum name="round" value="2" />
        </attr>
    </declare-styleable>
```

```
</resources>
```

Uso programático:

```
CustomTextView mtxt_name = (CustomTextView) findViewById(R.id.pager_title);
//then use
setStroke(float width, int color, Paint.Join join, float miter);
//method before setting
setText("Sample Text");
```

TextView de Spannable

Se puede usar un TextView `TextView` en Android para resaltar una parte particular del texto con un color, estilo, tamaño y / o evento de clic diferente en un solo widget de `TextView` .

Considere que ha definido un `TextView` siguiente manera:

```
TextView textview=findViewById(R.id.textview);
```

Luego puede aplicar diferentes resaltados como se muestra a continuación:

- **De color Spannable:** Con el fin de establecer un color diferente a alguna porción de texto, un `ForegroundColorSpan` se puede utilizar, como se muestra en el siguiente ejemplo:

  ```
  Spannable spannable = new SpannableString(firstWord+lastWord);
  spannable.setSpan(new ForegroundColorSpan(firstWordColor), 0, firstWord.length(),
  Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
  spannable.setSpan(new ForegroundColorSpan(lastWordColor), firstWord.length(),
  firstWord.length()+lastWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
  textview.setText( spannable );
  ```

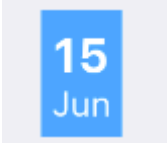
  Salida creada por el código anterior:

- **Fuente distribuible:** para establecer un tamaño de fuente diferente para una parte del texto, se puede usar un `RelativeSizeSpan` , como se muestra en el siguiente ejemplo:

  ```
  Spannable spannable = new SpannableString(firstWord+lastWord);
  spannable.setSpan(new RelativeSizeSpan(1.1f),0, firstWord.length(),
  Spannable.SPAN_EXCLUSIVE_EXCLUSIVE); // set size
  spannable.setSpan(new RelativeSizeSpan(0.8f), firstWord.length(), firstWord.length() +
  lastWord.length(),  Spannable.SPAN_EXCLUSIVE_EXCLUSIVE); // set size
  textview.setText( spannable );
  ```

  Salida creada por el código anterior:



- **Tipo de letra distribuible:** para configurar un tipo de letra diferente para una parte del texto, se puede usar un `TypefaceSpan` personalizado, como se muestra en el siguiente ejemplo:

```
Spannable spannable = new SpannableString(firstWord+lastWord);
spannable.setSpan( new CustomTypefaceSpan("SFUIText-Bold.otf",fontBold), 0,
firstWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
spannable.setSpan( new CustomTypefaceSpan("SFUIText-Regular.otf",fontRegular),
firstWord.length(), firstWord.length() + lastWord.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
text.setText( spannable );
```

Sin embargo, para que el código anterior funcione, la clase `CustomTypefaceSpan` debe derivarse de la clase `TypefaceSpan` . Esto puede hacerse de la siguiente manera:

```
public class CustomTypefaceSpan extends TypefaceSpan {
    private final Typeface newType;

    public CustomTypefaceSpan(String family, Typeface type) {
        super(family);
        newType = type;
    }

    @Override
    public void updateDrawState(TextPaint ds) {
        applyCustomTypeFace(ds, newType);
    }

    @Override
    public void updateMeasureState(TextPaint paint) {
        applyCustomTypeFace(paint, newType);
    }

    private static void applyCustomTypeFace(Paint paint, Typeface tf) {
        int oldStyle;
        Typeface old = paint.getTypeface();
        if (old == null) {
            oldStyle = 0;
        } else {
            oldStyle = old.getStyle();
        }
        int fake = oldStyle & ~tf.getStyle();
        if ((fake & Typeface.BOLD) != 0) {
            paint.setFakeBoldText(true);
        }

        if ((fake & Typeface.ITALIC) != 0) {
            paint.setTextSkewX(-0.25f);
        }

        paint.setTypeface(tf);
    }
}
```

TextView con imagen

Android permite a los programadores colocar imágenes en las cuatro esquinas de un `TextView` . Por ejemplo, si está creando un campo con un `TextView` y al mismo tiempo quiere mostrar que el campo es editable, los desarrolladores generalmente colocarán un icono de edición cerca de ese campo. Android nos ofrece una opción interesante llamada **compuesto** `TextView` para un `TextView` :

```
<TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:drawablePadding="4dp"
        android:drawableRight="@drawable/edit"
        android:text="Hello world"
        android:textSize="18dp" />
```

Puede configurar el dibujo en cualquier lado de su `TextView` siguiente manera:

```
android:drawableLeft="@drawable/edit"
android:drawableRight="@drawable/edit"
android:drawableTop="@drawable/edit"
android:drawableBottom="@drawable/edit"
```

La configuración del dibujable también se puede lograr mediante programación de la siguiente manera:

```
yourTextView.setCompoundDrawables(leftDrawable, rightDrawable, topDrawable, bottomDrawable);
```

La configuración de cualquiera de los parámetros entregados a `setCompoundDrawables()` en `null` eliminará el icono del lado correspondiente de `TextView` .

Strikethrough TextView

Tachar todo el texto.

```
String sampleText = "This is a test strike";
textView.setPaintFlags(tv.getPaintFlags()| Paint.STRIKE_THRU_TEXT_FLAG);
textView.setText(sampleText);
```

Salida: ~~Esta es una huelga de prueba~~

Tachar solo partes del texto

```
String sampleText = "This is a test strike";
```

```
SpannableStringBuilder spanBuilder = new SpannableStringBuilder(sampleText);
StrikethroughSpan strikethroughSpan = new StrikethroughSpan();
spanBuilder.setSpan(
        strikethroughSpan, // Span to add
        0, // Start
        4, // End of the span (exclusive)
        Spanned.SPAN_EXCLUSIVE_EXCLUSIVE // Text changes will not reflect in the strike
changing
);
textView.setText(spanBuilder);
```

Salida: ~~Esta~~ es una huelga de prueba

Personalización de temas y estilos.

MainActivity.java:

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:custom="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <com.customthemeattributedemo.customview.CustomTextView
        style="?mediumTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="@string/message_hello"
        custom:font_family="@string/bold_font" />

    <com.customthemeattributedemo.customview.CustomTextView
        style="?largeTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="@string/message_hello"
        custom:font_family="@string/bold_font" />
</LinearLayout>
```

CustomTextView.java:

```
public class CustomTextView extends TextView {

    private static final String TAG = "TextViewPlus";
    private Context mContext;

    public CustomTextView(Context context) {
        super(context);
        mContext = context;
    }

    public CustomTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
        mContext = context;
        setCustomFont(context, attrs);
    }

    public CustomTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        mContext = context;
        setCustomFont(context, attrs);
    }

    private void setCustomFont(Context ctx, AttributeSet attrs) {
        TypedArray customFontNameTypedArray = ctx.obtainStyledAttributes(attrs,
R.styleable.CustomTextView);
        String customFont =
customFontNameTypedArray.getString(R.styleable.CustomTextView_font_family);
        Typeface typeface = null;
        typeface = Typeface.createFromAsset(ctx.getAssets(), customFont);
        setTypeface(typeface);
        customFontNameTypedArray.recycle();
    }
}
```

attrs.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <attr name="mediumTextStyle" format="reference" />
    <attr name="largeTextStyle" format="reference" />

    <declare-styleable name="CustomTextView">

        <attr name="font_family" format="string" />
        <!--- Your other attributes -->

    </declare-styleable>
</resources>
```

strings.xml:

```
<resources>
    <string name="app_name">Custom Style Theme Attribute Demo</string>
    <string name="message_hello">Hello Hiren!</string>

    <string name="bold_font">bold.ttf</string>
</resources>
```

styles.xml:

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>

        <item name="mediumTextStyle">@style/textMedium</item>
        <item name="largeTextStyle">@style/textLarge</item>
    </style>


    <style name="textMedium" parent="textParentStyle">
        <item name="android:textAppearance">@android:style/TextAppearance.Medium</item>
    </style>

    <style name="textLarge" parent="textParentStyle">
        <item name="android:textAppearance">@android:style/TextAppearance.Large</item>
    </style>

    <style name="textParentStyle">
        <item name="android:textColor">@android:color/white</item>
        <item name="android:background">@color/colorPrimary</item>
        <item name="android:padding">5dp</item>
    </style>

</resources>
```

Hacer que RelativeSizeSpan se alinee hacia arriba

Para hacer que un `RelativeSizeSpan` alinee con la parte superior, se puede derivar una clase personalizada de la clase `SuperscriptSpan` . En el siguiente ejemplo, la clase derivada se llama `TopAlignSuperscriptSpan` :

activity_main.xml:

```
<TextView
    android:id="@+id/txtView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:textSize="26sp" />
```

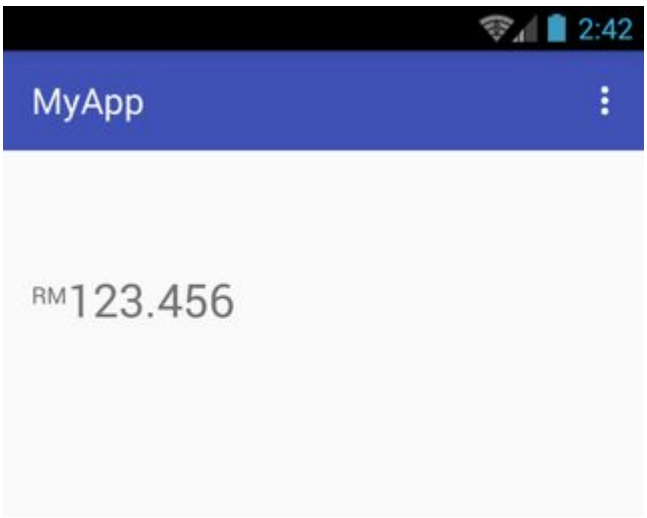
MainActivity.java:

```
TextView txtView = (TextView) findViewById(R.id.txtView);

SpannableString spannableString = new SpannableString("RM123.456");
spannableString.setSpan( new TopAlignSuperscriptSpan( (float)0.35 ), 0, 2,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE );
txtView.setText(spannableString);
```

TopAlignSuperscriptSpan.java:

```
private class TopAlignSuperscriptSpan extends SuperscriptSpan {
    //divide superscript by this number
    protected int fontScale = 2;

    //shift value, 0 to 1.0
    protected float shiftPercentage = 0;

    //doesn't shift
    TopAlignSuperscriptSpan() {}

    //sets the shift percentage
    TopAlignSuperscriptSpan( float shiftPercentage ) {
        if( shiftPercentage > 0.0 && shiftPercentage < 1.0 )
            this.shiftPercentage = shiftPercentage;
    }

    @Override
    public void updateDrawState( TextPaint tp ) {
        //original ascent
        float ascent = tp.ascent();

        //scale down the font
        tp.setTextSize( tp.getTextSize() / fontScale );

        //get the new font ascent
        float newAscent = tp.getFontMetrics().ascent;

        //move baseline to top of old font, then move down size of new font
        //adjust for errors with shift percentage
        tp.baselineShift += ( ascent - ascent * shiftPercentage )
                - (newAscent - newAscent * shiftPercentage );
    }

    @Override
    public void updateMeasureState( TextPaint tp ) {
        updateDrawState( tp );
    }
}
```

Captura de pantalla de referencia:



Pinchzoom en TextView

activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/mytv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="This is my sample text for pinch zoom demo, you can zoom in and out
using pinch zoom, thanks" />

</RelativeLayout>
```

MainActivity.java :

```
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.TextView;

public class MyTextViewPinchZoomClass extends Activity implements OnTouchListener {

    final static float STEP = 200;
    TextView mytv;
    float mRatio = 1.0f;
    int mBaseDist;
    float mBaseRatio;
    float fontsize = 13;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mytv = (TextView) findViewById(R.id.mytv);
        mytv.setTextSize(mRatio + 13);
    }

    public boolean onTouchEvent(MotionEvent event) {
        if (event.getPointerCount() == 2) {
            int action = event.getAction();
            int pureaction = action & MotionEvent.ACTION_MASK;
            if (pureaction == MotionEvent.ACTION_POINTER_DOWN) {
                mBaseDist = getDistance(event);
                mBaseRatio = mRatio;
            } else {
                float delta = (getDistance(event) - mBaseDist) / STEP;
                float multi = (float) Math.pow(2, delta);
                mRatio = Math.min(1024.0f, Math.max(0.1f, mBaseRatio * multi));
```

```
                mytv.setTextSize(mRatio + 13);
            }
        }
        return true;
    }

    int getDistance(MotionEvent event) {
        int dx = (int) (event.getX(0) - event.getX(1));
        int dy = (int) (event.getY(0) - event.getY(1));
        return (int) (Math.sqrt(dx * dx + dy * dy));
    }

    public boolean onTouch(View v, MotionEvent event) {
        return false;
    }
}
```

TextView único con dos colores diferentes

El texto coloreado se puede crear pasando el texto y un nombre de color de fuente a la siguiente función:

```
private String getColoredSpanned(String text, String color) {
    String input = "<font color=" + color + ">" + text + "</font>";
    return input;
}
```

El texto coloreado se puede configurar en `TextView` (o incluso en `Button` , `EditText` , etc.) usando el código de ejemplo a continuación.

Primero, define un `TextView` siguiente manera:

```
TextView txtView = (TextView)findViewById(R.id.txtView);
```

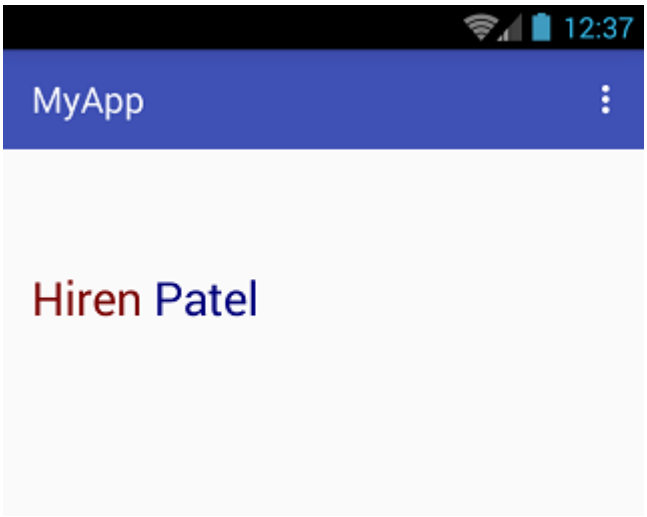
Luego, crea un texto de diferente color y asignalo a cadenas:

```
String name = getColoredSpanned("Hiren", "#800000");
String surName = getColoredSpanned("Patel","#000080");
```

Finalmente, establezca las dos cadenas de colores diferentes en `TextView` :

```
txtView.setText(Html.fromHtml(name+" "+surName));
```

Captura de pantalla de referencia:



Lea Vista de texto en línea: https://riptutorial.com/es/android/topic/4212/vista-de-texto

Capítulo 262: Vista inferior de la navegación

Introducción

La Vista de navegación inferior ha estado en las pautas de diseño del material durante algún tiempo, pero no ha sido fácil para nosotros implementarlo en nuestras aplicaciones.

Algunas aplicaciones han creado sus propias soluciones, mientras que otras se han basado en bibliotecas de código abierto de terceros para realizar el trabajo.

Ahora que la biblioteca de soporte de diseño está viendo la adición de esta barra de navegación inferior, ¡analicemos cómo podemos usarla!

Observaciones

Representa una barra de navegación inferior estándar para la aplicación. Es una implementación de material de diseño de navegación inferior.

Campo de golf:

- Javadoc oficial

Examples

Implementacion basica

Para agregar el `BottomNavigationView` siga estos pasos:

1. Agregue en su `build.gradle` la **dependencia** :

```
compile 'com.android.support:design:25.1.0'
```

2. Agregue el `BottomNavigationView` en **su diseño** :

```
<android.support.design.widget.BottomNavigationView
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:id="@+id/bottom_navigation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:menu="@menu/bottom_navigation_menu"/>
```

3. Crea el **menú** para rellenar la vista:

```
<!-- res/menu/bottom_navigation_menu.xml -->
```

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/my_action1"
        android:enabled="true"
        android:icon="@drawable/my_drawable"
        android:title="@string/text"
        app:showAsAction="ifRoom" />
    ....
</menu>
```

4. Adjuntar **un oyente** para los eventos de clic:

```
//Get the view
BottomNavigationView bottomNavigationView = (BottomNavigationView)
                findViewById(R.id.bottom_navigation);
//Attach the listener
bottomNavigationView.setOnNavigationItemSelectedListener(
        new BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                switch (item.getItemId()) {

                    case R.id.my_action1:
                        //Do something...
                        break;

                    //...
                }
                return true;//returning false disables the Navigation bar animations
            }
        });
```

Ver código de demostración en BottomNavigation-Demo

Personalización de BottomNavigationView

Nota: Supongo que usted sabe cómo utilizar ***`BottomNavigationView`*** *.*

En este ejemplo explicaré cómo agregar el selector para **`BottomNavigationView`** . Por lo que puede indicar en la interfaz de usuario para los iconos y textos.

