EBook Gratuito

APPRENDIMENTO Android

Free unaffiliated eBook created from
Stack Overflow contributors.

#android

Sommario

Di

Capitolo 1: Iniziare con Android

Osservazioni

Se vuoi saperne di più sull'impostazione del plug-in Gradle per Android, consulta la documentazione di `android-gradle` .

Se sei interessato a emulatori alternativi, puoi guardare Genymotion . Fornisce un piano gratuito e richiede una quantità minore di RAM.

Versioni

Versione	Livello API	Codice di versione	Data di rilascio
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	16	`JELLY_BEAN`	2012-07-09
4.2	17	`JELLY_BEAN_MR1`	2012/11/13

Versione	Livello API	Codice di versione	Data di rilascio
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` (Marshmallow)	2015/10/05
7.0	24	`N` (Nougat)	2016/08/22
7.1	25	`N_MR1` (Nougat MR1)	2016/10/04
8.0	26	`O` (Developer Preview 4)	2017/07/24

Examples

Configurazione di Android Studio

Android Studio è l'IDE di sviluppo Android ufficialmente supportato e consigliato da Google. Android Studio viene fornito in bundle con Android SDK Manager , uno strumento per scaricare i componenti di `Android SDK` necessari per avviare lo sviluppo di app.

Installazione degli strumenti Android Studio e `Android SDK` :

1. Scarica e installa Android Studio .
2. Scarica gli ultimi strumenti SDK e gli strumenti della piattaforma SDK aprendo Android Studio e seguendo le istruzioni degli aggiornamenti degli strumenti di Android SDK . Dovresti installare gli ultimi pacchetti stabili disponibili.

   Se è necessario lavorare su vecchi progetti che sono stati creati utilizzando versioni precedenti di SDK, potrebbe essere necessario scaricare anche queste versioni

Dal momento che Android Studio 2.2, una copia dell'ultimo OpenJDK viene fornito in bundle con l'installazione ed è il JDK consigliato (Java Development Kit) per tutti i progetti di Android Studio. Ciò elimina la necessità di installare il pacchetto JDK di Oracle. Per utilizzare l'SDK in dotazione, procedere come segue;

1. Apri il tuo progetto in Android Studio e seleziona **File> Struttura del progetto** nella barra dei menu.
2. Nella pagina **Ubicazione SDK** e nella **posizione JDK** , selezionare la casella di controllo **Usa JDK incorporato** .
3. Clicca **OK** .

Configura Android Studio

Android Studio fornisce l'accesso a due file di configurazione tramite il menu **Guida** :

- studio.vmoptions : personalizza le opzioni per la Java Virtual Machine (JVM) di Studio, come la dimensione dell'heap e la dimensione della cache. Nota che su macchine Linux questo file potrebbe essere chiamato *studio64.vmoptions* , a seconda della versione di Android Studio.
- idea.properties : personalizza le proprietà di Android Studio, come il percorso della cartella dei plug-in o la dimensione massima del file supportato.

Cambia / aggiungi tema

Puoi cambiarlo come preferisci. `File->Settings->Editor->Colors & Fonts->` e seleziona un tema. Puoi anche scaricare nuovi temi da http://color-themes.com/ Dopo aver scaricato il file `.jar.zip` , vai su `File -> Import Settings...` e scegli il file scaricato.

Compilare le app

Crea un nuovo progetto o apri un progetto esistente in Android Studio e premi il pulsante verde Riproduci  sulla barra degli strumenti in alto per eseguirlo. Se è grigio, devi attendere un secondo per consentire ad Android Studio di indicizzare correttamente alcuni file, i cui progressi possono essere visualizzati nella barra di stato in basso.

Se vuoi creare un progetto dalla shell assicurati di avere un file `local.properties` , che viene creato automaticamente da Android Studio. Se hai bisogno di creare il progetto senza Android Studio hai bisogno di una riga che inizi con `sdk.dir=` seguita dal percorso dell'installazione dell'SDK.

Apri una shell e vai nella directory del progetto. Immettere `./gradlew aR` e premere `./gradlew aR` . `aR` è una scorciatoia per `assembleRelease` , che scaricherà tutte le dipendenze per te e costruirà l'app. Il file APK finale sarà in `ProjectName/ModuleName/build/outputs/apk` e sarà chiamato `ModuleName-release.apk` .

Creare un nuovo progetto

Configura Android Studio

Inizia configurando Android Studio e quindi aprilo. Ora sei pronto per creare la tua prima app per Android!

> **Nota:** questa guida è basata su Android Studio 2.2, ma il processo su altre versioni è principalmente lo stesso.

Configura il tuo progetto

Configurazione di base

Puoi iniziare un nuovo progetto in due modi:

- Fai clic su `Start a New Android Studio Project` dalla schermata di benvenuto.
- Passa a `File` → `New Project` se hai già un progetto aperto.

Successivamente, è necessario descrivere la domanda compilando alcuni campi:

1. **Nome applicazione** : questo nome verrà mostrato all'utente.

   Esempio: `Hello World` . Puoi sempre modificarlo in un secondo momento nel file `AndroidManifest.xml` .

2. **Dominio aziendale** : è il qualificatore per il nome del pacchetto del progetto.

   Esempio: `stackoverflow.com` .

3. **Nome pacchetto** (aka `applicationId`): questo è il nome *completo del* pacchetto di progetto.

   Dovrebbe seguire *Reverse Domain Name Notation* (aka *DNS inverso*): *dominio di primo livello* **.** *Dominio dell'azienda* **.** [*Segmento aziendale* **.**] *Nome dell'applicazione* .

   Esempio: `com.stackoverflow.android.helloworld` o `com.stackoverflow.helloworld` . Puoi sempre modificare il tuo ID *applicazione* sovrascrivendolo nel file gradle .

   Non utilizzare il prefisso predefinito "com.example" a meno che tu non intenda inviare la tua richiesta al Google Play Store. Il nome del pacchetto sarà la tua **applicazione** unica in Google Play.

4. **Posizione del progetto** : questa è la directory in cui verrà archiviato il progetto.