Cree `bottom_navigation_view_selector.xml` como

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item  android:color="@color/bottom_nv_menu_selected" android:state_checked="true" />
    <item android:color="@color/bottom_nv_menu_default" />
</selector>
```

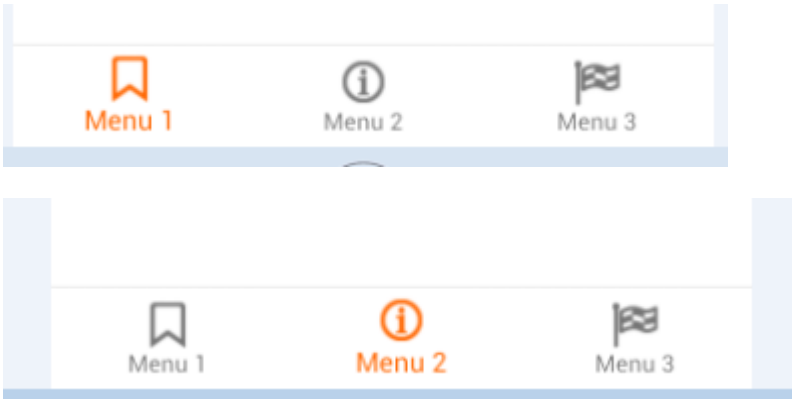
Y use los atributos siguientes en **`BottomNavigationView`** en el archivo de diseño

```
app:itemIconTint="@drawable/bottom_navigation_view_selector"
```

```
app:itemTextColor="@drawable/bottom_navigation_view_selector"
```

> En el ejemplo anterior, he usado el mismo selector `bottom_navigation_view_selector` para `app:itemIconTint` y `app:itemTextColor` para mantener los colores de texto e íconos iguales. Pero si su diseño tiene un color diferente para el texto y el ícono, puede definir 2 selectores diferentes y usarlos.

La salida será similar a la de abajo



Manejo de estados habilitados / deshabilitados

Crear selector para habilitar / deshabilitar elemento de menú.

selector.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="@color/white" android:state_enabled="true" />
    <item android:color="@color/colorPrimaryDark" android:state_enabled="false" />
</selector>
```

design.xml

```
<android.support.design.widget.BottomNavigationView
    android:id="@+id/bottom_navigation"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    app:itemBackground="@color/colorPrimary"
    app:itemIconTint="@drawable/nav_item_color_state"
    app:itemTextColor="@drawable/nav_item_color_state"
    app:menu="@menu/bottom_navigation_main" />
```

Permitiendo más de 3 menús.

Este ejemplo es estrictamente una solución alternativa, ya que actualmente no hay forma de deshabilitar un comportamiento conocido como ShiftMode.

Crear una función como tal.

```
public static void disableMenuShiftMode(BottomNavigationView view) {
    BottomNavigationMenuView menuView = (BottomNavigationMenuView) view.getChildAt(0);
    try {
        Field shiftingMode = menuView.getClass().getDeclaredField("mShiftingMode");
        shiftingMode.setAccessible(true);
        shiftingMode.setBoolean(menuView, false);
        shiftingMode.setAccessible(false);
        for (int i = 0; i < menuView.getChildCount(); i++) {
            BottomNavigationItemView item = (BottomNavigationItemView) menuView.getChildAt(i);
            //noinspection RestrictedApi
            item.setShiftingMode(false);
            // set once again checked value, so view will be updated
            //noinspection RestrictedApi
            item.setChecked(item.getItemData().isChecked());
        }
    } catch (NoSuchFieldException e) {
        Log.e("BNVHelper", "Unable to get shift mode field", e);
    } catch (IllegalAccessException e) {
        Log.e("BNVHelper", "Unable to change value of shift mode", e);
    }
}
```

Esto desactiva el comportamiento de desplazamiento del menú cuando el recuento de elementos supera los 3 números.

USO

```
BottomNavigationView navView = (BottomNavigationView)
findViewById(R.id.bottom_navigation_bar);
disableMenuShiftMode(navView);
```

Problema de progreso : Agregue la siguiente línea de archivo de configuración de progreso, de lo contrario, esto no funcionaría.

```
-keepclassmembers class android.support.design.internal.BottomNavigationMenuView {
boolean mShiftingMode;
}
```

Alternativamente, puede crear una Clase y acceder a este método desde allí. Vea la respuesta original aquí

NOTA : Este es un **HOTFIX** basado en Reflection, actualícelo una vez que la biblioteca de asistencia de Google se actualice con una llamada de función directa.

Lea Vista inferior de la navegación en línea: https://riptutorial.com/es/android/topic/7565/vista-inferior-de-la-navegacion

Capítulo 263: Visualización de anuncios de Google

Examples

Configuración básica de anuncios

Deberá agregar lo siguiente a sus dependencias:

```
compile 'com.google.firebase:firebase-ads:10.2.1'
```

y luego poner esto en el mismo archivo.

```
apply plugin: 'com.google.gms.google-services'
```

A continuación, deberá agregar información relevante a su archivo strings.xml.

```
<string name="banner_ad_unit_id">ca-app-pub-####/####</string>
```

A continuación, coloque una vista donde la desee y aplíquela como cualquier otra vista.

```
    <com.google.android.gms.ads.AdView
        android:id="@+id/adView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        ads:adSize="BANNER"
        ads:adUnitId="@string/banner_ad_unit_id">
    </com.google.android.gms.ads.AdView>
```

Y por último, pero no menos importante, lanza esto en tu onCreate.

```
MobileAds.initialize(getApplicationContext(), "ca-app-pub-YOUR_ID");
AdView mAdView = (AdView) findViewById(R.id.adView);
    AdRequest adRequest = new AdRequest.Builder().build();
    mAdView.loadAd(adRequest);
```

Si copió y pegó exactamente, ahora debería tener un pequeño banner publicitario. Simplemente coloque más AdViews donde los necesite para obtener más.

Añadiendo anuncio intersticial

Los anuncios intersticiales son anuncios a pantalla completa que cubren la interfaz de su aplicación host. Normalmente se muestran en los puntos de transición natural en el flujo de una aplicación, como entre las actividades o durante la pausa entre niveles en un juego.

Asegúrese de tener los permisos necesarios en su archivo de `Manifest` :

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

1. Ve a tu cuenta de AdMob .
2. Haga clic en la pestaña **Monetizar** .
3. Selecciona o Crea la aplicación y elige la plataforma.
4. Seleccione Intersticial y asigne un nombre de bloque de anuncios.
5. Una vez que se crea el bloque de anuncios, puede observar la ID del bloque de anuncios en el panel de control. Por ejemplo: ca-app-pub-00000000000/000000000
6. Añadir dependencias

```
compile 'com.google.firebase:firebase-ads:10.2.1'
```

Este debe estar en la parte inferior.

```
apply plugin: 'com.google.gms.google-services'
```

Agregue su **ID** de `strings.xml` **anuncios** a su archivo `strings.xml`

```
<string name="interstitial_full_screen">ca-app-pub-00000000/00000000</string>
```

Agregue ConfigChanges y metadatos a su manifiesto:

```
<activity
            android:name="com.google.android.gms.ads.AdActivity"

android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScree

            android:theme="@android:style/Theme.Translucent" />
```

y

```
<meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
```

Actividad:

```
public class AdActivity extends AppCompatActivity {

    private String TAG = AdActivity.class.getSimpleName();
    InterstitialAd mInterstitialAd;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_second);

        mInterstitialAd = new InterstitialAd(this);

        // set the ad unit ID
        mInterstitialAd.setAdUnitId(getString(R.string.interstitial_full_screen));

        AdRequest adRequest = new AdRequest.Builder()
                .build();

        // Load ads into Interstitial Ads
        mInterstitialAd.loadAd(adRequest);

        mInterstitialAd.setAdListener(new AdListener() {
            public void onAdLoaded() {
                showInterstitial();
            }
        });
    }

    private void showInterstitial() {
        if (mInterstitialAd.isLoaded()) {
            mInterstitialAd.show();
        }
    }

}
```

Este AdActivity mostrará un anuncio de pantalla completa ahora.

Lea Visualización de anuncios de Google en línea:
https://riptutorial.com/es/android/topic/5984/visualizacion-de-anuncios-de-google

Capítulo 264: Voleo

Introducción

Volley es una biblioteca HTTP de Android que fue introducida por Google para hacer que las llamadas de red sean mucho más simples. Por defecto, todas las llamadas de la red de Volley se realizan de forma asíncrona, manejando todo en un hilo de fondo y devolviendo los resultados en primer plano con el uso de devoluciones de llamada. Como la obtención de datos a través de una red es una de las tareas más comunes que se realizan en cualquier aplicación, la biblioteca Volley se creó para facilitar el desarrollo de aplicaciones para Android.

Sintaxis

- RequestQueue queue = Volley.newRequestQueue (contexto); // configurar la cola
- Request request = new SomeKindOfRequestClass (Request.Method, String url, Response.Listener, Response.ErrorListener); // configura algún tipo de solicitud, el tipo exacto y los argumentos cambian para cada tipo de solicitud
- queue.add (solicitud); // agregar la solicitud a la cola; se llamará al oyente de respuesta apropiado una vez que la solicitud haya finalizado (o finalizado por cualquier motivo)

Observaciones

Instalación

Puedes construir Volley desde el código fuente oficial de Google . Por un tiempo, esa fue la única opción. O usando una de las versiones pre-construidas de terceros. Sin embargo, Google finalmente lanzó un paquete oficial de maven en jcenter.

En su archivo `build.gradle` nivel de `build.gradle` , agregue esto a su lista de dependencias:

```
dependencies {
    ...
    compile 'com.android.volley:volley:1.0.0'
}
```

Asegúrese de que el permiso de `INTERNET` esté configurado en el manifiesto de su aplicación:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Documentacion oficial

Google no ha proporcionado una documentación muy extensa sobre esta biblioteca, y no la han

tocado en años. Pero lo que está disponible se puede encontrar en:

https://developer.android.com/training/volley/index.html

Hay documentación no oficial alojada en GitHub, aunque debería haber una mejor ubicación para alojar esto en el futuro:

https://pablobaxter.github.io/volley-docs/

Examples

StringRequest básico utilizando el método GET

```
final TextView mTextView = (TextView) findViewById(R.id.text);
...

// Instantiate the RequestQueue.
RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://www.google.com";

// Request a string response from the provided URL.
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
            new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        // Display the first 500 characters of the response string.
        mTextView.setText("Response is: "+ response.substring(0,500));
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        mTextView.setText("That didn't work!");
    }
});
// Add the request to the RequestQueue.
queue.add(stringRequest);
```

Cancelar una solicitud

```
// assume a Request and RequestQueue have already been initialized somewhere above

public static final String TAG = "SomeTag";

// Set the tag on the request.
request.setTag(TAG);

// Add the request to the RequestQueue.
mRequestQueue.add(request);

// To cancel this specific request
request.cancel();

// ... then, in some future life cycle event, for example in onStop()
// To cancel all requests with the specified tag in RequestQueue
mRequestQueue.cancelAll(TAG);
```

Agregar atributos de tiempo de diseño personalizados a NetworkImageView

Hay varios atributos adicionales que el Volley `NetworkImageView` agrega al `ImageView` estándar. Sin embargo, estos atributos solo se pueden establecer en código. El siguiente es un ejemplo de cómo hacer una clase de extensión que recogerá los atributos de su archivo de diseño XML y los aplicará a la instancia de `NetworkImageView` por usted.

En su directorio `~/res/xml` , agregue un archivo llamado `attrx.xml` :

```
<resources>
    <declare-styleable name="MoreNetworkImageView">
        <attr name="defaultImageResId" format="reference"/>
        <attr name="errorImageResId" format="reference"/>
    </declare-styleable>
</resources>
```

Agrega un nuevo archivo de clase a tu proyecto:

```
package my.namespace;

import android.content.Context;
import android.content.res.TypedArray;
import android.support.annotation.NonNull;
import android.util.AttributeSet;

import com.android.volley.toolbox.NetworkImageView;

public class MoreNetworkImageView extends NetworkImageView {
    public MoreNetworkImageView(@NonNull final Context context) {
        super(context);
    }

    public MoreNetworkImageView(@NonNull final Context context, @NonNull final AttributeSet
attrs) {
        this(context, attrs, 0);
    }

    public MoreNetworkImageView(@NonNull final Context context, @NonNull final AttributeSet
attrs, final int defStyle) {
        super(context, attrs, defStyle);

        final TypedArray attributes = context.obtainStyledAttributes(attrs,
R.styleable.MoreNetworkImageView, defStyle, 0);

        // load defaultImageResId from XML
        int defaultImageResId =
attributes.getResourceId(R.styleable.MoreNetworkImageView_defaultImageResId, 0);
        if (defaultImageResId > 0) {
            setDefaultImageResId(defaultImageResId);
        }

        // load errorImageResId from XML
        int errorImageResId =
attributes.getResourceId(R.styleable.MoreNetworkImageView_errorImageResId, 0);
        if (errorImageResId > 0) {
            setErrorImageResId(errorImageResId);
        }
```

```
    }
}
```

Un archivo de diseño de ejemplo que muestra el uso de los atributos personalizados:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="wrap_content"
  android:layout_height="fill_parent">

  <my.namespace.MoreNetworkImageView
    android:layout_width="64dp"
    android:layout_height="64dp"
    app:errorImageResId="@drawable/error_img"
    app:defaultImageResId="@drawable/default_img"
    tools:defaultImageResId="@drawable/editor_only_default_img"/>
    <!--
      Note: The "tools:" prefix does NOT work for custom attributes in Android Studio 2.1 and
      older at least, so in this example the defaultImageResId would show "default_img" in the

      editor, not the "editor_only_default_img" drawable even though it should if it was
      supported as an editor-only override correctly like standard Android properties.
    -->

</android.support.v7.widget.CardView>
```

Solicita JSON

```
final TextView mTxtDisplay = (TextView) findViewById(R.id.txtDisplay);
ImageView mImageView;
String url = "http://ip.jsontest.com/";

final JsonObjectRequest jsObjRequest = new JsonObjectRequest
        (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {
    @Override
    public void onResponse(JSONObject response) {
        mTxtDisplay.setText("Response: " + response.toString());
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        // ...
    }
});

requestQueue.add(jsObjRequest);
```

Agregar encabezados personalizados a sus solicitudes [por ejemplo, para autenticación básica]

Si necesita agregar encabezados personalizados a sus solicitudes de volea, no puede hacer esto después de la inicialización, ya que los encabezados se guardan en una variable privada.

En su lugar, debe anular el método `getHeaders()` de `Request.class` como tal:

```
new JsonObjectRequest(REQUEST_METHOD, REQUEST_URL, REQUEST_BODY, RESP_LISTENER, ERR_LISTENER)
{
    @Override
    public Map<String, String> getHeaders() throws AuthFailureError {
        HashMap<String, String> customHeaders = new Hashmap<>();

        customHeaders.put("KEY_0", "VALUE_0");
        ...
        customHeaders.put("KEY_N", "VALUE_N");

        return customHeaders;
    }
};
```

Explicación de los parámetros:

- `REQUEST_METHOD` : cualquiera de las constantes `Request.Method.*` .
- `REQUEST_URL` : la URL completa para enviar su solicitud.
- `REQUEST_BODY` : un objeto `JSONObject` contiene el cuerpo POST que se enviará (o nulo).
- `RESP_LISTENER` : un objeto `Response.Listener<?>` , Cuyo `onResponse(T data)` cuando se completa con éxito.
- `ERR_LISTENER` : un objeto `Response.ErrorListener` , cuyo `onErrorResponse(VolleyError e)` se llama a una solicitud fallida.

Si desea crear una solicitud personalizada, también puede agregar los encabezados en ella:

```
public class MyCustomRequest extends Request {
    ...
    @Override
    public Map<String, String> getHeaders() throws AuthFailureError {
        HashMap<String, String> customHeaders = new Hashmap<>();

        customHeaders.put("KEY_0", "VALUE_0");
        ...
        customHeaders.put("KEY_N", "VALUE_N");

        return customHeaders;
    }
    ...
}
```

Clase de ayuda para manejar los errores de volea

```
public class VolleyErrorHelper {
        /**
         * Returns appropriate message which is to be displayed to the user
         * against the specified error object.
         *
         * @param error
         * @param context
         * @return
         */
```

```
        public static String getMessage (Object error , Context context){
            if(error instanceof TimeoutError){
                return context.getResources().getString(R.string.timeout);
            }else if (isServerProblem(error)){
                return handleServerError(error ,context);

            }else if(isNetworkProblem(error)){
                return context.getResources().getString(R.string.nointernet);
            }
            return context.getResources().getString(R.string.generic_error);

        }

        private static String handleServerError(Object error, Context context) {

            VolleyError er = (VolleyError)error;
            NetworkResponse response = er.networkResponse;
            if(response != null){
                switch (response.statusCode){

                    case 404:
                    case 422:
                    case 401:
                        try {
                            // server might return error like this { "error": "Some error
occured" }
                            // Use "Gson" to parse the result
                            HashMap<String, String> result = new Gson().fromJson(new
String(response.data),
                                    new TypeToken<Map<String, String>>() {
                                    }.getType());

                            if (result != null && result.containsKey("error")) {
                                return result.get("error");
                            }

                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                        // invalid request
                        return ((VolleyError) error).getMessage();

                    default:
                        return context.getResources().getString(R.string.timeout);
                }
            }

            return context.getResources().getString(R.string.generic_error);
        }

        private static boolean isServerProblem(Object error) {
            return (error instanceof ServerError || error instanceof AuthFailureError);
        }

        private static boolean isNetworkProblem (Object error){
            return (error instanceof NetworkError || error instanceof NoConnectionError);
        }
```

Autenticación del servidor remoto usando StringRequest a través del método POST

Por el bien de este ejemplo, supongamos que tenemos un servidor para manejar las solicitudes POST que haremos desde nuestra aplicación de Android:

```
// User input data.
String email = "my@email.com";
String password = "123";

// Our server URL for handling POST requests.
String URL = "http://my.server.com/login.php";

// When we create a StringRequest (or a JSONRequest) for sending
// data with Volley, we specify the Request Method as POST, and
// the URL that will be receiving our data.
StringRequest stringRequest =
    new StringRequest(Request.Method.POST, URL,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            // At this point, Volley has sent the data to your URL
            // and has a response back from it. I'm going to assume
            // that the server sends an "OK" string.
            if (response.equals("OK")) {
                // Do login stuff.
            } else {
                // So the server didn't return an "OK" response.
                // Depending on what you did to handle errors on your
                // server, you can decide what action to take here.
            }
        }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // This is when errors related to Volley happen.
            // It's up to you what to do if that should happen, but
            // it's usually not a good idea to be too clear as to
            // what happened here to your users.
        }
    }) {
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            // Here is where we tell Volley what it should send in
            // our POST request. For this example, we want to send
            // both the email and the password.

            // We will need key ids for our data, so our server can know
            // what is what.
            String key_email = "email";
            String key_password = "password";

            Map<String, String> map = new HashMap<String, String>();
            // map.put(key, value);
            map.put(key_email, email);
            map.put(key_password, password);
            return map;
        }
    };

    // This is a policy that we need to specify to tell Volley, what
    // to do if it gets a timeout, how many times to retry, etc.
```

```
    stringRequest.setRetryPolicy(new RetryPolicy() {
            @Override
            public int getCurrentTimeout() {
                // Here goes the timeout.
                // The number is in milliseconds, 5000 is usually enough,
                // but you can up or low that number to fit your needs.
                return 50000;
            }
            @Override
            public int getCurrentRetryCount() {
                // The maximum number of attempts.
                // Again, the number can be anything you need.
                return 50000;
            }
            @Override
            public void retry(VolleyError error) throws VolleyError {
                // Here you could check if the retry count has gotten
                // to the maximum number, and if so, send a VolleyError
                // message or similar. For the sake of the example, I'll
                // show a Toast.
                Toast.makeText(getContext(), error.toString(), Toast.LENGTH_LONG).show();
            }
    });

    // And finally, we create a Volley Queue. For this example, I'm using
    // getContext(), because I was working with a Fragment. But context could
    // be "this", "getContext()", etc.
    RequestQueue requestQueue = Volley.newRequestQueue(getContext());
    requestQueue.add(stringRequest);

} else {
    // If, for example, the user inputs an email that is not currently
    // on your remote DB, here's where we can inform the user.
    Toast.makeText(getContext(), "Wrong email", Toast.LENGTH_LONG).show();
}
```

Usando Volley para peticiones HTTP

Agregue la dependencia de Gradle en build.gradle a nivel de aplicación

```
compile 'com.android.volley:volley:1.0.0'
```

Además, agregue el permiso android.permission.INTERNET al manifiesto de su aplicación.

** Crear instancia de Volley RequestQueue singleton en su aplicación **

```
public class InitApplication extends Application {

private RequestQueue queue;
private static InitApplication sInstance;

private static final String TAG = InitApplication.class.getSimpleName();


@Override
public void onCreate() {
    super.onCreate();
```

```
    sInstance = this;

    Stetho.initializeWithDefaults(this);

}

public static synchronized InitApplication getInstance() {
    return sInstance;
}

public <T> void addToQueue(Request<T> req, String tag) {
    req.setTag(TextUtils.isEmpty(tag) ? TAG : tag);
    getQueue().add(req);
}

public <T> void addToQueue(Request<T> req) {
    req.setTag(TAG);
    getQueue().add(req);
}

public void cancelPendingRequests(Object tag) {
    if (queue != null) {
        queue.cancelAll(tag);
    }
}

public RequestQueue getQueue() {
    if (queue == null) {
        queue = Volley.newRequestQueue(getApplicationContext());
        return queue;
    }
    return queue;
}
}
```

Ahora, puede usar la instancia de volley usando el método getInstance () y agregar una nueva solicitud en la cola usando `InitApplication.getInstance().addToQueue(request);`

Un ejemplo simple para solicitar JsonObject desde el servidor es

```
JsonObjectRequest myRequest = new JsonObjectRequest(Method.GET,
        url, null,
        new Response.Listener<JSONObject>() {

            @Override
            public void onResponse(JSONObject response) {
                Log.d(TAG, response.toString());
            }
        }, new Response.ErrorListener() {

            @Override
            public void onErrorResponse(VolleyError error) {
                Log.d(TAG, "Error: " + error.getMessage());
            }
});

myRequest.setRetryPolicy(new DefaultRetryPolicy(
        MY_SOCKET_TIMEOUT_MS,
        DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
```

```
        DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
```

Para manejar los tiempos de espera de Volley, debes usar una `RetryPolicy` . Se usa una política de reintento en caso de que una solicitud no pueda completarse debido a una falla de la red o en otros casos.

Volley proporciona una manera fácil de implementar su `RetryPolicy` para sus solicitudes. De forma predeterminada, Volley establece todos los tiempos de espera de conexión y socket en 5 segundos para todas las solicitudes. `RetryPolicy` es una interfaz en la que necesita implementar su lógica de cómo desea reintentar una solicitud en particular cuando se produce un tiempo de espera.

El constructor toma los siguientes tres parámetros:

- `initialTimeoutMs` : especifica el tiempo de espera del socket en milisegundos para cada intento de reintento.
- `maxNumRetries` : el número de veces que se intenta reintentar.
- `backoffMultiplier` : un multiplicador que se utiliza para determinar el tiempo exponencial establecido en socket para cada intento de reintento.

Respuesta variable booleana del servidor con solicitud json en volea

puedes personalizar la clase debajo de una

```
private final String PROTOCOL_CONTENT_TYPE = String.format("application/json; charset=%s",
PROTOCOL_CHARSET);

    public BooleanRequest(int method, String url, String requestBody,
Response.Listener<Boolean> listener, Response.ErrorListener errorListener) {
        super(method, url, errorListener);
        this.mListener = listener;
        this.mErrorListener = errorListener;
        this.mRequestBody = requestBody;
    }

    @Override
    protected Response<Boolean> parseNetworkResponse(NetworkResponse response) {
        Boolean parsed;
        try {
            parsed = Boolean.valueOf(new String(response.data,
HttpHeaderParser.parseCharset(response.headers)));
        } catch (UnsupportedEncodingException e) {
            parsed = Boolean.valueOf(new String(response.data));
        }
        return Response.success(parsed, HttpHeaderParser.parseCacheHeaders(response));
    }

    @Override
    protected VolleyError parseNetworkError(VolleyError volleyError) {
        return super.parseNetworkError(volleyError);
    }

    @Override
    protected void deliverResponse(Boolean response) {
```

```
        mListener.onResponse(response);
    }

    @Override
    public void deliverError(VolleyError error) {
        mErrorListener.onErrorResponse(error);
    }

    @Override
    public String getBodyContentType() {
        return PROTOCOL_CONTENT_TYPE;
    }

    @Override
    public byte[] getBody() throws AuthFailureError {
        try {
            return mRequestBody == null ? null : mRequestBody.getBytes(PROTOCOL_CHARSET);
        } catch (UnsupportedEncodingException uee) {
            VolleyLog.wtf("Unsupported Encoding while trying to get the bytes of %s using %s",
                    mRequestBody, PROTOCOL_CHARSET);
            return null;
        }
    }
}
```

usa esto con tu actividad

```
try {
        JSONObject jsonBody;
        jsonBody = new JSONObject();
        jsonBody.put("Title", "Android Demo");
        jsonBody.put("Author", "BNK");
        jsonBody.put("Date", "2015/08/28");
        String requestBody = jsonBody.toString();
        BooleanRequest booleanRequest = new BooleanRequest(0, url, requestBody, new
Response.Listener<Boolean>() {
            @Override
            public void onResponse(Boolean response) {
                Toast.makeText(mContext, String.valueOf(response), Toast.LENGTH_SHORT).show();
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Toast.makeText(mContext, error.toString(), Toast.LENGTH_SHORT).show();
            }
        });
        // Add the request to the RequestQueue.
        queue.add(booleanRequest);
    } catch (JSONException e) {
        e.printStackTrace();
    }
```

Usa JSONArray como cuerpo de solicitud

Las solicitudes predeterminadas integradas en volley no permiten pasar un `JSONArray` como cuerpo de solicitud en una solicitud `POST` . En su lugar, solo puede pasar un objeto `JSON` como parámetro.

Sin embargo, en lugar de pasar un `JSON` objeto como un parámetro para la solicitud constructor, es necesario anular el `getBody()` método de la `Request.class` . Debes pasar `null` como tercer parámetro también:

```
JSONArray requestBody = new JSONArray();

new JsonObjectRequest(Request.Method.POST, REQUEST_URL, null, RESP_LISTENER, ERR_LISTENER) {
    @Override
    public byte[] getBody() {
        try {
            return requestBody.toString().getBytes(PROTOCOL_CHARSET);
        } catch (UnsupportedEncodingException uee) {
            // error handling
            return null;
        }
    }
};
```

Explicación de los parámetros:

- `REQUEST_URL` : la URL completa para enviar su solicitud.
- `RESP_LISTENER` : un objeto `Response.Listener<?>` , Cuyo `onResponse(T data)` cuando se completa con éxito.
- `ERR_LISTENER` : un objeto `Response.ErrorListener` , cuyo `onErrorResponse(VolleyError e)` se llama a una solicitud fallida.

Lea Voleo en línea: https://riptutorial.com/es/android/topic/2800/voleo

Capítulo 265: WebView

Introducción

WebView es una vista que muestra páginas web dentro de su aplicación. Por esto puedes agregar tu propia URL.

Observaciones

Por favor, no olvide agregar permiso en su archivo de manifiesto de Android

```
<uses-permission android:name="android.permission.INTERNET" />
```

Examples

Los diálogos de alerta de JavaScript en WebView - Cómo hacer que funcionen

De forma predeterminada, WebView no implementa diálogos de alerta de JavaScript, es decir. `alert()` no hará nada. Para que sea necesario, primero debe habilitar JavaScript (obviamente ...), y luego configurar un `WebChromeClient` para manejar las solicitudes de diálogos de alerta desde la página:

```
webView.setWebChromeClient(new WebChromeClient() {
    //Other methods for your WebChromeClient here, if needed..

    @Override
    public boolean onJsAlert(WebView view, String url, String message, JsResult result) {
        return super.onJsAlert(view, url, message, result);
    }
});
```

Aquí, `onJsAlert` , y luego llamamos a la súper implementación, que nos da un diálogo estándar de Android. También puede usar el mensaje y la URL usted mismo, por ejemplo, si desea crear un diálogo de estilo personalizado o si desea iniciar sesión.

Comunicación de Javascript a Java (Android)

Actividad de Android

```
package com.example.myapp;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class WebViewActivity extends Activity {
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        WebView webView = new WebView(this);
        setContentView(webView);

       /*
        *   Note the label Android, this is used in the Javascript side of things
        *   You can of course change this.
        */
        webView.addJavascriptInterface(new JavascriptHandler(), "Android");

        webView.loadUrl("http://example.com");
    }
}
```

Java Javascript Handler

```
import android.webkit.JavascriptInterface;

public class JavascriptHandler {

    /**
     *  Key point here is the annotation @JavascriptInterface
     *
     */
    @JavascriptInterface
    public void jsCallback() {
       // Do something
    }

    @JavascriptInterface
    public void jsCallbackTwo(String dummyData) {
       // Do something
    }
}
```

Página web, llamada Javascript

```
<script>
...
Android.jsCallback();
...
Android.jsCallback('hello test');
...
</script>
```

Consejo Extra

Al pasar en una estructura de datos compleja, una posible solución es usar JSON.