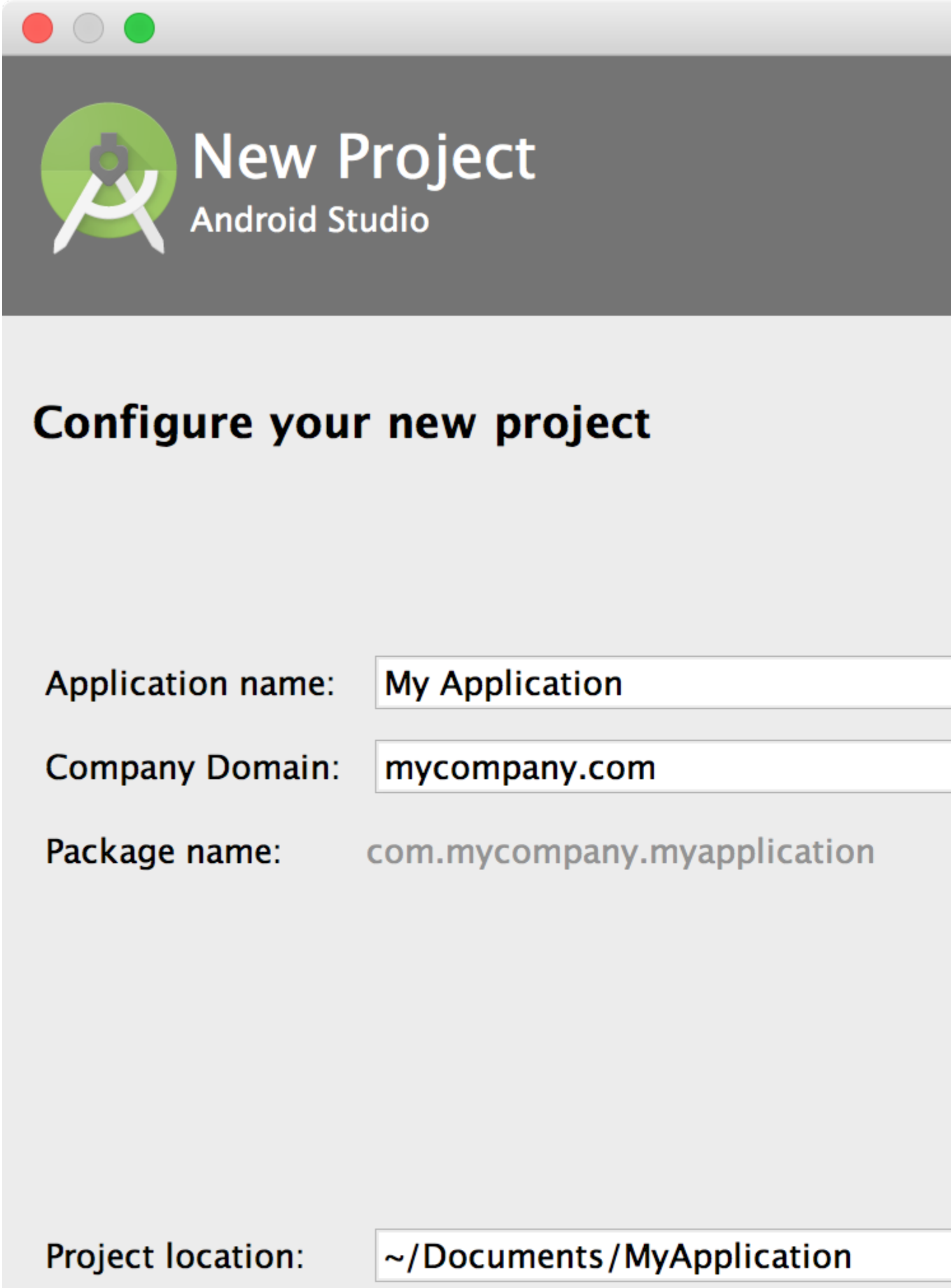
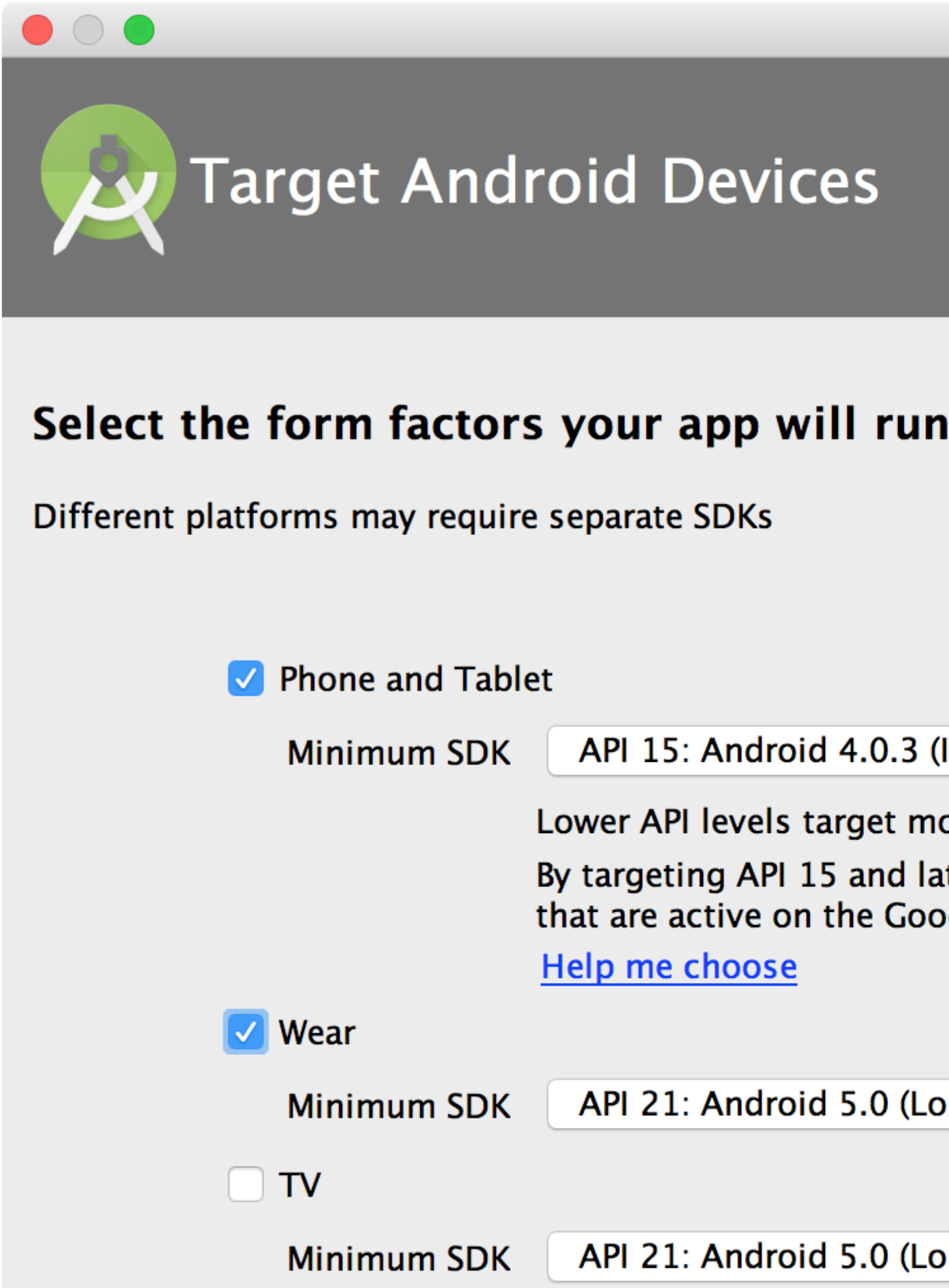