```
Android.jsCallback('{ "fake-var" : "fake-value", "fake-array" : [0,1,2] }');
```

En el lado de Android use su analizador JSON favorito, es decir: JSONObject

Comunicación de Java a Javascript

Ejemplo básico

```
package com.example.myapp;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class WebViewActivity extends Activity {

    private Webview webView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);

       webView = new WebView(this);
       webView.getSettings().setJavaScriptEnabled(true);

       setContentView(webView);

       webView.loadUrl("http://example.com");

       /*
        * Invoke Javascript function
        */
       webView.loadUrl("javascript:testJsFunction('Hello World!')");
    }

   /**
    * Invoking a Javascript function
    */
   public void doSomething() {
     this.webView.loadUrl("javascript:testAnotherFunction('Hello World Again!')");
   }
}
```

Ejemplo de marcador abierto

Si la página web a contiene un número de teléfono, puede hacer una llamada utilizando el marcador de su teléfono. Este código comprueba la url que comienza con tel: luego intente abrir el marcador y puede hacer una llamada al número de teléfono seleccionado:

```
public boolean shouldOverrideUrlLoading(WebView view, String url) {
    if (url.startsWith("tel:")) {
            Intent intent = new Intent(Intent.ACTION_DIAL,
                    Uri.parse(url));
            startActivity(intent);
    }else if(url.startsWith("http:") || url.startsWith("https:")) {
        view.loadUrl(url);
    }
    return true;
}
```

Solución de problemas de WebView mediante la impresión de los mensajes de la consola o la depuración remota

Imprimiendo mensajes de consola webview a logcat

Para manejar los mensajes de la `console` desde la página web, puede anular `onConsoleMessage` en `WebChromeClient` :

```
final class ChromeClient extends WebChromeClient {
    @Override
    public boolean onConsoleMessage(ConsoleMessage msg) {
        Log.d(
            "WebView",
            String.format("%s %s:%d", msg.message(), msg.lineNumber(), msg.sourceId())
        );
        return true;
    }
}
```

Y ponlo en tu actividad o fragmento:

```
webView.setWebChromeClient(new ChromeClient());
```

Así que esta página de muestra:

```
<html>
<head>
    <script type="text/javascript">
        console.log('test message');
    </script>
</head>
<body>
</body>
</html>
```

escribirá el registro 'mensaje de prueba' a logcat:

> WebView: muestra de mensaje de prueba.html: 4

`console.info()` , `console.warn()` y `console.error()` también son compatibles con chrome-client.

Depuración remota de dispositivos Android con Chrome

Tu puedes depurar de forma remota la aplicación basada en webview desde tu escritorio Chrome.

Habilitar la depuración USB en su dispositivo Android

En su dispositivo Android, abra Configuración, busque la sección de Opciones para desarrolladores y habilite la depuración USB.

Conecta y descubre tu dispositivo Android

Abra la página en chrome página siguiente: chrome: // inspect / / devices

En el cuadro de diálogo Inspeccionar dispositivos, seleccione su dispositivo y presione **inspeccionar** . Una nueva instancia de DevTools de Chrome se abre en su máquina de desarrollo.

Puede encontrar una guía y descripción más detallada de DevTools en developers.google.com

Abrir archivo local / Crear contenido dinámico en Webview

Layout.xml

```
            <WebView
            android:id="@+id/WebViewToDisplay"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:fadeScrollbars="false" />
```

Cargar datos en WebViewToDisplay

```
        WebView webViewDisplay;
        StringBuffer LoadWEb1;

        webViewDisplay = (WebView) findViewById(R.id.WebViewToDisplay);
        LoadWEb1 = new StringBuffer();
        LoadWEb1.append("<html><body><h1>My First Heading</h1><p>My first paragraph.</p>");
        //Sample code to read parameters at run time
        String strName = "Test Paragraph";
        LoadWEb1.append("<br/><p>"+strName+"</p>");
        String result = LoadWEb1.append("</body></html>").toString();
                WebSettings webSettings = webViewDisplay.getSettings();
                webSettings.setJavaScriptEnabled(true);
                webViewDisplay.getSettings().setBuiltInZoomControls(true);
                if (android.os.Build.VERSION.SDK_INT >= 11){
                    webViewDisplay.setLayerType(View.LAYER_TYPE_SOFTWARE, null);
                    webViewDisplay.getSettings().setDisplayZoomControls(false);
                }

                webViewDisplay.loadDataWithBaseURL(null, result, "text/html", "utf-8",
                        null);
                //To load local file directly from assets folder use below code
                //webViewDisplay.loadUrl("file:///android_asset/aboutapp.html");
```

Lea WebView en línea: https://riptutorial.com/es/android/topic/153/webview

Capítulo 266: Widgets

Observaciones

SDv

Examples

Declaración Manifiesta -

Declare la clase `AppWidgetProvider` en el archivo `AndroidManifest.xml` su aplicación. Por ejemplo:

```
<receiver android:name="ExampleAppWidgetProvider" >
<intent-filter>
    <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
</intent-filter>
<meta-data android:name="android.appwidget.provider"
           android:resource="@xml/example_appwidget_info" />
</receiver>
```

Metadatos

Agregue los metadatos de AppWidgetProviderInfo en `res/xml` :

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="40dp"
    android:minHeight="40dp"
    android:updatePeriodMillis="86400000"
    android:previewImage="@drawable/preview"
    android:initialLayout="@layout/example_appwidget"
    android:configure="com.example.android.ExampleAppWidgetConfigure"
    android:resizeMode="horizontal|vertical"
    android:widgetCategory="home_screen">
</appwidget-provider>
```

Clase AppWidgetProvider

La devolución de llamada de `AppWidgetProvider` más importante es **`onUpdate()`** . Se llama cada vez que se agrega un appwidget.

```
public class ExampleAppWidgetProvider extends AppWidgetProvider {

    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[]
appWidgetIds) {
        final int N = appWidgetIds.length;

        // Perform this loop procedure for each App Widget that belongs to this provider
        for (int i=0; i<N; i++) {
            int appWidgetId = appWidgetIds[i];
```

```
            // Create an Intent to launch ExampleActivity
            Intent intent = new Intent(context, ExampleActivity.class);
            PendingIntent pendingIntent = PendingIntent.getActivity(context, 0, intent, 0);

            // Get the layout for the App Widget and attach an on-click listener
            // to the button
            RemoteViews views = new RemoteViews(context.getPackageName(),
R.layout.appwidget_provider_layout);
            views.setOnClickPendingIntent(R.id.button, pendingIntent);

            // Tell the AppWidgetManager to perform an update on the current app widget
            appWidgetManager.updateAppWidget(appWidgetId, views);
        }
    }
}
```

`onAppWidgetOptionsChanged()` cuando el widget se coloca o cambia de tamaño.

`onDeleted(Context, int[])` se llama cuando se elimina el widget.

Dos widgets con diferentes diseños de declaración.

1. Declarar dos receptores en un archivo de manifiesto:

```
<receiver
    android:name=".UVMateWidget"
    android:label="UVMate Widget 1x1">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>

    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/widget_1x1" />
</receiver>
<receiver
    android:name=".UVMateWidget2x2"
    android:label="UVMate Widget 2x2">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>

    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/widget_2x2" />
</receiver>
```

2. Crear dos diseños
    - `@xml/widget_1x1`
    - `@xml/widget_2x2`
3. Declare la subclase `UVMateWidget2x2` de la clase `UVMateWidget` con comportamiento extendido:

```
package au.com.aershov.uvmate;

import android.content.Context;
import android.widget.RemoteViews;
```

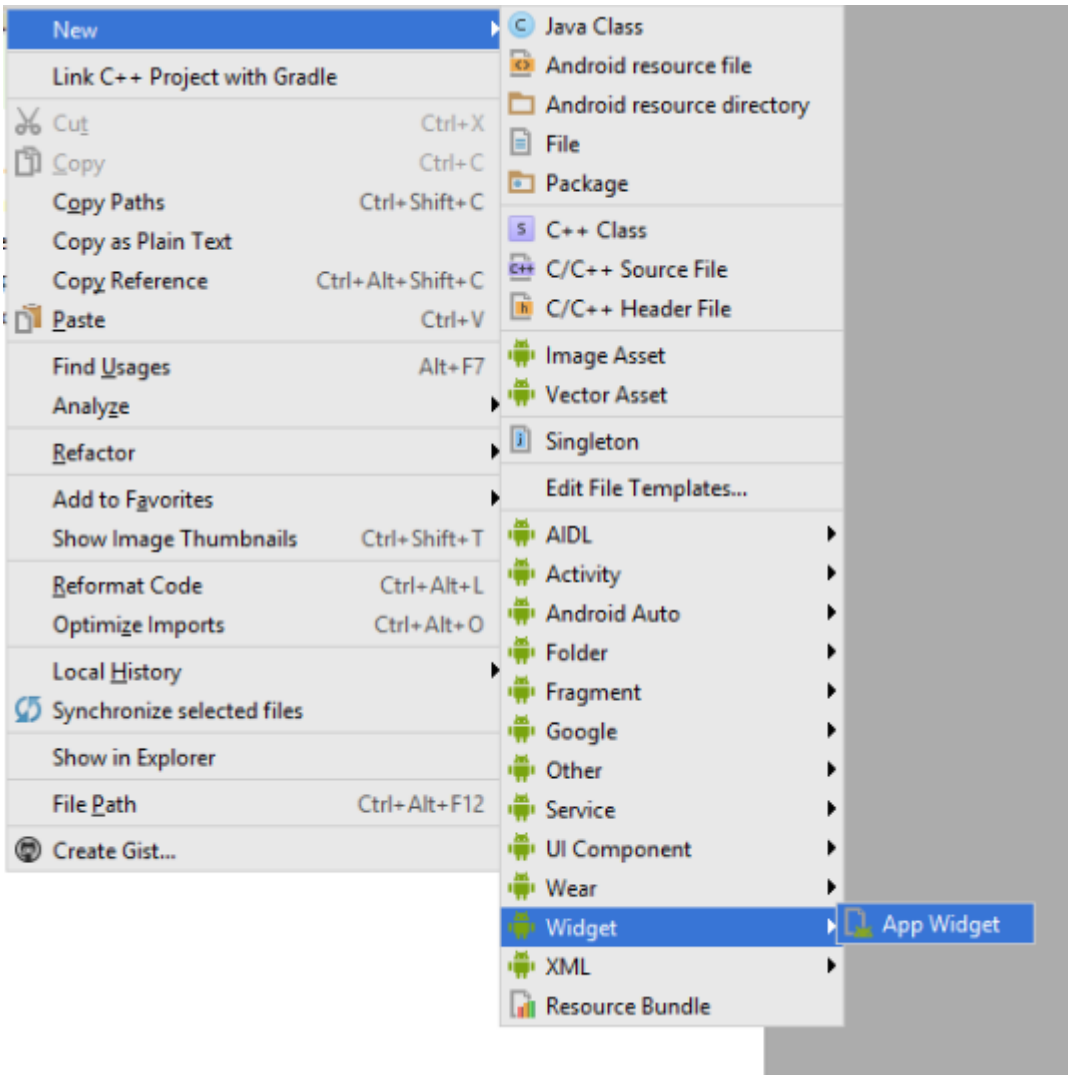
```
public class UVMateWidget2x2 extends UVMateWidget {

    public RemoteViews getRemoteViews(Context context, int minWidth,
                                      int minHeight) {

        mUVMateHelper.saveWidgetSize(mContext.getString(R.string.app_ws_2x2));
        return new RemoteViews(context.getPackageName(), R.layout.widget_2x2);
    }
}
```

Crear / Integrar Widget básico utilizando Android Studio

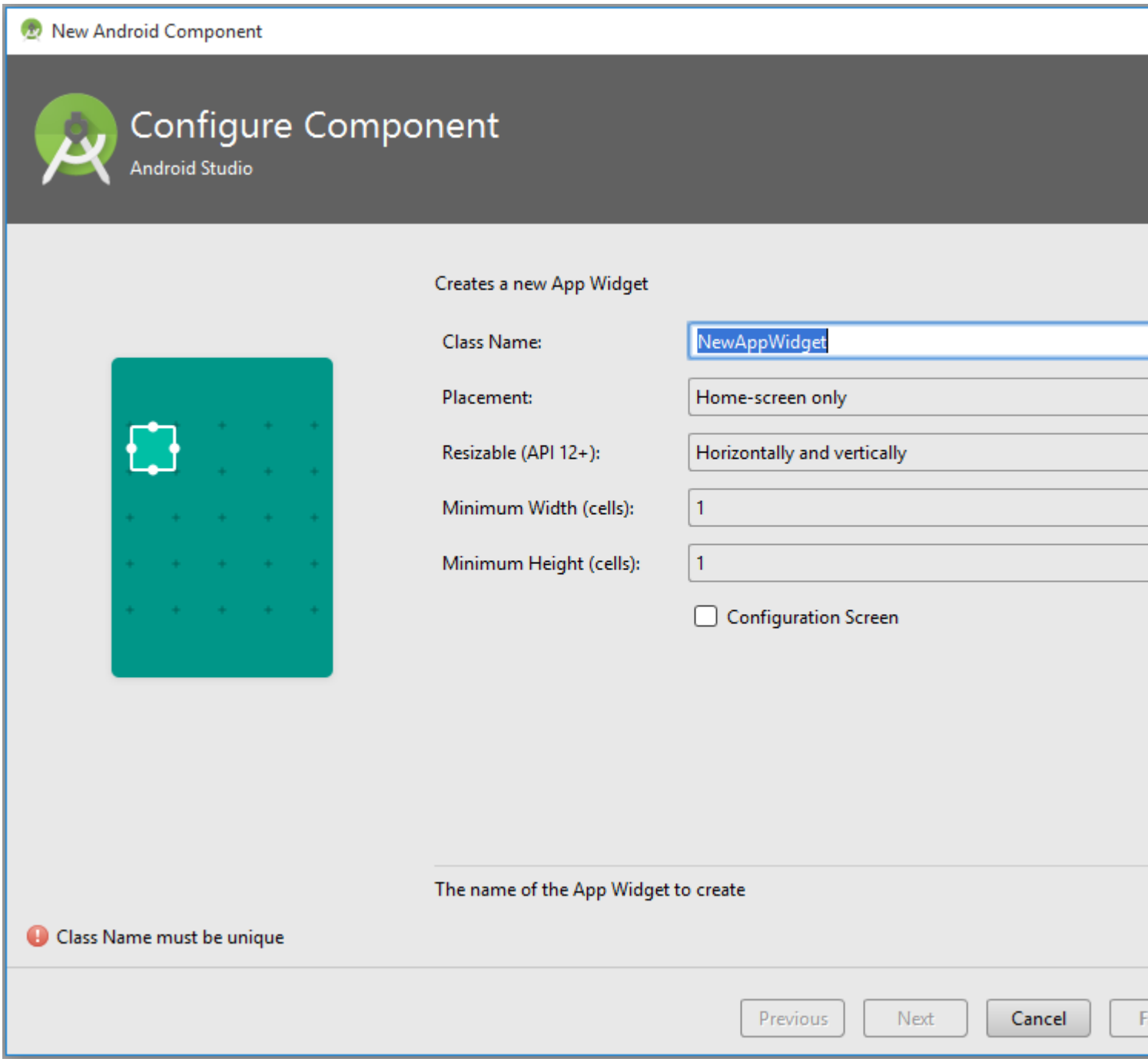
Android Studio creará e integrará un widget básico a su aplicación en 2 pasos.

Justo en su aplicación ==> Nuevo ==> Widget ==> Widget de aplicación

.



Mostrará una pantalla como abajo y llenará los campos.



Está hecho.

Creará e integrará un widget HelloWorld básico (incluido el archivo de diseño, el archivo de metadatos, la declaración en el archivo de manifiesto, etc.) a su aplicación.

Lea Widgets en línea: https://riptutorial.com/es/android/topic/2812/widgets

Capítulo 267: XMPP registro de inicio de sesión y chat simple ejemplo

Examples

Registro XMPP inicio de sesión y ejemplo básico de chat.

Instale Openfire o cualquier servidor de chat en su sistema o en el servidor. Para más detalles haga clic aquí.

Crear proyecto de Android y agregar estas bibliotecas en gradle:

```
compile 'org.igniterealtime.smack:smack-android:4.2.0'
compile 'org.igniterealtime.smack:smack-tcp:4.2.0'
compile 'org.igniterealtime.smack:smack-im:4.2.0'
compile 'org.igniterealtime.smack:smack-android-extensions:4.2.0'
```

A continuación, cree una clase xmpp desde el propósito de conexión xmpp:

```
public class XMPP {

public static final int PORT = 5222;
private static XMPP instance;
private XMPPTCPConnection connection;
private static String TAG = "XMPP-EXAMPLE";
public static final String ACTION_LOGGED_IN = "liveapp.loggedin";
private String HOST = "192.168.0.10";

private XMPPTCPConnectionConfiguration buildConfiguration() throws XmppStringprepException {
    XMPPTCPConnectionConfiguration.Builder builder =
            XMPPTCPConnectionConfiguration.builder();


    builder.setHost(HOST);
    builder.setPort(PORT);
    builder.setCompressionEnabled(false);
    builder.setDebuggerEnabled(true);
    builder.setSecurityMode(ConnectionConfiguration.SecurityMode.disabled);
    builder.setSendPresence(true);

    if (Build.VERSION.SDK_INT >= 14) {
        builder.setKeystoreType("AndroidCAStore");
        // config.setTruststorePassword(null);
        builder.setKeystorePath(null);
    } else {
        builder.setKeystoreType("BKS");
        String str = System.getProperty("javax.net.ssl.trustStore");
        if (str == null) {
            str = System.getProperty("java.home") + File.separator + "etc" + File.separator +
"security"
                    + File.separator + "cacerts.bks";
        }
        builder.setKeystorePath(str);
```

```
    }
    DomainBareJid serviceName = JidCreate.domainBareFrom(HOST);
    builder.setServiceName(serviceName);


    return builder.build();
}

private XMPPTCPConnection getConnection() throws XMPPException, SmackException, IOException,
InterruptedException {
    Log.logDebug(TAG, "Getting XMPP Connect");
    if (isConnected()) {
        Log.logDebug(TAG, "Returning already existing connection");
        return this.connection;
    }

    long l = System.currentTimeMillis();
    try {
        if(this.connection != null){
            Log.logDebug(TAG, "Connection found, trying to connect");
            this.connection.connect();
        }else{
            Log.logDebug(TAG, "No Connection found, trying to create a new connection");
            XMPPTCPConnectionConfiguration config = buildConfiguration();
            SmackConfiguration.DEBUG = true;
            this.connection = new XMPPTCPConnection(config);
            this.connection.connect();
        }
    } catch (Exception e) {
        Log.logError(TAG,"some issue with getting connection :" + e.getMessage());

    }

    Log.logDebug(TAG, "Connection Properties: " + connection.getHost() + " " +
connection.getServiceName());
    Log.logDebug(TAG, "Time taken in first time connect: " + (System.currentTimeMillis() -
l));
    return this.connection;
}

public static XMPP getInstance() {
    if (instance == null) {
        synchronized (XMPP.class) {
            if (instance == null) {
                instance = new XMPP();
            }
        }
    }
    return instance;
}

public void close() {
    Log.logInfo(TAG, "Inside XMPP close method");
    if (this.connection != null) {
        this.connection.disconnect();
    }
}

private XMPPTCPConnection connectAndLogin(Context context) {
    Log.logDebug(TAG, "Inside connect and Login");
    if (!isConnected()) {
```

```
        Log.logDebug(TAG, "Connection not connected, trying to login and connect");
        try {
           // Save username and password then use here
            String username = AppSettings.getUser(context);
            String password = AppSettings.getPassword(context);
            this.connection = getConnection();
            Log.logDebug(TAG, "XMPP username :" + username);
            Log.logDebug(TAG, "XMPP password :" + password);
            this.connection.login(username, password);
            Log.logDebug(TAG, "Connect and Login method, Login successful");
            context.sendBroadcast(new Intent(ACTION_LOGGED_IN));
        } catch (XMPPException localXMPPException) {
            Log.logError(TAG, "Error in Connect and Login Method");
            localXMPPException.printStackTrace();
        } catch (SmackException e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        } catch (IOException e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        } catch (InterruptedException e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        } catch (IllegalArgumentException e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        } catch (Exception e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        }
    }
    Log.logInfo(TAG, "Inside getConnection - Returning connection");
    return this.connection;
}

public boolean isConnected() {
    return (this.connection != null) && (this.connection.isConnected());
}

public EntityFullJid getUser() {
    if (isConnected()) {
        return connection.getUser();
    } else {
        return null;
    }
}

public void login(String user, String pass, String username)
        throws XMPPException, SmackException, IOException, InterruptedException,
PurplKiteXMPPConnectException {
    Log.logInfo(TAG, "inside XMPP getlogin Method");
    long l = System.currentTimeMillis();
    XMPPTCPConnection connect = getConnection();
    if (connect.isAuthenticated()) {
        Log.logInfo(TAG, "User already logged in");
        return;
    }

    Log.logInfo(TAG, "Time taken to connect: " + (System.currentTimeMillis() - l));

    l = System.currentTimeMillis();
```

```
    try{
        connect.login(user, pass);
    }catch (Exception e){
        Log.logError(TAG, "Issue in login, check the stacktrace");
        e.printStackTrace();
    }

    Log.logInfo(TAG, "Time taken to login: " + (System.currentTimeMillis() - l));

    Log.logInfo(TAG, "login step passed");

    PingManager pingManager = PingManager.getInstanceFor(connect);
    pingManager.setPingInterval(5000);

}

public void register(String user, String pass) throws XMPPException,
SmackException.NoResponseException, SmackException.NotConnectedException {
    Log.logInfo(TAG, "inside XMPP register method, " + user + " : " + pass);
    long l = System.currentTimeMillis();
    try {
        AccountManager accountManager = AccountManager.getInstance(getConnection());
        accountManager.sensitiveOperationOverInsecureConnection(true);
        accountManager.createAccount(Localpart.from(user), pass);
    } catch (SmackException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (PurplKiteXMPPConnectException e) {
        e.printStackTrace();
    }
    Log.logInfo(TAG, "Time taken to register: " + (System.currentTimeMillis() - l));
}


public void addStanzaListener(Context context, StanzaListener stanzaListener){
    XMPPTCPConnection connection =  connectAndLogin(context);
    connection.addAsyncStanzaListener(stanzaListener, null);
}

public void removeStanzaListener(Context context, StanzaListener stanzaListener){
    XMPPTCPConnection connection =  connectAndLogin(context);
    connection.removeAsyncStanzaListener(stanzaListener);
}

public void addChatListener(Context context, ChatManagerListener chatManagerListener){
    ChatManager.getInstanceFor(connectAndLogin(context))
            .addChatListener(chatManagerListener);
}

public void removeChatListener(Context context, ChatManagerListener chatManagerListener){

ChatManager.getInstanceFor(connectAndLogin(context)).removeChatListener(chatManagerListener);
}

public void getSrvDeliveryManager(Context context){
    ServiceDiscoveryManager sdm = ServiceDiscoveryManager
            .getInstanceFor(XMPP.getInstance().connectAndLogin(
                    context));
```

```
    //sdm.addFeature("http://jabber.org/protocol/disco#info");
    //sdm.addFeature("jabber:iq:privacy");
    sdm.addFeature("jabber.org/protocol/si");
    sdm.addFeature("http://jabber.org/protocol/si");
    sdm.addFeature("http://jabber.org/protocol/disco#info");
    sdm.addFeature("jabber:iq:privacy");

}

public String getUserLocalPart(Context context){
   return  connectAndLogin(context).getUser().getLocalpart().toString();
}

public EntityFullJid getUser(Context context){
    return  connectAndLogin(context).getUser();
}

public Chat getThreadChat(Context context, String party1, String party2){
    Chat chat = ChatManager.getInstanceFor(
            XMPP.getInstance().connectAndLogin(context))
            .getThreadChat(party1 + "-" + party2);
    return chat;
}

public Chat createChat(Context context, EntityJid jid, String party1, String party2,
ChatMessageListener messageListener){
    Chat chat = ChatManager.getInstanceFor(
            XMPP.getInstance().connectAndLogin(context))
            .createChat(jid, party1 + "-" + party2,
                    messageListener);
    return chat;
}

public void sendPacket(Context context, Stanza packet){
    try {
        connectAndLogin(context).sendStanza(packet);
    } catch (SmackException.NotConnectedException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

}
```

Finalmente, agregue esta actividad:

```
private UserLoginTask mAuthTask = null;
private ChatManagerListener chatListener;
private Chat chat;
private Jid opt_jid;
private ChatMessageListener messageListener;
private StanzaListener packetListener;

private boolean register(final String paramString1,final String paramString2) {
    try {
        XMPP.getInstance().register(paramString1, paramString2);
        return true;

    } catch (XMPPException localXMPPException) {
```

```
        localXMPPException.printStackTrace();
    } catch (SmackException.NoResponseException e) {
        e.printStackTrace();
    } catch (SmackException.NotConnectedException e) {
        e.printStackTrace();
    }
    return false;
}

private boolean login(final String user,final String pass,final String username) {

    try {

        XMPP.getInstance().login(user, pass, username);
        sendBroadcast(new Intent("liveapp.loggedin"));

        return true;
    } catch (Exception e) {
        e.printStackTrace();
        try {

            XMPP.getInstance()
                    .login(user, pass, username);
            sendBroadcast(new Intent("liveapp.loggedin"));

            return true;
        } catch (XMPPException e1) {
            e1.printStackTrace();
        } catch (SmackException e1) {
            e1.printStackTrace();
        } catch (InterruptedException e1) {
            e1.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
        }catch (Exception e1){
            e1.printStackTrace();
        }
    }
    return false;
}

public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {

    public UserLoginTask() {
    }

    protected Boolean doInBackground(Void... paramVarArgs) {
        String mEmail = "abc";
        String mUsername = "abc";
        String mPassword = "welcome";

        if (register(mEmail, mPassword)) {
            try {
                XMPP.getInstance().close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return login(mEmail, mPassword, mUsername);

    }
```

```
    protected void onCancelled() {
        mAuthTask = null;

    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();

    }

    protected void onPostExecute(Boolean success) {
        mAuthTask = null;
        try {
            if (success) {

              messageListener = new ChatMessageListener() {
                @Override
                public void processMessage(Chat chat, Message message) {

                // here you will get only connected user by you

                }
        };


        packetListener = new StanzaListener() {
                @Override
                public void processPacket(Stanza packet) throws
SmackException.NotConnectedException, InterruptedException {

                    if (packet instanceof Message) {
                        final Message message = (Message) packet;

                 // here you will get all messages send by anybody
                    }
                }
        };

        chatListener = new ChatManagerListener() {

                @Override
                public void chatCreated(Chat chatCreated, boolean local) {
                    onChatCreated(chatCreated);
                }
        };


        try {
        String opt_jidStr = "abc";

        try {
            opt_jid = JidCreate.bareFrom(Localpart.from(opt_jidStr), Domainpart.from(HOST));
        } catch (XmppStringprepException e) {
            e.printStackTrace();
        }
        String addr1 = XMPP.getInstance().getUserLocalPart(getActivity());
        String addr2 = opt_jid.toString();
        if (addr1.compareTo(addr2) > 0) {
            String addr3 = addr2;
```

```
            addr2 = addr1;
            addr1 = addr3;
        }
        chat = XMPP.getInstance().getThreadChat(getActivity(), addr1, addr2);
        if (chat == null) {
            chat = XMPP.getInstance().createChat(getActivity(), (EntityJid) opt_jid, addr1,
addr2, messageListener);
            PurplkiteLogs.logInfo(TAG, "chat value single chat 1 :" + chat);
        } else {
            chat.addMessageListener(messageListener);
            PurplkiteLogs.logInfo(TAG, "chat value single chat  2:" + chat);
        }

        } catch (Exception e) {
        e.printStackTrace();
        }


        XMPP.getInstance().addStanzaListener(getActivity(), packetListener);
        XMPP.getInstance().addChatListener(getActivity(), chatListener);
        XMPP.getInstance().getSrvDeliveryManager(getActivity());

            } else {

            }
        } catch (Exception e) {
            e.printStackTrace();
        }

    }
}

/**
 * user attemptLogin for xmpp
 *
 */

private void attemptLogin() {
    if (mAuthTask != null) {
        return;
    }

    boolean cancel = false;
    View focusView = null;

    if (cancel) {
        focusView.requestFocus();
    } else {
        try {
            mAuthTask = new UserLoginTask();
            mAuthTask.execute((Void) null);
        } catch (Exception e) {

        }

    }
}

void onChatCreated(Chat chatCreated) {
    if (chat != null) {
        if (chat.getParticipant().getLocalpart().toString().equals(
```

```
                chatCreated.getParticipant().getLocalpart().toString())) {
            chat.removeMessageListener(messageListener);
            chat = chatCreated;
            chat.addMessageListener(messageListener);
        }
    } else {
        chat = chatCreated;
        chat.addMessageListener(messageListener);
    }
}

private void sendMessage(String message) {
    if (chat != null) {
        try {
            chat.sendMessage(message);
        } catch (SmackException.NotConnectedException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

 @Override
public void onDestroy() {
    // TODO Auto-generated method stub
    super.onDestroy();
    try {
        XMPP.getInstance().removeChatListener(getActivity(), chatListener);
        if (chat != null && messageListener != null) {
            XMPP.getInstance().removeStanzaListener(getActivity(), packetListener);
            chat.removeMessageListener(messageListener);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

}
```

Asegúrese de que el permiso de Internet esté agregado en su archivo de manifiesto.

Lea XMPP registro de inicio de sesión y chat simple ejemplo en línea: https://riptutorial.com/es/android/topic/6747/xmpp-registro-de-inicio-de-sesion-y-chat-simple-ejemplo

Capítulo 268: Xposed

Examples

Creando un Módulo Xposed

Xposed es un marco que te permite conectar llamadas de método de otras aplicaciones. Cuando realiza una modificación al descompilar un APK, puede insertar / cambiar comandos directamente donde lo desee. Sin embargo, tendrá que volver a compilar / firmar el APK después y solo podrá distribuir todo el paquete. Con Xposed, puede inyectar su propio código antes o después de los métodos, o reemplazar los métodos completos por completo. Desafortunadamente, solo puedes instalar Xposed en dispositivos rooteados. Debería usar Xposed siempre que quiera manipular el comportamiento de otras aplicaciones o del sistema Android principal y no quiera pasar por la molestia de descompilar, recompilar y firmar archivos APK.

Primero, creas una aplicación estándar sin una actividad en Android Studio.

Luego tienes que incluir el siguiente código en tu *build.gradle* :

```
repositories {
    jcenter();
}
```

Después de eso agrega las siguientes dependencias:

```
provided 'de.robv.android.xposed:api:82'
provided 'de.robv.android.xposed:api:82:sources'
```

Ahora tiene que colocar estas etiquetas dentro de la etiqueta de la *aplicación que se* encuentra en *AndroidManifest.xml* para que Xposed reconozca su módulo:

```
<meta-data
        android:name="xposedmodule"
        android:value="true" />
<meta-data
        android:name="xposeddescription"
        android:value="YOUR_MODULE_DESCRIPTION" />
<meta-data
        android:name="xposedminversion"
        android:value="82" />
```

NOTA: Reemplace siempre el **82** con la última versión de Xposed .