New Project
Android Studio
Configure your new project
Application name:
My Application
Company Domain:
mycompany.com
Package name:
com.mycompany.myapplication
Project location:
~/Documents/MyApplication

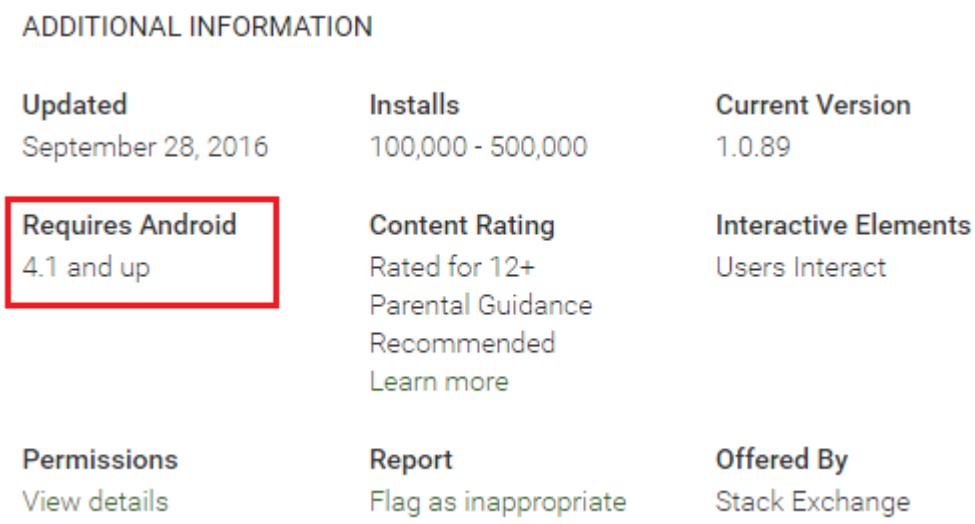
Grafico delle attuali distribuzioni della versione Android, mostrate quando si fa clic su Aiuto me scegliere.

La finestra Distribuzione piattaforma Android mostra la distribuzione dei dispositivi mobili su cui è in esecuzione ciascuna versione di Android, come illustrato nella Figura 2. Fare clic su un livello API per visualizzare un elenco di funzionalità introdotte nella corrispondente versione di Android. Questo ti aiuta a scegliere il livello API minimo che ha tutte le funzionalità necessarie alle tue app, in modo da poter raggiungere il maggior numero di dispositivi possibile. Quindi fare **clic** su **OK** .

Ora, scegli quali piattaforme e versioni di Android SDK supportano l'applicazione.


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

utilizza per determinare su quali dispositivi è possibile installare un'app. Ad esempio, l'app di Stack Exchange supporta Android 4.1+.



Android Studio indicherà (approssimativamente) la percentuale di dispositivi supportati in base all'SDK minimo specificato.

> Livelli di API inferiori mirano a più dispositivi ma hanno meno funzioni disponibili.

Al momento di decidere l' **SDK minimo** , dovresti prendere in considerazione le statistiche di Dashboard , che ti forniranno informazioni sulla versione dei dispositivi che hanno visitato il Google Play Store a livello globale nell'ultima settimana.

Da: Dashboard sul sito web degli sviluppatori Android.

Aggiungi un'attività

Ora selezioneremo un'attività predefinita per la nostra applicazione. In Android, un `Activity` è un unico schermo che sarà presentato per l'utente. Un'applicazione può ospitare più attività e navigare tra di loro. Per questo esempio, selezionare `Empty Activity` e fare clic su next.

Qui, se lo desideri, puoi cambiare il nome dell'attività e il layout. Una buona pratica è mantenere l' `Activity` come suffisso per il nome `activity_` e `activity_` come prefisso per il nome del layout. Se li