Enganchando un método

Cree una nueva clase implementando `IXposedHookLoadPackage` e implemente el método `handleLoadPackage` :

```
public class MultiPatcher implements IXposedHookLoadPackage
{
    @Override
    public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
    {

    }
}
```

Dentro del método, verifica `loadPackageParam.packageName` para el nombre del paquete de la aplicación que deseas enganchar:

```
@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }
}
```

Ahora puede conectar su método y manipularlo antes de que se ejecute el código, o después de:

```
@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }

    XposedHelpers.findAndHookMethod(
        "other.package.name",
        loadPackageParam.classLoader,
        "otherMethodName",
        YourFirstParameter.class,
        YourSecondParameter.class,
        new XC_MethodHook()
    {
        @Override
        protected void beforeHookedMethod(MethodHookParam param) throws Throwable
        {
            Object[] args = param.args;

            args[0] = true;
            args[1] = "example string";
            args[2] = 1;

            Object thisObject = param.thisObject;

            // Do something with the instance of the class
        }

        @Override
        protected void afterHookedMethod(MethodHookParam param) throws Throwable
```

```
        {
            Object result = param.getResult();

            param.setResult(result + "example string");
        }
    });
}
```

Lea Xposed en línea: https://riptutorial.com/es/android/topic/4627/xposed

Creditos

S. No	Capítulos	Contributors
1	Empezando con Android	6londe, Abhishek Jain, Adam Johns, AesSedai101, Ahmad Aghazadeh, Akash Patel, Ala Eddine JEBALI, Aleksandar Stefanović, Andrea, Andrew Brooke, AndroidMechanic, ankit dassor, Apoorv Parmar, auval, Blachshma, Blundering Philosopher, cascal, cdeange, Charlie H, Charuಬ, ChemicalFlash, Cold Fire, Community, Dalija Prasnikar, Daniel Nugent, Daniele Segato, Doron Behar, Dr. Nitpick, Duan Bressan, EKN, Erik Minarini, Gabriele Mariotti, Gaket , gattsbr, geekygenius, hankide, Harish Gyanani, HCarrasko, Ibrahim, Ichthyocentaurs, inetphantom, Intrications, Irfan, Jeeter, JSON C11, Kevin, Kinjal, Kiran Benny Joseph, Laurel, Mark Yisri, Matas Vaitkevicius, MathaN, Menasheh, Michael Allan, mnoronha, mohit, MrEngineer13, Nick, Nick, opt05, Patel Pinkal, Pavneet_Singh, Pro Mode, PSN, RamenChef, Ravi Rupareliya, rekire, ridsatrio, russt, saul, Seelass, Shiven, Siddharth Venu, Simplans, Sneh Pandya, Sree, sudo, sun-solar-arrow, Tanis.7x, Thomas Gerot, ThomasThiebaud, Tot Zam, Vivek Mishra, Yury Fedorov, Zarul Izham, Ziad Akiki, Zoe, ושי בהוא רתוא
2	¿Qué es ProGuard? ¿Qué es el uso en Android?	Ayush Bansal, Daniel Nugent, Ghanshyam Sharma, Pratik Butani
3	Accediendo a bases de datos SQLite usando la clase ContentValues	Adil Saiyad, emecas, honk
4	ACRA	Zarul Izham, Zoe
5	Actividad	anoo_radha, Apoorv Parmar, Brenden Kromhout, Code.IT, Daniel Nugent, Floern, g4s8, Gunhan, H. Pauwelyn, HDehghani, Hiren Patel, Jacob Malachowski, johnrao07, Jordan, monK_, Nicolai Weitkemper, pRaNaY, RediOne1, SMR, Venner, Yury Fedorov, Zeeshan Shabbir
6	Actividades de pantalla dividida / multipantalla	Vishal Puri
7	ADB (Android Debug	3VYZkz7t, adao7000, Ahmad Aghazadeh, Amit Thakkar,

	Bridge)	AndroidMechanic, Anirudh Sharma, Anup Kulkarni, auval, Barend, Blackbelt, Burak Day, Charuක, Chris Stratton, Da-Jin C, Dale, Daniel Nugent, David Cheung, Erik, Fabio, fyfyone Google, g4s8, Gabriele Mariotti, grebulon, Hannoun Yassir, Hi I'm Frogatto, hichris123, honk, jim, Kashyap Jha, Laurel, MCeley, Menasheh, Natali, Nemus, Pavel Durov, Piyush, R. Zagórski, RishbhSharma, stkent, Sudip Bhandari, sukumar, theFunkyEngineer, thiagolr, Tien , Xaver Kapeller, Yassie, younes zeboudj, Yury Fedorov
8	AdMob	Carlos Borau, honk, RamenChef, Sukrit Kumar, Zarul Izham, Zoe
9	Advertencias de la pelusa	ben75, Daniel Nugent, Gabriele Mariotti, GensaGames, R. Zagórski, rekire, SuperBiasedMan
10	AIDL	Krishnakanth
11	AlarmManager	Daniel Nugent, devnull69, Greg T, honk, TR4Android
12	Almacenamiento de archivos en almacenamiento interno y externo	Amit Vaghela, Andrew Brooke, AnV, Daniel Nugent, Gabriele Mariotti, Nickan B, Uttam Panchasara
13	Añadiendo un FuseView a un proyecto de Android	Tudor Luca
14	Android NDK	Alex, astuter, Doron Yakovlev-Golani, Flayn, Onik, samgak , still_learning, Täg, thiagolr
15	Android Studio	AndroidMechanic, auval, Blackbelt, Charuක, Daniel Nugent, Gabriele Mariotti, Hiren Patel, Inzimam Tariq IT, Jon Adams, N J, Phan Van Linh, R. Zagórski, Squidward, Sujith Niraikulathan, ThomasThiebaud
16	Android Vk Sdk	alexey polusov
17	Android-x86 en VirtualBox	Daniel Nugent, Enrique de Miguel
18	Animadores	Aryan, Bartek Lipinski, Blundering Philosopher, Brenden Kromhout, Charuක, Daniel Nugent, Eixx, Hiren Patel, Lewis McGeary, Piyush, TR4Android, Uriel Carrillo, Yury Fedorov
19	Anotaciones Typedef: @IntDef, @StringDef	Gabriele Mariotti, hardik m, mmBs, Pongpat

20	API de Android Places	busradeniz, honk, Karan Razdan, Murali
21	API de conocimiento de Google	Dus, honk, Willie Chalmers III
22	API de Google Drive	Christlin Joseph, honk
23	API de Google Maps v2 para Android	AL., AndroidMechanic, antonio, Aryan, BadCash, Charuක, CptEric, Daniel Nugent, Hiren Patel, jgm, Mina Samy, narko, Onik, Pablo Baxter, RamenChef, Stephen Leppik, stkent, sukumar, Suresh Kumar, Vasily Kabunov
24	API de la cámara 2	ChemicalFlash, devnull69, RamenChef, webo80
25	API de Twitter	Mahmoud Ibrahim
26	API de Youtube	abhishesh, Giannis, honk, MashukKhan, Zarul Izham, Zeeshan Shabbir
27	Archivo zip en android	Adnan
28	Arquitectura MVP	Atif Farrukh, Harish Gyanani, honk, Jon Adams, Magesh Pandian, N J, zmingchun
29	AsyncTask	Ahmad Aghazadeh, Aiyaz Parmar, AndroidMechanic, Ashish Rathee, Brenden Kromhout, Carlos Borau, Daniel Nugent, devnull69, Dima Rostopira, Disk Crasher, Fabian Tamp, faranjit, Freddie Coleman, FredMaggiowski, Freek Nortier, Gabriele Mariotti, Hi I'm Frogatto, Hiren Patel, Ichigo Kurosaki, Jeeter, Joel Prada, Joost Verbraeken, JoxTraex, k3b, Leos Literak, marshmallow, MathaN, Michael Spitsin, Mike Laren, Mina Samy, Mohammed Farhan, Nick Cardoso, Nilanchala Panigrahy, Piyush, Raman, RamenChef, rciovati, Rohit Arya, Shashanth, SOFe, sudo, TameHog, Tejas Pawar, user1506104, Vasily Kabunov, vipsy, Zilk
30	AudioManager	honk, Nicolai Weitkemper
31	Autentificador de Android	4444, kRiZ
32	AutocompletarTextView	Harish Gyanani, Jon Adams, Ricardo Vieira, Vivek Mishra
33	Autosize TextViews	honk, Priyank Patel
34	Barra de progreso	Gabriele Mariotti, Hiren Patel, mpkuth, Sanoop, shtolik
35	Base de datos en tiempo real de Firebase	Aawaz Gyawali, drulabs, Gabriele Mariotti, honk, Md. Ali Hossain, RamenChef, Sneh Pandya, Stephen Leppik, yennsarah, Zarul Izham

36	Base de fuego	Albert, AndiGeeky, AndroidMechanic, Chintan Soni, Cows quack, Daniel Nugent, Egek92, Gabriele Mariotti, krunal patel, Leo, Omar Aflak, ppeterka, RamenChef, Saeed-rz, Sanket Berde, shahharshil46, Sneh Pandya, Stephen Leppik, sukumar
37	Biblioteca de enlace de datos	Ahmad Aghazadeh, astuter, Avinash R, Bryan Bryce, Caique Oliveira, Daniel Nugent, David Argyle Thacker, Fabian Mizieliński, gaara87, Gabriele Mariotti, Guillaume Imbert, H. Pauwelyn, Iulian Popescu, Jon Adams, Lauri Koskela, Long Ranger, MidasLefko, RamenChef, Ravi Rupareliya, Razan, rciovati, Rule, Segun Famisa, Stephen Leppik, Tanis.7x, Vlonjat Gashi, yennsarah
38	Bluetooth Low Energy	Roberto Betancourt
39	Bluetooth y Bluetooth LE API	antonio, Jon Adams, Lukas, Myon, Pavel Durov, R. Zagórski, Reaz Murshed, V-PTR, WMios
40	Botón	Aleksandar Stefanović, BlitzKraig, Carlos Borau, Community, Daniel Nugent, Gabriele Mariotti, James_Parsons, Jordi Castilla, Mauro Frezza, Michael Spitsin, Muhammed Refaat, Nick Cardoso, Nougat Lover, r3flss ExlUtr, RamenChef, Ricardo Vieira, sun-solar-arrow, webo80
41	Botón de acción flotante	Ahmad Aghazadeh, Charuක, Daniel Nugent, Gabriele Mariotti, mattfred, RamenChef, Shinil M S, Stephen Leppik
42	Caché de mapa de bits	Lokesh Desai
43	Camara y galeria	Ahmad Aghazadeh, carvaq, Daniel Nugent, Hiren Patel, johnrao07, RediOne1, Squidward, Yasin Kaçmaz
44	Cambios de orientación	EmmanuelMess, k3b, Ricardo Vieira, y.feizi
45	Captura de capturas de pantalla	Ayush Bansal, Daniel Nugent, honk, Onik, sushant kumar, W0rmH0le
46	CardView	Carlos, Dan, Er. Kaushik Kajavadara, Gabriele Mariotti, Kaushik, Nougat Lover, RamenChef, S.R, Sneh Pandya, Somesh Kumar, Stephen Leppik, sud007, WarrenFaith, Yury Fedorov
47	Cargador	chandsie, g4s8, jefry jacky, Marcus Becker, RamenChef, Stephen Leppik
48	Cargador de Imagen Universal	Greg T, honk, Jon Adams, priyankvex, Stephen Leppik

49	Cargando Bitmaps Efectivamente	iDevRoids
50	carril rápido	Gokhan Arik
51	Ciclo de vida de la interfaz de usuario	Daniel Nugent, Dinesh Choudhary, Floern, Lewis McGeary, orelzion, R. Zagórski, Sergey Glotov
52	Cifrado / descifrado de datos	honk, HoseinIT, Robert
53	CleverTap	Jordan, judepereira
54	Colores	Carlos Borau, Dalija Prasnikar, Daniel Nugent, Erfan Mowlaei, Jon Adams, N J, Sujith Niraikulathan
55	Comenzando con OpenGL ES 2.0+	MarGenDo
56	Cómo almacenar contraseñas de forma segura	honk, Jaggs
57	Cómo utilizar SparseArray	honk, Robert Banyai
58	Componentes de la arquitectura de Android	DeKaNszn
59	Compresión de imagen	Hiren Patel, mnoronha
60	Compruebe la conectividad a internet	AndiGeeky, Bill, Daniel Nugent, gbansal, Ichigo Kurosaki, Jon Adams, sukumar, TameHog, Yousha Aleayoub
61	Compruebe la conexión de datos	sukumar, Suresh Kumar
62	Conexiones Wi-Fi	4444, AndroidMechanic, Daniel Nugent, gus27
63	Configuración de Jenkins CI para proyectos de Android	honk, Ichthyocentaurs
64	Construyendo aplicaciones compatibles hacia atrás	Jon Adams, mnoronha, RamenChef, SoroushA
65	Contador regresivo	privatestaticint
66	Contexto	Will Evers

67	Conversión de voz a texto	Hitesh Sahu, honk, RamenChef, Stephen Leppik
68	Convertir cadena vietnamita a la cadena inglesa Android	1SStorm
69	Coordinador de Aula y Comportamientos	Adarsh Ashok, Gabriele Mariotti, honk, RamenChef, Stephen Leppik
70	Cosas de Android	Fabio, honk
71	Crea ROMs personalizadas de Android	honk, Pradumn Kumar Mahanta
72	Creación de superposición (siempre en la parte superior) de Windows	honk, mnoronha, NitZRobotKoder, Rupali, Sujith Niraikulathan
73	Creación de vistas personalizadas	AndroidMechanic, Barend, Bartek Lipinski, Charuज, Daniel Nugent, Dinesh, g4s8, Harish Gyanani, Hiren Patel, Joel Gritter, Jon Adams, Omar Al Halabi, PcAF, R. Zagórski, rciovati, Sneh Pandya, Sujith Niraikulathan, Suragch, TR4Android, Yury Fedorov
74	Creando pantalla de bienvenida	honk, Kiran Benny Joseph, Zoe
75	Creando tus propias bibliotecas para aplicaciones de Android	EpicPandaForce, honk, mnoronha
76	Crear una clase Singleton para un mensaje de Toast	Emad, Ishan Fernando
77	Cuadro de diálogo animado de alerta	krunal patel, Thomas Easo
78	Cuchillo de mantequilla	Abdellah, Alex Sullivan, Andrei Ancuța, AndroidMechanic, AndroidRuntimeException, astuter, FiN, H. Pauwelyn, Joaquin Iurchuk, Jordan, Max, mmBs, Nougat Lover, Paresh Mayani, RamenChef, ridsatrio, Rucha Bhatt, Sir SC , Stephen Leppik, StuStirling, Thibstars, Tot Zam, Volodymyr Buberenko, ZeroOne
79	Cuentas y	gaara87, systemovich

	AccountManager	
80	Daga 2	Aurasphere, Cabezas, David Medenjak, EpicPandaForce, honk, mattfred, Tomik
81	Defina el valor del paso (incremento) para la barra de barras personalizada	Romu Dizzy
82	Desarrollo de juegos para Android	Zoe
83	Descomprimir archivo en Android	Arth Tilva, Daniel Nugent, mnoronha
84	Deslizamiento	Anand Singh, AndroidMechanic, AndroidRuntimeException, antonio, Chol, Daniel Nugent, Daniele Segato, Gabriele Mariotti, Ilya Krol, Lewis McGeary, Lucas Paolillo, Mauker, Max, mhenryk, Milad Nouri, RamenChef, Ramzy Hassan, Ravi Rupareliya, Reaz Murshed, Rohit Arya, Rucha Bhatt, Sam Judd, Sneh Pandya, Stephen Leppik, sukumar, Vlonjat Gashi, ZeroOne
85	Deslizar para actualizar	Chirag SolankI, Daniel Nugent, Gabriele Mariotti, Malek Hijazi, RamenChef, Stephen Leppik
86	Detección de gestos	mpkuth
87	Detect Shake Event en Android	N-JOY, tynn, Xiaozou
88	Diálogo	Ab_, adalPaRi, Aleks G, alexey polusov, Brenden Kromhout, Daniel Nugent, Ichigo Kurosaki, Jaymes Bearden, JJ86, Lewis McGeary, M D P, Mochamad Taufik Hidayat, Rajesh, RamenChef, Ravi Rupareliya, RediOne1, Sanket Berde, ShivBuyya, Yojimbo, Zoe
89	Dibujables	alanv, B001, Daniel Nugent, Greg T, Hiren Patel, Jinesh Francis, Nick Cardoso, TR4Android
90	Dibujos vectoriales	Priyank Patel, ShahiM
91	Diseño de materiales	Akash Patel, Aleksandar Stefanović, Alex Chengalan, AndroidMechanic, Anirudh Sharma, ankit dassor, Bartek Lipinski, Bulwinkel, cascal, Charuක, dakshbhatt21, Dan Hulme, Daniel Nugent, dev.mi, Eixx, fyfyone Google, Gabriele Mariotti, Gal Yedidovich, Guillermo García, honk, Ibrahim, Ichigo Kurosaki, Ishita Sinha, Jaiprakash Soni,