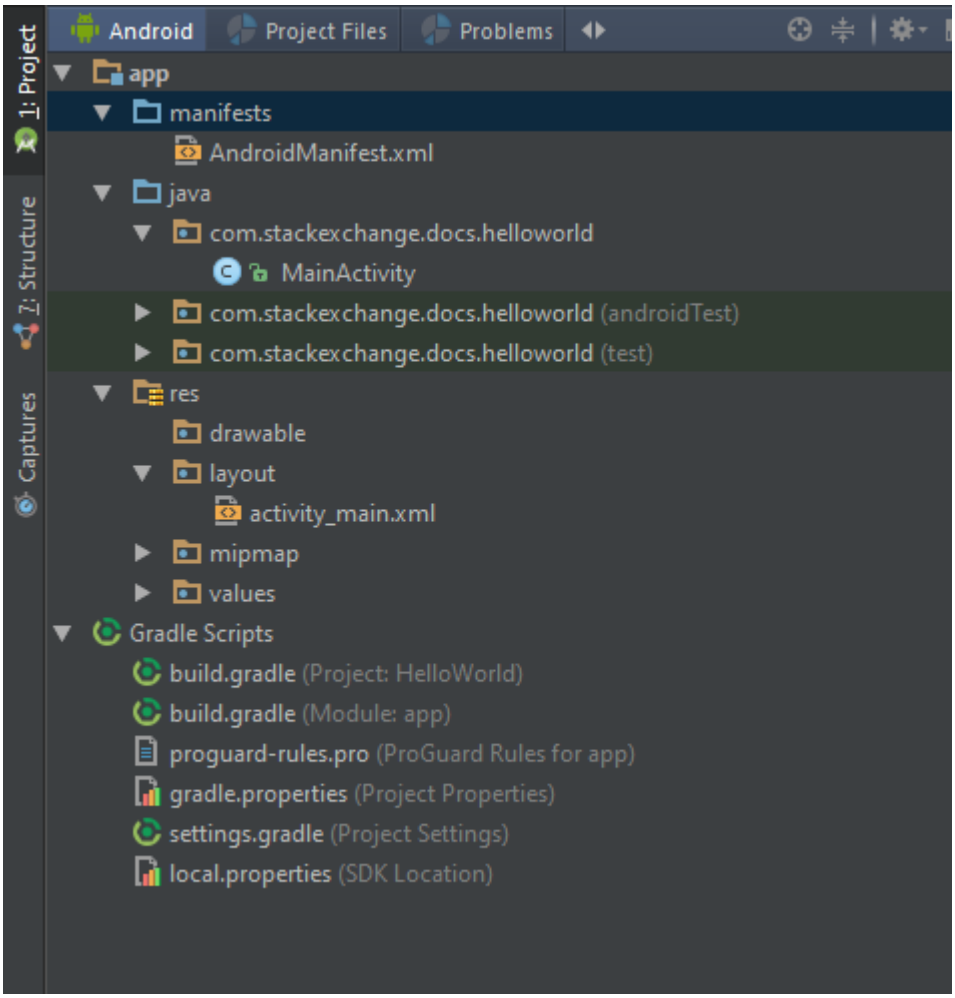
lasciamo come predefiniti, Android Studio genererà un'attività per noi chiamata `MainActivity` e un file di layout chiamato `activity_main` . Ora fai clic su `Finish` .

Android Studio creerà e configurerà il nostro progetto, che può richiedere del tempo a seconda del sistema.

Ispezionando il progetto

Per capire come funziona Android, diamo un'occhiata ad alcuni dei file che sono stati creati per noi.

Nel riquadro sinistro di Android Studio, possiamo vedere la struttura della nostra applicazione Android .



Per prima cosa, apriamo `AndroidManifest.xml` facendo doppio clic su di esso. Il file manifest di Android descrive alcune delle informazioni di base su un'applicazione Android. Contiene la dichiarazione delle nostre attività e alcuni componenti più avanzati.

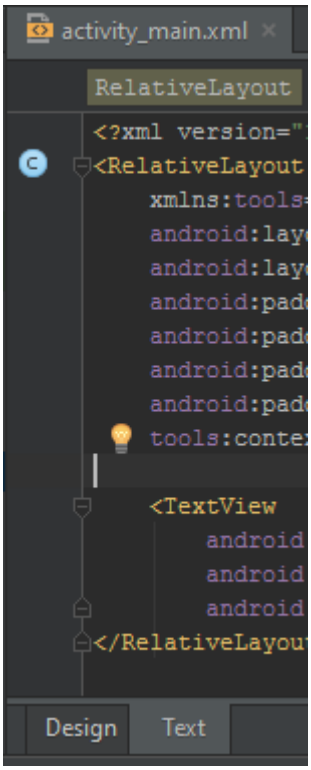
Se un'applicazione richiede l'accesso a una funzione protetta da un'autorizzazione, deve dichiarare che richiede tale autorizzazione con un elemento `<uses-permission>` nel manifest. Quindi, quando l'applicazione è installata sul dispositivo, il programma di installazione determina se concedere o meno l'autorizzazione richiesta controllando le autorità che hanno firmato i

certificati dell'applicazione e, in alcuni casi, chiedendo all'utente. Un'applicazione può inoltre proteggere i propri componenti (attività, servizi, ricevitori di trasmissione e fornitori di contenuti) con autorizzazioni. Può utilizzare qualsiasi autorizzazione definita da Android (elencata in android.Manifest.permission) o dichiarata da altre applicazioni. O può definire il suo.

```
<?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>
```

Quindi, apriamo `activity_main.xml` che si trova in `app/src/main/res/layout/` . Questo file contiene dichiarazioni per i componenti visivi della nostra MainActivity. Vedrai visual designer. Ciò consente di trascinare e rilasciare elementi sul layout selezionato.

Puoi anche passare al designer di layout xml facendo clic su "Testo" nella parte inferiore di Android Studio, come mostrato qui:



```
<?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>
```

Vedrai un widget chiamato `TextView` all'interno di questo layout, con la proprietà `android:text` impostata su "Hello World!". Questo è un blocco di testo che verrà mostrato all'utente quando esegue l'applicazione.

Puoi leggere di più su Layout e attributi .

Quindi, diamo un'occhiata a `MainActivity` . Questo è il codice Java che è stato generato per `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);
    }
}
```

Come definito nel nostro manifest Android, `MainActivity` verrà avviato di default quando un utente avvia l'app `HelloWorld` .

Infine, apri il file denominato `build.gradle` situato in `app/` .
Android Studio utilizza il sistema di generazione **Gradle** per compilare e creare applicazioni e librerie 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'
}
```

Questo file contiene informazioni sulla build e sulla versione dell'app e puoi anche utilizzarlo per aggiungere dipendenze a librerie esterne. Per ora, non apportare modifiche.

Si consiglia di selezionare sempre l'ultima versione disponibile per le dipendenze:

- buildToolsVersion : 26.0.0
- com.android.support:appcompat-v7 : 26.0.0 (luglio 2017)
- Firebase : 11.0.4 (agosto 2017)

compileSdkVersion

`compileSdkVersion` è il modo per dire a *Gradle* quale versione dell'SDK di Android compilare la tua app. L'utilizzo del nuovo SDK Android è un requisito per l'utilizzo di una qualsiasi delle nuove API aggiunte in tale livello.

Va sottolineato che la modifica di `compileSdkVersion` non modifica il comportamento di runtime. Mentre possono essere presenti nuovi avvisi / errori del compilatore quando si modifica `compileSdkVersion` , `compileSdkVersion` non è incluso nel tuo APK: è puramente utilizzato in fase di compilazione.

Pertanto, si consiglia vivamente di compilare sempre l'ultimo SDK. Otterrai tutti i vantaggi dei nuovi controlli di compilazione sul codice esistente, eviterà le API appena deprecate e sarai pronto a utilizzare le nuove API.

minSdkVersion

Se `compileSdkVersion` imposta le nuove API disponibili, `minSdkVersion` è il *limite inferiore* per la tua

app. `minSdkVersion` è uno dei segnali che Google Play Store utilizza per determinare su quale dispositivo di un utente può essere installata un'app.

Svolge anche un ruolo importante durante lo sviluppo: per impostazione predefinita, lint viene eseguito contro il tuo progetto, avvisandoti quando usi qualsiasi API sopra la tua `minSdkVersion` , aiutandoti ad evitare il problema del runtime di tentare di chiamare un'API che non esiste. Il controllo della versione del sistema in fase di esecuzione è una tecnica comune quando si utilizzano le API solo su versioni di piattaforme più recenti.

targetSdkVersion

`targetSdkVersion` è il modo principale in cui Android offre la compatibilità `targetSdkVersion` non applicando le modifiche al comportamento a meno che la versione `targetSdkVersion` venga aggiornata. Ciò consente di utilizzare nuove API prima di utilizzare le modifiche al comportamento. L'aggiornamento a target dell'ultimo SDK dovrebbe essere una priorità elevata per ogni app. Ciò non significa che devi usare ogni nuova funzionalità introdotta, né dovresti aggiornare ciecamente la tua versione di `targetSdkVersion` senza test.

targetSDKVersion è la versione di Android che è il limite superiore per gli strumenti disponibili. Se targetSDKVersion è inferiore a 23, l'app non deve richiedere autorizzazioni in fase di esecuzione per un'istanza, anche se l'app viene eseguita su API 23+. TargetSDKVersion **non** impedisce alle versioni Android sopra la versione Android selezionata di eseguire l'app.

Puoi trovare maggiori informazioni sul plugin Gradle:

- Un esempio di base
- Introduzione al plugin Gradle per Android e il wrapper
- Introduzione alla configurazione di build.gradle e dei metodi DSL

Esecuzione dell'applicazione

Ora, eseguiamo la nostra applicazione HelloWorld. È possibile eseguire un dispositivo virtuale Android (che è possibile impostare utilizzando AVD Manager in Android Studio, come descritto nell'esempio seguente) o collegare il proprio dispositivo Android tramite un cavo USB.

Configurazione di un dispositivo Android

Per eseguire un'applicazione da Android Studio sul tuo dispositivo Android, devi abilitare `USB Debugging` nelle `Developer Options` nelle impostazioni del tuo dispositivo.

`Settings > Developer options > USB debugging`

Se le `Developer Options` non sono visibili nelle impostazioni, vai a `About Phone` e tocca il `Build Number` sette volte. Ciò consentirà alle `Developer Options` di apparire nelle tue impostazioni.

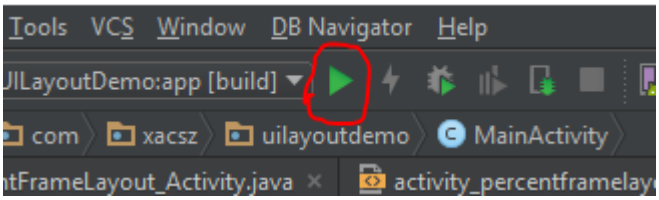
`Settings > About phone > Build number`

Potrebbe anche essere necessario modificare la configurazione di `build.gradle` per creare una

versione del dispositivo.

Esecuzione da Android Studio

Fai clic sul pulsante verde `Run` dalla barra degli strumenti nella parte superiore di Android Studio. Nella finestra visualizzata, seleziona il dispositivo su cui desideri eseguire l'applicazione (avvia un dispositivo virtuale Android, se necessario, oppure consulta Impostazione di un dispositivo virtuale Android (Android Device) se è necessario configurarne uno) e fai `OK` su `OK` .



Sui dispositivi con Android 4.4 (KitKat) e possibilmente più in alto, verrà mostrato un pop-up per autorizzare il debug USB. Fare `OK` su `OK` per accettare.

L'applicazione ora verrà installata ed eseguita sul tuo dispositivo o emulatore Android.

Posizione del file APK

Quando si prepara la domanda per la versione, si configura, si crea e si verifica una versione di rilascio dell'applicazione. Le attività di configurazione sono semplici e comportano operazioni di pulizia del codice di base e di modifica del codice che consentono di ottimizzare l'applicazione. Il processo di compilazione è simile al processo di compilazione di debug e può essere eseguito utilizzando gli strumenti JDK e Android SDK. Le attività di test servono come controllo finale, assicurando che l'applicazione funzioni come previsto in condizioni reali. Quando hai finito di preparare la tua domanda di rilascio, hai un file APK firmato, che puoi distribuire direttamente agli utenti o distribuire attraverso un marketplace di applicazioni come Google Play.

Studio Android

Poiché negli esempi precedenti viene utilizzato Gradle, la posizione del file APK generato è: `<Your Project Location>/app/build/outputs/apk/app-debug.apk`

IntelliJ

Se sei un utente di IntelliJ prima di passare a Studio e stai importando direttamente il tuo progetto IntelliJ, non è cambiato nulla. La posizione dell'output sarà la stessa sotto:

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

Nota: questo è diventato deprecato a volte intorno a 1.0

Eclisse

Se stai importando direttamente il progetto Android Eclipse, non farlo! Non appena avrai delle

dipendenze nel tuo progetto (vasi o progetti di biblioteca), questo non funzionerà e il tuo progetto non sarà configurato correttamente. Se non hai dipendenze, l'apk si trova nella stessa posizione in cui lo troverai in Eclipse:

```
bin/...
```

Programmazione Android senza IDE

Questo è un esempio di Hello World minimalista che utilizza solo gli strumenti Android più basilari.

Requisiti e ipotesi

- Oracle JDK 1.7 o successivo
- Strumenti Android SDK (solo gli strumenti da riga di comando)

Questo esempio presuppone Linux. Potrebbe essere necessario regolare la sintassi per la propria piattaforma.

Impostazione dell'SDK Android

Dopo aver disimballato la versione dell'SDK:

1. Installa pacchetti aggiuntivi usando il gestore SDK. Non utilizzare `android update sdk --no-ui` come indicato nel file Readme.txt; scarica circa 30 GB di file non necessari. Usa invece il gestore SDK interattivo di `android sdk` per ottenere il minimo raccomandato di pacchetti.