		jlynch630, Jon Adams, Lewis McGeary, Lucas Paolillo, Machado, mahmoud moustafa, Marina K., MathaN, Max, Menasheh, mmBs, mpkuth, N J, Nikita Kurtin, noongiya95, oshurmamadov, pavel163, Piyush, Pravin Sonawane, Rajesh, RamenChef, rciovati, Reaz Murshed, RediOne1, ridsatrio, Sagar Chavada, Sanoop, sat, Saveen, Shashanth , Simo, SimplyProgrammer, Sneh Pandya, Stephen Leppik, sud007, sudo, sukumar, Uttam Panchasara, Vasily Kabunov, vguzzi, Vivek Mishra, Willie Chalmers III, X3Btel, Xaver Kapeller, Yasin Kaçmaz, Yury Fedorov
92	Diseños	a.ch., Adarsh Ashok, Adinia, AesSedai101, Ahmad Aghazadeh, Aleksandar Stefanović, ankit dassor, Aurasphere, Bartek Lipinski, Björn Kechel, bjrne, Brenden Kromhout, Charuज, Dan Hulme, Daniel Nugent, devnull69, Floern, Gabriele Mariotti, Gaurav Jindal, Gurgen Hakobyan , Infinite Recursion, Kaushik NP, Knossos, Lewis McGeary, Michael Spitsin, MiguelHincapieC, Mr.7, Nepster, Patrick Dattilio, Phan Van Linh, Rajesh, rciovati, rekire, Sir SC, Sneh Pandya, Talha Mir, ThomasThiebaud, Tim Kranen, Trilarion, ubuntudroid, Vasily Kabunov, Yury Fedorov
93	Editar texto	Daniel Nugent, Gabriele Mariotti, Kaushik NP, Muthukrishnan Rajendran, Rubin Nellikunnathu, Yousha Aleayoub
94	Ejecución instantánea en Android Studio	AndroidMechanic, Daniel Nugent, ridsatrio, Zoe
95	El archivo de manifiesto	John Snow, Jon Adams, kit, mayojava, Menasheh
96	Emulador	Ahmad Aghazadeh, Dan Hulme, fyfyone Google, honk, rekire, Rubin Nellikunnathu, ThomasThiebaud
97	Entrenador de animales	Daniel Nugent, Floern, Hasif Seyd, Lewis McGeary, Mike Scamell, Muhammed Refaat, Sweeper, Tomik, TR4Android
98	Escribir pruebas de interfaz de usuario - Android	Atif Farrukh, Daniel Nugent, Gabriele Mariotti, honk, Jon Adams, originx
99	Eventos / intenciones de botón de hardware (PTT, LWP, etc.)	JensV
100	Eventos táctiles	Yvette Colomb
101	Excepciones	abhishesh, AesSedai101, Alex Gittemeier, antonio, astuter, Buddy, Damian Kozlak, Gabe Sechan, Greg T, Jeeter,

		Lewis McGeary, M D P, Nick Cardoso, PhilLab, Simone Carletti, THelper, ThomasThiebaud, Xaver Kapeller
102	ExoPlayer	Hamed Gh
103	Facebook SDK para Android	Akeshwar Jha, AndiGeeky, Community, Daniel Nugent, honk, Zarul Izham
104	Facturación en la aplicación	Hussein El Feky, Pro Mode, Zoe
105	Fastjson	KeLiuyue
106	Fecha / Hora localizada en Android	Geert, honk, mnoronha
107	FileIO con Android	h22, sun-solar-arrow
108	FileProvider	Joost Verbraeken, pedros
109	Firebase Cloud Messaging	Gabriele Mariotti, shikhar bansal, Shubham Shukla, Zarul Izham
110	Firebase Crash Reporting	AndiGeeky, Gabriele Mariotti, honk, RamenChef, Stephen Leppik, Zarul Izham
111	Firma tu aplicación de Android para su lanzamiento	Gabriele Mariotti, M M
112	Formato de cadenas	Beena, Daniel Nugent, gaara87, Greg T, Michele, RamenChef, Suresh Kumar
113	Formato de números de teléfono con patrón.	Pedro Varela
114	Fragmentos	Adarsh Ashok, A-Droid Tech, Ahmad Aghazadeh, Amit, Anish Mittal, auval, Ben P., Chirag Jain, cricket_007, Damian Kozlak, Daniel Nugent, Erfan Mowlaei, Erik Minarini, g4s8, Gabriele Mariotti, Hi I'm Frogatto, Hiren Patel, jgm, Jordan, K_7, Makille, Nandagopal T, Narayan Acharya, Parsania Hardik, Phan Van Linh, RamenChef, Stephen Leppik
115	Fresco	Alexander Oprisnik, Daniel Nugent, honk, Nilesh Singh, Zarul Izham
116	Fuentes personalizadas	Daniel Nugent, Erik Ghonyan, Gabriele Mariotti, Hiren Patel, honk, kit, Nougat Lover, Simon Schubert, Stanojkovic, Sujith Niraikulathan

117	Genymotion para android	Atef Hares, Harish Gyanani
118	Gerente de empaquetación	FredMaggiowski, Hi I'm Frogatto, Muthukrishnan Rajendran, Piyush, Squidward
119	Google Play Store	dakshbhatt21, Daniel Nugent, reVerse
120	Gradle para Android	4444, Aaron He, Abdul Wasae, abhi, Abhishek Jain, Ahmad Aghazadeh, Alex T., AndroidMechanic, AndroidRuntimeException, Anirudh Sharma, Ankit Sharma, Arpit Patel, auval, Bartek Lipinski, Ben, Brenden Kromhout, bwegs, cascal, cdeange, Charuಎ, ChemicalFlash, cricket_007, Daniel Nugent, enrico.bacis, Eugen Martynov, Fabio, Floern, Florent Spahiu, Gabriele Mariotti, hankide, Ibrahim, Ichthyocentaurs, Irfan, jgm, k3b, Kevin Crain, kevinpelgrims, Matt, mshukla, N J, Pavel Strelchenko, Pavneet_Singh, R. Zagórski, RamenChef, rciovati, Reaz Murshed, rekire, Revanth Gopi, Sneh Pandya, sun-solar-arrow, ThomasThiebaud, ɻolɐəz əɥɻ qoq, Vlonjat Gashi, Yassie, yuku, Yury Fedorov
121	GreenDAO	Allan Pereira, Carl Poole, Grundy, MiguelHincapieC, R. Zagórski, RamenChef, Stephen Leppik
122	GreenRobot EventBus	CaseyB, Daniel Nugent, Hamed Momeni, RamenChef
123	Gson	AndroidRuntimeException, baozi, cdeange, Code_Life, cricket_007, Daniel Nugent, DanielDiSu, devnull69, Duan Bressan, Gabriele Mariotti, Ginandi, Graham Smith, Harish Gyanani, L. Swifter, Mauker, Oleksandr, Prownage, Rucha Bhatt, Sneh Pandya, Tim Kranen, Vincent D., Yury Fedorov
124	Herramientas Atributos	Dalija Prasnikar, Gabriele Mariotti, Harsh Sharma, Kayvan N, TR4Android
125	Herramientas de informes de bloqueo	Ajit Singh, Charuಎ, Ekin, Gabriele Mariotti, Ishita Sinha, Jason Bourne, Madhukar Hebbar, pRaNaY
126	Hilandero	AndroidMechanic, Anonsage, Daniel Nugent, Vishwesh Jainkuniya
127	Hilo	Daniel Nugent, PRIYA PARASHAR, RamenChef
128	Hojas inferiores	Daniel Nugent, Gabriele Mariotti, Magesh Pandian, MiguelHincapieC, RamenChef, Stephen Leppik, sud007, Zarul Izham
129	HttpURLConnection	Aleks G, Daniel Nugent, Duan Bressan, honk,

		KDeogharkar, marshmallow, Shantanu Paul, Simone Carletti
130	Huella digital API en Android	Doron Yakovlev-Golani, RamenChef, user01232
131	Imágenes de 9 parches	Knossos, Nissim R, Tomik
132	ImageView	Ahmad Aghazadeh, Ali Sherafat, Chip, Daniel Nugent, DanielDiSu, Dinesh, Gabriele Mariotti, Harish Gyanani, kit, lax1089, Pratik Butani, Squidward, Sup
133	Indexación de la aplicación Firebase	shalini, tynn
134	Instalando aplicaciones con ADB	Ahmad Aghazadeh, fyfyone Google, Laurel, Xaver Kapeller
135	Integración de Android Paypal Gateway	A-Droid Tech
136	Integración de inicio de sesión de Google en Android	jagapathi
137	Integrar el inicio de sesión de Google	AndiGeeky, RamenChef, Tot Zam
138	Integrar OpenCV en Android Studio	MashukKhan, RamenChef, ssimm
139	Intención	4444, Abdallah Alaraby, Abdullah, abhi, Abhishek Jain, AER, ahmadalibaloch, Akshit Soota, Alex Logan, Andrew Brooke, Andrew Fernandes, AndroidMechanic, AndroidRuntimeException, Anirudh Sharma, Anish Mittal, antonio, Apoorv Parmar, auval, Avinash R, Bartek Lipinski, Blundering Philosopher, bpoiss, cascal, Charuಬ, Clinton Yeboah, Code.IT, Cold Fire, dakshbhatt21, Dalija Prasnikar , Daniel Käfer, Daniel Nugent, Daniel Stradowski, DanielDiSu, Dave Thomas, David G., Devid Farinelli, devnull69, DoNot, DVarga, Eixx, EKN, Erik Minarini, faranjit , Floern, fracz, Franck Dernoncourt, g4s8, Gabriele Mariotti , GingerHead, granmirupa, Harish Gyanani, Hi I'm Frogatto , Ibrahim, iliketocode, insomniac, Irfan, Irfan Raza, Ironman , Ivan Wooll, Jarrod Dixon, jasonlam604, Jean Vitor, jhoanna, JSON C11, Justcurious, kann, Karan Nagpal, Kayvan N, Lee, leodev, Lewis McGeary, MalhotraUrmil, Mark Ormesher, MathaN, Mauker, Max, mnoronha, Mr. Sajid Shaikh, Muhammed Refaat, muratgu, N J, Nick

		Cardoso, niknetniko, noɥʎdʎzɐɹƆ, Oren, Paresh Mayani, Parsania Hardik, Paul Lammertsma, Pavneet_Singh, penkzhou, Peter Mortensen, Phan Van Linh, Piyush, R. Zagórski, Radouane ROUFID, Rajesh, RamenChef, rap-2-h, rciovati, Reaz Murshed, RediOne1, rekire, reVerse, russjr08, Ryan Hilbert, sabadow, Saveen, Simon, Simplans , SoroushA, spaceplane, Stelian Matei, Stephane Mathis, Stephen Leppik, sukumar, tainy, theFunkyEngineer, ThomasThiebaud, ɹolɐəz əɥɹ qoq, Tyler Sebastian, vasili111, Vasily Kabunov, Vinay , Vivek Mishra, Xaver Kapeller, younes zeboudj, Yury Fedorov, Zoe
140	Intenciones implícitas	Blundering Philosopher, Daniel Nugent, mnoronha, Pratik Butani, SoroushA
141	Inter-app UI testing con UIAutomator	Timo Bähr
142	Interfaces	appersiano, Brenden Kromhout, Daniel Nugent, RediOne1
143	Interfaz nativa de Java para Android (JNI)	Doron Yakovlev-Golani, Muthukrishnan Rajendran, samgak
144	Internacionalización y localización (I18N y L10N)	Ankur Aggarwal
145	Jackson	KeLiuyue
146	Java en Android	Eugen Martynov
147	JCodec	Adhikari Bishwash
148	JSON en Android con org.json	Abhishek Jain, AndroidMechanic, AndroidRuntimeException, baozi, Ben Trengrove, cdeange , Daniel Nugent, Diti, Eliezer, Endzeit, Florent Spahiu, Gabriele Mariotti, ganesshkumar, gerard, Graham Smith, harsh_v, Ic2h, IncrediApp, johnrao07, Kaushik, L. Swifter, Linda, Luca Faggianelli, Mannaz, Mauker, Michael Spitsin, Monish Kamble, Muhammed Refaat, N J, Oleksandr, Parsania Hardik, Prownage, rekire, Siddhesh, StuStirling, ThomasThiebaud, Tim Kranen, user01232, Vincent D., Xaver Kapeller, younes zeboudj, Yury Fedorov
149	Leakcanary	Rakshit Nawani, tynn
150	Lectura de códigos de barras y códigos QR	FlyingPumba

151	Library Dagger 2: Inyección de dependencia en aplicaciones	Er. Kaushik Kajavadara, honk
152	Lienzo de dibujo utilizando SurfaceView	davidgiga1993
153	Localización con recursos en Android.	AndroidMechanic, electroid, Fabio, Gubbel, Harish Gyanani, honk, Jinesh Francis, mpkuth, RamenChef, USKMobility
154	Looper	tynn
155	LruCache	Daniel Nugent, honk, LordSidious, RamenChef, Stephen Leppik
156	Manejo de enlaces profundos	Doron Yakovlev-Golani, Harsh Sharma, mnoronha, Tanis.7x
157	Manejo de eventos táctiles y de movimiento.	honk, Zoe
158	Mapeo de puertos usando la biblioteca Cling en Android	Shinil M S
159	MediaSession	Disk Crasher, honk, KuroObi, RamenChef
160	Mediastore	Daniel Nugent, honk, RamenChef, Uttam Panchasara
161	Mejora de los diálogos de alerta	Adil Saiyad, honk
162	Mejora del rendimiento de Android utilizando fuentes de iconos	Beto Caldas, honk, Neeraj
163	Menú	Bhargavi Yamanuri, Chip, Daniel Nugent, Hi I'm Frogatto, honk, Iman Hamidi
164	Métricas de la pantalla del dispositivo	Daniel Nugent, Hiren Patel, Talha, W3hri
165	Modo Doze	Daniel Nugent, Fabio, honk, NitZRobotKoder, RamenChef, Rosário Pereira Fernandes, Rupali
166	Modo PorterDuff	Adarsh Ashok, AndroidMechanic, Knossos, PhilLab, S.D., Vasily Kabunov

167	Moshi	Blundell
168	Multidex y el método Dex Limit	Adarsh Ashok, Ben, bigbaldy, cdeange, Daniel Nugent, Gabriele Mariotti, Mike, Pongpat, R. Zagórski, Shirane85
169	MVVM (Arquitectura)	Daniel W., RamenChef, Stephen Leppik
170	NavigationView	Adam Lear, akshay, Charuಎ, Daniel Nugent, Gabriele Mariotti, Kedar Tendolkar, petrumo, RamenChef, rekire, SANAT, Sevle, Stephen Leppik, sud007
171	Notificacion canal android o	Lokesh Desai
172	Notificaciones	alexey polusov, bricklore, Da-Jin C, Daniel Nugent, Dus, gbansal, Jeeter, piotrek1543, RediOne1, Rupali, TR4Android, weston
173	Obtención de dimensiones de vista calculadas	mnoronha, stkent
174	Obtención de nombres de fuentes del sistema y uso de las fuentes	Adil Saiyad, honk
175	OkHttp	A-Droid Tech, Daniel Nugent, Gabriele Mariotti, Gubbel, noob, Rohit Arya, Vucko, Zarul Izham
176	Okio	Adhikari Bishwash
177	Optimización del núcleo de Android	honk, Sneh Pandya
178	Optimización del rendimiento	honk, Jonas Köritz
179	ORMLite en Android	Manos
180	Otto Event Bus	gus27, tynn
181	Paginación en RecyclerView	Muhammad Younas
182	Pantallas de apoyo con diferentes resoluciones, tamaños	Eduardo, Guilherme Torres Castro, kalan, mpkuth, Onur, ppeterka
183	Parcelable	Alex Sullivan, Andrei T, HoseinIT, Nick Cardoso