2. Aggiungi le seguenti directory JDK e SDK al tuo PATH di esecuzione. Questo è opzionale, ma le istruzioni seguenti lo assumono.

    - JDK / bin
    - SDK / platform-tools
    - SDK / strumenti
    - SDK / build-tools / LATEST *(come installato nel passaggio 1)*

3. Crea un dispositivo virtuale Android. Utilizzare l'AVD Manager interattivo (`android avd`). Dovresti giocherellare un po 'e cercare consigli; le istruzioni in loco non sono sempre utili.

    (Puoi anche usare il tuo dispositivo)

4. Esegui il dispositivo:

    ```
    emulator -avd DEVICE
    ```

5. Se la schermata del dispositivo sembra bloccata, fai scorrere per sbloccarla.

    Lasciarlo in esecuzione mentre codifichi l'app.

Codifica l'app

6. Passare a una directory di lavoro vuota.

7. Crea il file sorgente:

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

   Soddisfare:

```
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. Aggiungi un manifest:

```
touch AndroidManifest.xml
```

   Soddisfare:

```
<?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. Crea una sottodirectory per le risorse dichiarate:

```
mkdir res
```

   Lascialo vuoto per ora.

Costruire il codice

10. Genera l'origine per le dichiarazioni di risorse. Sostituisci qui il percorso corretto per il tuo **SDK** e l' **API** installata da costruire contro (ad esempio "android-23"):

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

    Le dichiarazioni delle risorse (descritte più avanti di seguito) sono in realtà facoltative. Nel frattempo la chiamata sopra non fa nulla se res / è ancora vuoto.

11. Compilare il codice sorgente in bytecode Java (.java → .class):

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

12. Tradurre il bytecode da Java ad Android (.class → .dex):

    Innanzitutto usando Jill (.class → .jayce):

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

    Quindi Jack (.jayce → .dex):

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

    Il bytecode di Android era chiamato "codice eseguibile Dalvik" e quindi "dex".

    Puoi sostituire i passaggi 11 e 12 con una sola chiamata a Jack, se lo desideri; può compilare direttamente dal sorgente Java (.java → .dex). Ma ci sono dei vantaggi nella compilazione con `javac` . È uno strumento più conosciuto, meglio documentato e più ampiamente applicabile.

13. Imballare i file di risorse, incluso il manifest:

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

    Ciò si traduce in un file APK parziale (pacchetto di applicazioni Android).

14. Crea l'APK completo utilizzando lo strumento `ApkBuilder` :

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

Si avvisa, "QUESTO STRUMENTO È DEPRECATO. Vedi --help per ulteriori informazioni." Se `--help` fallisce con una `ArrayIndexOutOfBoundsException` , allora non passa alcun argomento:

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

Spiega che la CLI (`ApkBuilderMain`) è deprecata a favore di chiamare direttamente l'API Java (`ApkBuilder`). (Se sai come farlo dalla riga di comando, ti preghiamo di aggiornare questo esempio.)

15. Ottimizza l'allineamento dei dati dell'APK (pratica consigliata):

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

Installazione e funzionamento

16. Installa l'app sul dispositivo Android:

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

17. Avvia l'app:

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

Dovrebbe correre e salutare.

È tutto. Questo è quello che serve per dire ciao usando gli strumenti base di Android.

Dichiarare una risorsa

Questa sezione è facoltativa. Le dichiarazioni delle risorse non sono richieste per una semplice app "ciao mondo". Se non sono necessari nemmeno per la tua app, puoi semplificare la compilazione della versione omettendo il punto 10 e rimuovendo il riferimento alla directory res / dal punto 13.

Altrimenti, ecco un breve esempio di come dichiarare una risorsa e come farla riferimento.

18. Aggiungi un file di risorse:

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

Soddisfare:

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

19. Fare riferimento alla risorsa dal manifest XML. Questo è uno stile di riferimento dichiarativo:

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

20. Fare riferimento alla stessa risorsa dal codice sorgente Java. Questo è un riferimento imperativo:

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

21. Provare le modifiche precedenti ricostruendo, reinstallando e rieseguendo l'applicazione (passaggi 10-17).

    Dovrebbe riavviare e dire "Questa app si chiama Dire ciao".

Disinstallare l'app

```
adb uninstall dom.domain
```

Guarda anche

- domanda originale - La domanda originale che ha richiesto questo esempio
- esempio di lavoro : uno script di compilazione funzionante che utilizza i comandi precedenti

Fondamenti applicativi

Le app Android sono scritte in Java. Gli strumenti di Android SDK compilano il codice, i dati e i file di risorse in un APK (pacchetto Android). Generalmente, un file APK contiene tutto il contenuto dell'app.

Ogni app viene eseguita sulla propria macchina virtuale (VM) in modo che l'app possa essere isolata da altre app. Il sistema Android funziona con il principio del minimo privilegio. Ogni app ha solo accesso ai componenti necessari per svolgere il proprio lavoro, e non di più. Tuttavia, ci sono modi per un'app di condividere dati con altre app, ad esempio condividendo l'ID utente di Linux tra

le app, oppure le app possono richiedere l'autorizzazione per accedere ai dati del dispositivo come scheda SD, contatti ecc.

Componenti dell'app

I componenti dell'app sono gli elementi costitutivi di un'app Android. Ogni componente svolge un ruolo specifico in un'app Android che ha uno scopo ben preciso e ha cicli di vita distinti (il flusso di come e quando il componente viene creato e distrutto). Ecco i quattro tipi di componenti dell'app:

1. **Attività:** un'attività rappresenta una singola schermata con un'interfaccia utente (UI). Un'app Android può avere più di un'attività. (ad esempio un'app di posta elettronica potrebbe avere un'attività per elencare tutte le e-mail, un'altra per mostrare il contenuto di ciascuna e-mail e un'altra per comporre una nuova e-mail.) Tutte le attività in un'app interagiscono per creare un utente eXperience (UX).
2. **Servizi:** un servizio viene eseguito in background per eseguire operazioni di lunga durata o per eseguire lavori per processi remoti. Un servizio non fornisce alcuna interfaccia utente, viene eseguito solo in background con l'input dell'utente. (ad esempio un servizio può riprodurre musica in sottofondo mentre l'utente si trova in un'altra App, o potrebbe scaricare dati da Internet senza bloccare l'interazione dell'utente con il dispositivo Android.)
3. **Content Provider:** un fornitore di contenuti gestisce i dati delle app condivise. Esistono quattro modi per archiviare i dati in un'app: possono essere scritti su un file e memorizzati nel file system, inseriti o aggiornati in un database SQLite, pubblicati sul Web o salvati in qualsiasi altra posizione di archiviazione permanente a cui l'app può accedere . Attraverso i fornitori di contenuti, altre app possono interrogare o persino modificare i dati. (ad esempio, il sistema Android fornisce un fornitore di contenuti che gestisce le informazioni di contatto dell'utente in modo che qualsiasi app che disponga dell'autorizzazione possa interrogare i contatti.) I provider di contenuti possono anche essere utilizzati per salvare i dati privati dell'applicazione per una migliore integrità dei dati.
4. **Ricevitori di trasmissione:** un ricevitore di trasmissione risponde alle trasmissioni di annunci a livello di sistema (ad esempio una trasmissione che annuncia lo spegnimento dello schermo, la batteria è scarica, ecc.) O dalle app (ad esempio per far sapere alle altre app che alcuni dati sono stati scaricato sul dispositivo ed è disponibile per l'uso). I ricevitori broadcast non dispongono di interfacce utente ma possono mostrare notifiche nella barra di stato per avvisare l'utente. Solitamente i ricevitori di trasmissione vengono utilizzati come gateway per altri componenti dell'app, costituiti principalmente da attività e servizi.

Un aspetto unico del sistema Android è che ogni app può avviare un componente di un'altra app (ad esempio, se si desidera effettuare chiamate, inviare SMS, aprire una pagina Web o visualizzare una foto, c'è un'app che già lo fa e l'app può farne uso, invece di sviluppare una nuova attività per lo stesso compito).

Quando il sistema avvia un componente, avvia il processo per quell'app (se non è già in esecuzione, ovvero solo un processo foreground per app può essere eseguito in un dato momento su un sistema Android) e crea un'istanza delle classi necessarie per quel componente. Pertanto il componente viene eseguito sul processo di quell'app a cui appartiene. Pertanto, a differenza delle app su altri sistemi, le app Android non hanno un singolo punto di ingresso (non esiste un metodo `main()`).

Poiché il sistema esegue ciascuna app in un processo separato, un'app non può attivare direttamente i componenti di un'altra app, tuttavia il sistema Android può. Pertanto, per avviare un componente di un'altra app, un'app deve inviare un messaggio al sistema che specifica l'intenzione di avviare quel componente, quindi il sistema avvierà quel componente.

Contesto

Le istanze della classe `android.content.Context` forniscono la connessione al sistema Android che esegue l'applicazione. È necessaria l'istanza del contesto per ottenere l'accesso alle risorse del progetto e alle informazioni globali sull'ambiente dell'app.

Facciamo un esempio facile da digerire: considera di essere in un hotel e vuoi mangiare qualcosa. Chiami il servizio in camera e chiedi loro di portarti qualcosa o ripulire le cose per te. Ora pensa a questo hotel come a un'app per Android, a te stesso come a un'attività, e la persona del servizio in camera è quindi il tuo contesto, che ti consente di accedere alle risorse dell'hotel come servizio in camera, cibo, ecc.

Ancora un altro esempio: sei in un ristorante seduto su un tavolo, ogni tavolo ha un accompagnatore, quando mai vuoi ordinare prodotti alimentari chiedi all'addetto di farlo. L'addetto quindi effettua l'ordine e i prodotti alimentari vengono serviti sul tavolo. Anche in questo esempio, il ristorante è un'app per Android, i tavoli o i clienti sono componenti App, gli articoli di cibo sono le tue risorse App e l'addetto è il tuo contesto, dandoti così un modo per accedere alle risorse come gli alimenti.

L'attivazione di uno dei componenti sopra richiede l'istanza del contesto. Non solo solo sopra, ma quasi tutte le risorse di sistema: creazione dell'interfaccia utente che utilizza le viste (discusse in seguito), creazione di istanze di servizi di sistema, avvio di nuove attività o servizi: tutto richiede un contesto.

Una descrizione più dettagliata è scritta qui .

Configurazione di un AVD (dispositivo virtuale Android)

TL; DR Fondamentalmente ci consente di simulare dispositivi reali e testare le nostre app senza un dispositivo reale.