184	Patrones de diseño	Adhikari Bishwash, honk, Steve.P
185	Pérdidas de memoria	Abhishek Jain, Anand Singh, auval, Ben, cascal, CodeHarmonics, commonSenseCode, Daniel Nugent, david.schreiber, Disk Crasher, Gabriele Mariotti, geniushkg , honk, Kingfisher Phuoc, Leos Literak, Mikael Ohlson, Mohammod Hossain, mrtuovinen, Oren, RamenChef, Risch, Saveen, ɬolɐəz əɥɬ qoq
186	Permisos de tiempo de ejecución en API-23 +	Ahmad Aghazadeh, AndroidMechanic, AndroidRuntimeException, Buddy, Daniel Nugent, Erik Minarini, Floern, Gubbel, honk, Jaseem Abbas, Kayvan N, Lewis McGeary, Luksprog, Madhukar Hebbar, nagyben, null pointer, Olu, Pavneet_Singh, Piyush, Prakash Gajera, RamenChef, RediOne1, Vivek Mishra, yuku, Yvette Colomb
187	Picasso	astuter, Brenden Kromhout, Daniel Nugent, Gabriele Mariotti, Ichthyocentaurs, LoungeKatt, Milad Nouri, once2go, oshurmamadov, Piyush, pRaNaY, Pro Mode, RamenChef, Rucha Bhatt, Sanket Berde, Shinil M S, Ufkoku, VISHWANATH N P, vrbsm, y.feizi
188	Ping ICMP	Carl Poole
189	Pintar	Nicolas Maltais
190	Pista de audio	Ayush Bansal
191	Política de modo estricto: una herramienta para detectar el error en el tiempo de compilación.	Shekhar
192	Preferencias compartidas	Abhishek Jain, Ahmad Aghazadeh, akshay, AndroidMechanic, Anggrayudi H, antonio, Ashish Ranjan, Blackbelt, Blundering Philosopher, Buddy, Dalija Prasnikar, Damian Kozlak, Dan Hulme, Daniel Nugent, FisheyLP, Gabriele Mariotti, gbansal, Greg T, IncrediApp, Jon Adams, JonasCz, jonathan3087, Jordan, Kayvan N, LordSidious, Makille, Max McKinney, Pawel Cala, Piyush, rajan ks, rekire, Rohit Arya, Sándor Mátyás Márton, Shinil M S, ShivBuyya, Suchi Gupta, TanTN, TheLittleNaruto, Trevor Clarke, user1506104, Vasily Kabunov, vipsy, Vishva Dave, Volodymyr Buberenko, xmoex, Yury Fedorov
193	Procesador de anotaciones	krishan

194	Programación de Android con Kotlin.	Gian Patrick Quintana, Govinda Paliwal, Oknesif, Zarul Izham
195	Programación de trabajos	RamenChef, reflective_mind
196	ProGuard - ofuscar y encoger su código	activesince93, Aman Anguralla, Anirudh Sharma, auval, Daniel Nugent, EKN, Ibrahim, J j, Jon Adams, Lewis McGeary, Lukas Abfalterer, Max, Nikita Shaposhnik, R. Zagórski, Ricardo Vieira
197	Proveedor de contenido	Andrew Siplas, cdeange, Daniel Nugent, Dinesh Choudhary, Lewis McGeary, RamenChef
198	Prueba de interfaz de usuario con espresso	Daniel Nugent, Gabriele Mariotti, Jason Robinson, Michael Vescovo, Milad Faridnia, N J, RamenChef, Víctor Albertos
199	Pruebas unitarias en Android con JUnit.	abhi, Andre Perkins, AndroidMechanic, Eugen Martynov, honk, Lewis McGeary, N J, Namnodorel, Patrick Dattilio, Rolf ツ
200	Publicar el archivo .aar en Apache Archiva con Gradle	Marian Klühspies
201	Publicar en Play Store	Carlos Borau, Fabio, mnoronha, Zoe
202	Publicar una biblioteca en Repositorios Maven	Farid
203	Receptor de radiodifusión	0x0000eWan, Adarsh Ashok, anupam_kamble, Daniel Nugent, g4s8, Hiren Patel, Ichthyocentaurs, Jon Adams, Joscandreu, Kirill Kulakov, Lazy Ninja, Leo.Han, Medusalix , param, Phil, Rajesh, Squidward, W0rmH0le
204	Recolectores de fecha y hora	adalPaRi, Brenden Kromhout, Daniel Nugent, Harish Gyanani, Ironman, Milad Nouri, RediOne1, Rohan Arora
205	Reconocimiento de actividad	Pablo Baxter
206	Recursos	biddulph.r, Brenden Kromhout, Charuಎ, Dalija Prasnikar, Daniel Nugent, Floern, Gabriele Mariotti, Graham Smith, Harish Gyanani, honk, KDeogharkar, Menasheh, Nick Cardoso, Noise Generator, Piyush, R. Zagórski, reVerse, Tanis.7x, ThomasThiebaud, Vivek Mishra, Xavier
207	RecyclerView	Abhishek Jain, Abilash, Adinia, Ahmad Aghazadeh, Akash Patel, Alex Bonel, Alok Omkar, anatoli, Andrii Abramov,

		AndroidMechanic, Anirudh Sharma, BalaramNayak, Barend, Bartek Lipinski, Bryan, cascal, Charuඞ, Chirag SolankI, Daniel Nugent, Fahad Al-malki, Felix Edelmann, FromTheSeventhSky, Gabriele Mariotti, GensaGames, humazed, Ironman, Jacob, jgm, Joel Mathew, Jon Adams, Joshua, Kayvan N, keineantwort, Kevin DiTraglia, Knossos , kyp, MathaN, MidasLefko, MKJParekh, mklimek, Pablo Baxter, Patrick Dattilio, Piyush, raktale, RamenChef, rciovati, Reaz Murshed, Rohan Arora, Sagar Chavada, Sanket Berde, Sasank Sunkavalli, Sneh Pandya, Stephen Leppik, sukumar, Sunday G Akinsete, thetonrifles, Tot Zam , Uttam Panchasara, V. Kalyuzhnyu, Vasily Kabunov, Xaver Kapeller, Yasin Kaçmaz, Yura Ivanov, Yury Fedorov, Zilk
208	RecyclerView Decoraciones	Barend, David Medenjak, Gabriele Mariotti, Muthukrishnan Rajendran, Peter Gordon, RamenChef, Stephen Leppik, Yasin Kaçmaz
209	RecyclerView onClickListeners	abhishesh, Braj Bhushan Singh, Bryan, FromTheSeventhSky, fuwaneko, Gabriele Mariotti, honk, RamenChef, Smit.Satodia, Stephen Leppik
210	RecyclerView y LayoutManagers	4444, BalaramNayak, Felix Edelmann, Gabriele Mariotti, Kayvan N, MidasLefko, RamenChef, Stephen Leppik
211	Registro y uso de Logcat	Adam Ratzman, akshay, Alexander Mironov, alexey polusov, Anand Singh, AndroidMechanic, astuter, auval, Daniel Nugent, Eugen Martynov, faranjit, FromTheSeventhSky, gattsbr, Jeeter, Jon Adams, Laurel, LaurentY, Manan Sharma, Mario Lenci, Piyush, pRaNaY, Pratik Butani, rekire, russt, Sujith Niraikulathan, TDG, thiagolr, Yury Fedorov, Zachary David Saunders
212	Reino	bdash, Dan, EpicPandaForce, Hi I'm Frogatto, iurysza, null pointer, RamenChef, Stephen Leppik, sukumar
213	RenderScript	Ankit Popli, Dalija Prasnikar, Froyo, honk, Rucha Bhatt, Xaver Kapeller
214	Reproductor multimedia	Ahmad Aghazadeh, Carlos Vázquez Losada, hello_world, Makille, R. Zagórski, Redman
215	RestricciónDisposición	Adarsh Ashok, Bryan, Daniel Nugent, Darish, Florent Spahiu, Gabriele Mariotti, KorolevSM, Marcola, MathaN, Pratik Butani, RamenChef, Samvid Mistry, Sneh Pandya, Stephen Leppik, Yury Fedorov, Zarul Izham
216	RestricciónSet	Pratik Butani

217	Retrofit2	Adarsh Ashok, Adnan, Anderson K, AndroidMechanic, AndyRoid, aquib23, arcticwhite, CaseyB, Cassio Landim, Dan, Daniel Nugent, DanielDiSu, devnull69, Dhaval Solanki, FiN, Greg T, Kamran Ahmed, KATHYxx, Kaushik, mrtuovinen, NashHorn, Omar Al Halabi, param, Pavneet_Singh, Pinaki Acharya, R. Zagórski, RamenChef, SKen, Sneh Pandya, Stephen Leppik, xdk78, Zarul Izham
218	Retrofit2 con RxJava	Anand Singh, gaara87, GurpreetSK95, Lukas, mrtuovinen, R. Zagórski, Zarul Izham
219	RoboGuice	AndroidRuntimeException, Lewis McGeary, Rajesh
220	Robolectric	Blundell, g4s8
221	SearchView	Daniel Nugent, Dmide, Hiren Patel, RamenChef, Stephen Leppik, sud007
222	Secure SharedPreferences	Christlin Joseph
223	Seguridad	xDragonZ
224	SensorManager	honk, Simon, TDG
225	Servicio	adao7000, AndroidMechanic, Apoorv Parmar, BadCash, B-GangsteR, Daniel Nugent, g4s8, Hiren Patel, JonasCz, Lazai, Lucas Paolillo, Michael Spitsin, Nougat Lover, rakeshdas, Vinícius Barros
226	Servicio de Intención	Anax, Daniel Nugent, honk, JonasCz, TRINADH KOYA, Yashaswi Maharshi
227	shell adb	3VYZkz7t, Ahmad Aghazadeh, auval, Burak Day, Fabio, fyfyone Google, Hannoun Yassir, Natali, Pavel Durov, R. Zagórski, sukumar, Yury Fedorov
228	ShortcutManager	g4s8, Sukrit Kumar
229	Sincronización de datos con el adaptador de sincronización	Arpit Gandhi, mnoronha
230	Snackbar	AndroidRuntimeException, Charuක, Daniel Nugent, Gabriele Mariotti, Harsh Pandey, Jinesh Francis, Lithimlin, marshmallow, Mike Scamell, miss C, Mochamad Taufik Hidayat, Patrick Dattilio, Piyush, RamenChef, Rasoul Miri, Rosário Pereira Fernandes, Sneh Pandya, Stephen Leppik , Zarul Izham

231	Sonido y Medios Android	johnrao07, Muhammad Umair Shafique, Squidward
232	SpannableString	S.R
233	SQLite	Abhishek Jain, AndroidMechanic, ankit dassor, Ashwani Kumar, astuter, CL., dakshbhatt21, Damian Kozlak, Daniel Nugent, falvojr, Gabriele Mariotti, Gorg, H. Pauwelyn, Ilya Blokh, Jitesh Dalsaniya, JJ86, John Slegers, Lazy Ninja, Leos Literak, Lewis McGeary, Lucas Paolillo, Mauker, McSullivan D'Ander, MIkka Marmik, MPhil, Robin Dijkhof, Scott W, Uriel Carrillo, Vasily Kabunov, WMios, Xaver Kapeller, Yury Fedorov
234	SyncAdapter con periódicamente hacer sincronización de datos	Bhargavi Yamanuri
235	TabLayout	Daniel Nugent, Willie Chalmers III
236	Tarjeta electrónica	shadygoneinsane
237	Teclado	Hiren Patel, Kayvan N
238	Tema DayNight (AppCompat v23.2 / API 14+)	Ishita Sinha
239	Tema, Estilo, Atributo	alanv, Aleksandar Stefanović, cdeange, Daniel Nugent, DanielDiSu, Gabriele Mariotti, Hiren Patel, Ishita Sinha, Jason Robinson, Laurel, noob, Piyush, R. Zagórski, RamenChef, Tot Zam, Vlonjat Gashi
240	TensorFlow	Pratik Butani
241	TextInputLayout	Adarsh Ashok, BrickTop, Gabriele Mariotti, Hi I'm Frogatto, RamenChef, Shashanth, Sneh Pandya, Stephen Leppik
242	Texto a voz (TTS)	Ahmad Aghazadeh, honk, Jordan, Lukas, nibarius, Peter Taylor, RamenChef, Stephen Leppik
243	tostada	Adam Ratzman, adao7000, Aida Isay, Amit, Andrew Brooke, AndroidMechanic, Avijit Karmakar, Bartek Lipinski, cdeange, Charu๓, Daniel Nugent, Erik Ghonyan, Gabriele Mariotti, Lewis McGeary, LordSidious, Lukas, mpkuth, MrSalmon, RamenChef, Rohit Arya, Sammy T, saurav, SoroushA, sukumar, Vicky, Vucko
244	Transiciones de elementos compartidos	noongiya95

245	TransitionDrawable	S.R, Yogesh Umesh Vaity
246	Ubicación	Alex Chengalan, Aryan, BadCash, Daniel Nugent, Hiren Patel, Mahmoud Ibrahim, MidasLefko, Pablo Baxter, RamenChef, Stephen Leppik
247	Una forma rápida de configurar Retrolambda en un proyecto de Android.	anatoli, Md. Ali Hossain
248	URL de devolución de llamada	Atif Farrukh, honk, RamenChef, Stephen Leppik
249	Utilidades de tiempo	Burhanuddin Rashid, Mukesh Kumar Swami, Muthukrishnan Rajendran
250	Validación de correo electrónico	Hiren Patel, honk, iravul, Nicolas Maltais
251	VectorDrawable y AnimatedVectorDrawable	Ahmad Aghazadeh, Aleksandar Stefanović, gaara87, honk, Lewis McGeary, RamenChef, Stephen Leppik
252	Versiones de android	4444, AndroidMechanic, athor, BooleanCheese, Dalija Prasnikar, Daniel Nugent, Fildor, Gabriele Mariotti, H. Pauwelyn, Matt, RediOne1, tynn
253	Versiones de Project SDK	Arnav M., Ranveer, Tanis.7x
254	Vibración	cdeange, Zerntrino
255	VideoView	iravul, Sashabrava
256	VideoView optimizado	Chip
257	ViewFlipper	Anita Kunjir, Daniel Nugent, honk
258	ViewPager	Adarsh Ashok, Adrián Pérez, Daniel Nugent, Gabriele Mariotti, Moustachauve, RamenChef, RediOne1, Rucha Bhatt, Sneh Pandya, Stephen Leppik, Usman, ZeroOne
259	Vista de la lista	A.A., brainless, Daniel Nugent, Diti, Douglas Drumond, Fabian Tamp, Gabriele Mariotti, Hiren Patel, Mr.7, Ruben Pirotte, Saeed-rz, shaonAshraf, Squidward
260	Vista de texto	Beena, Daniel Nugent, Eyad Mhanna, gaara87, Gabriele Mariotti, Hiren Patel, honk, keno, Michele, Sohail Zahid, Sujith Niraikulathan, sun-solar-arrow

261	Vista inferior de la navegación	Abdul Wasae, Daniel Nugent, Gabriele Mariotti, guik, Pankaj Kumar, Pratik Butani, Priyank Patel, RamenChef, rciovati, Stephen Leppik, sud007
262	Visualización de anuncios de Google	Egek92, RamenChef, ReverseCold, Stephen Leppik, Zarul Izham
263	Voleo	2943, Ankur Aggarwal, Endzeit, Harsh Dalwadi, herrmartell , honk, Jon Adams, Pablo Baxter, RamenChef, Rubin Nellikunnathu, Rucha Bhatt, sameera lakshitha, Stephen Leppik, VISHWANATH N P
264	WebView	Amod Gokhale, Daniel Nugent, g4s8, j2ko, jasonlam604, JonasCz, Mohammad Yahia, ppeterka, Prakash Bala, shtolik, Squidward, Sukrit Kumar, sukumar
265	Widgets	4444, Alex Ershov, Daniel Nugent, Don Chakkappan, Imdad, nenofite, sun-solar-arrow
266	XMPP registro de inicio de sesión y chat simple ejemplo	4444, RamenChef, Saveen
267	Xposed	Medusalix