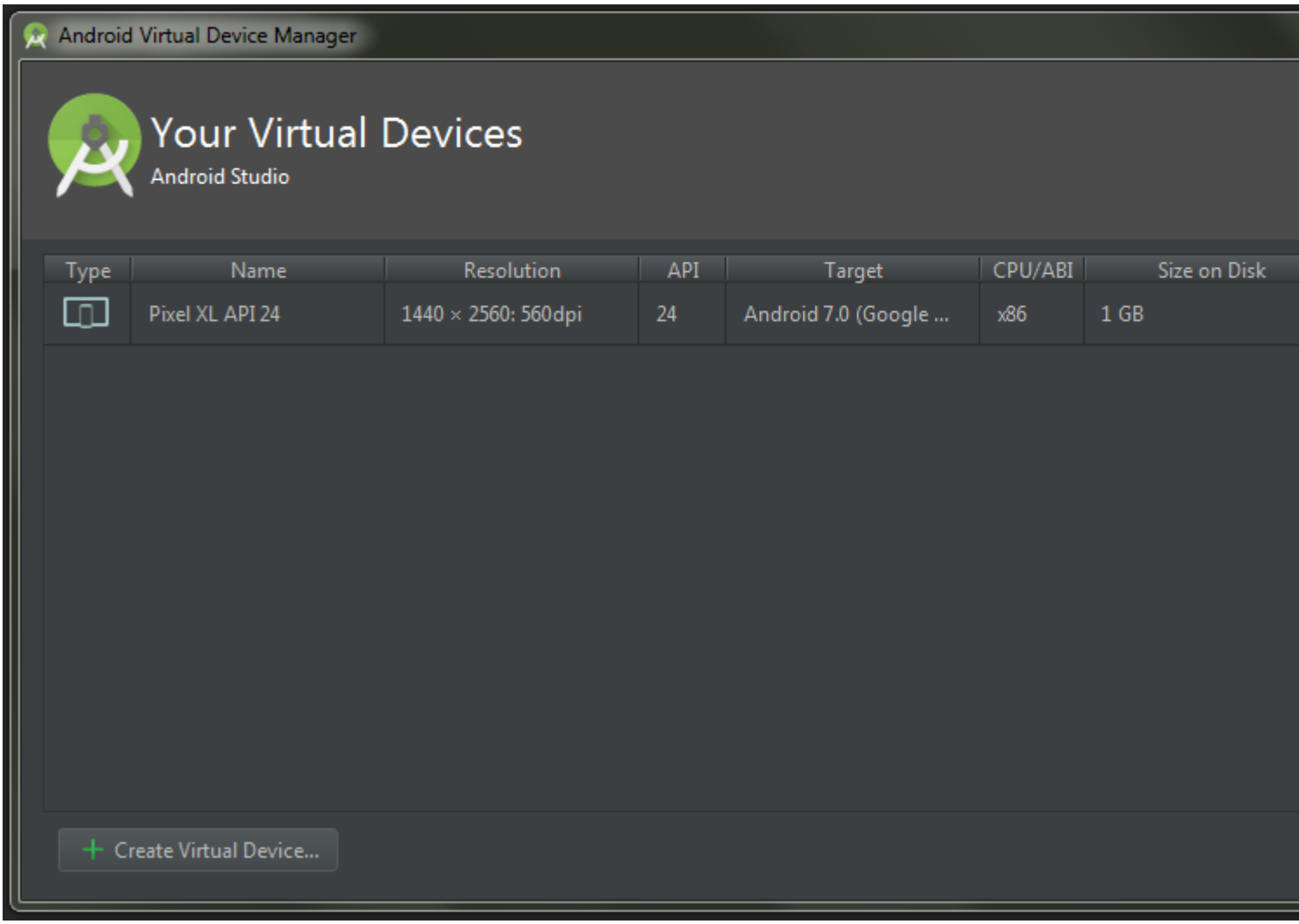
Secondo la documentazione per gli sviluppatori Android ,

> una *definizione di* ***dispositivo virtuale Android (AVD)*** consente di definire le caratteristiche di un dispositivo Android Phone, Tablet, Android Wear o Android TV che si desidera simulare nell'emulatore Android. AVD Manager ti aiuta a creare e gestire facilmente gli AVD.

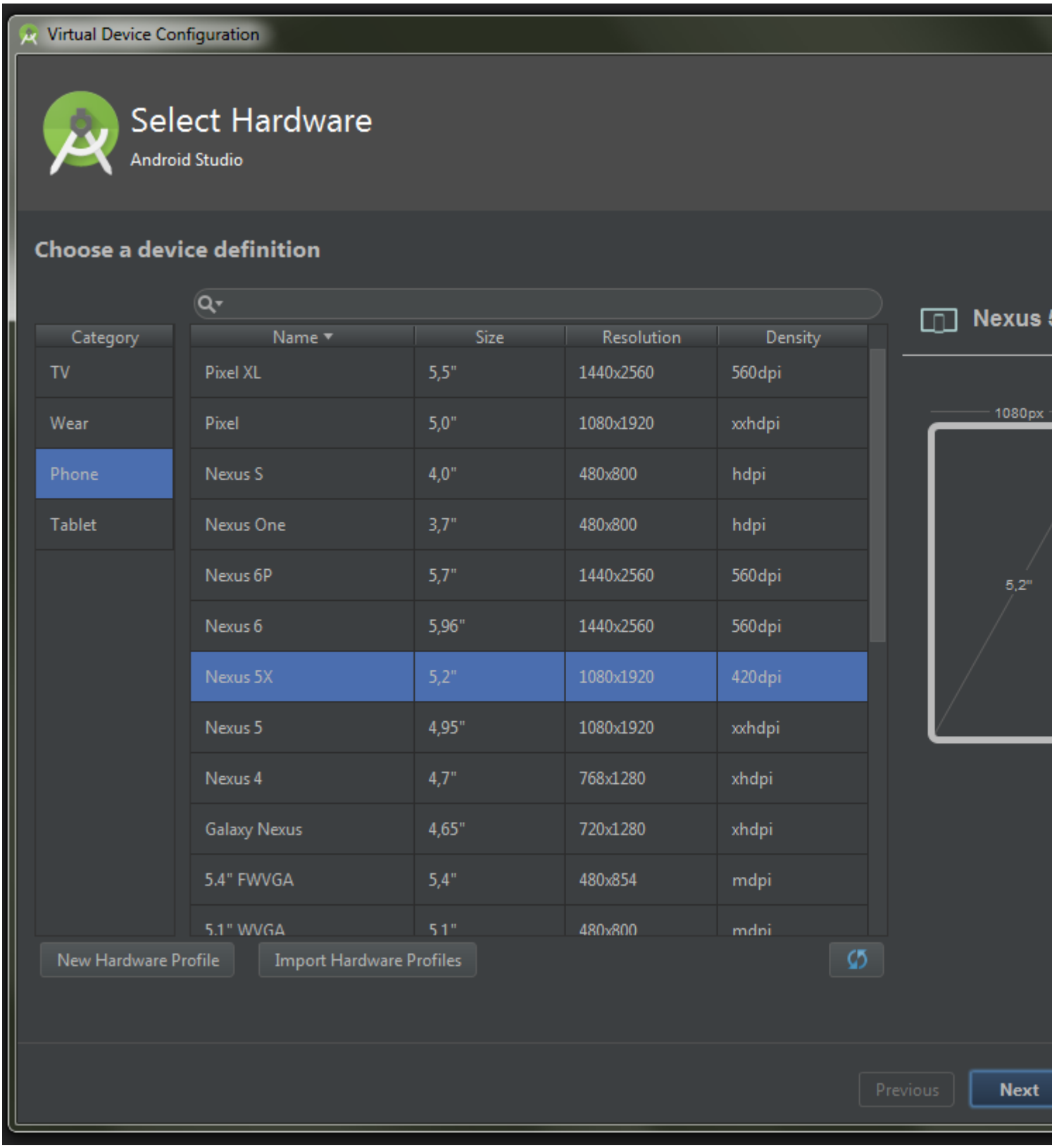
Per configurare un AVD, attenersi alla seguente procedura:

1. Fare clic su questo pulsante per visualizzare AVD Manager:

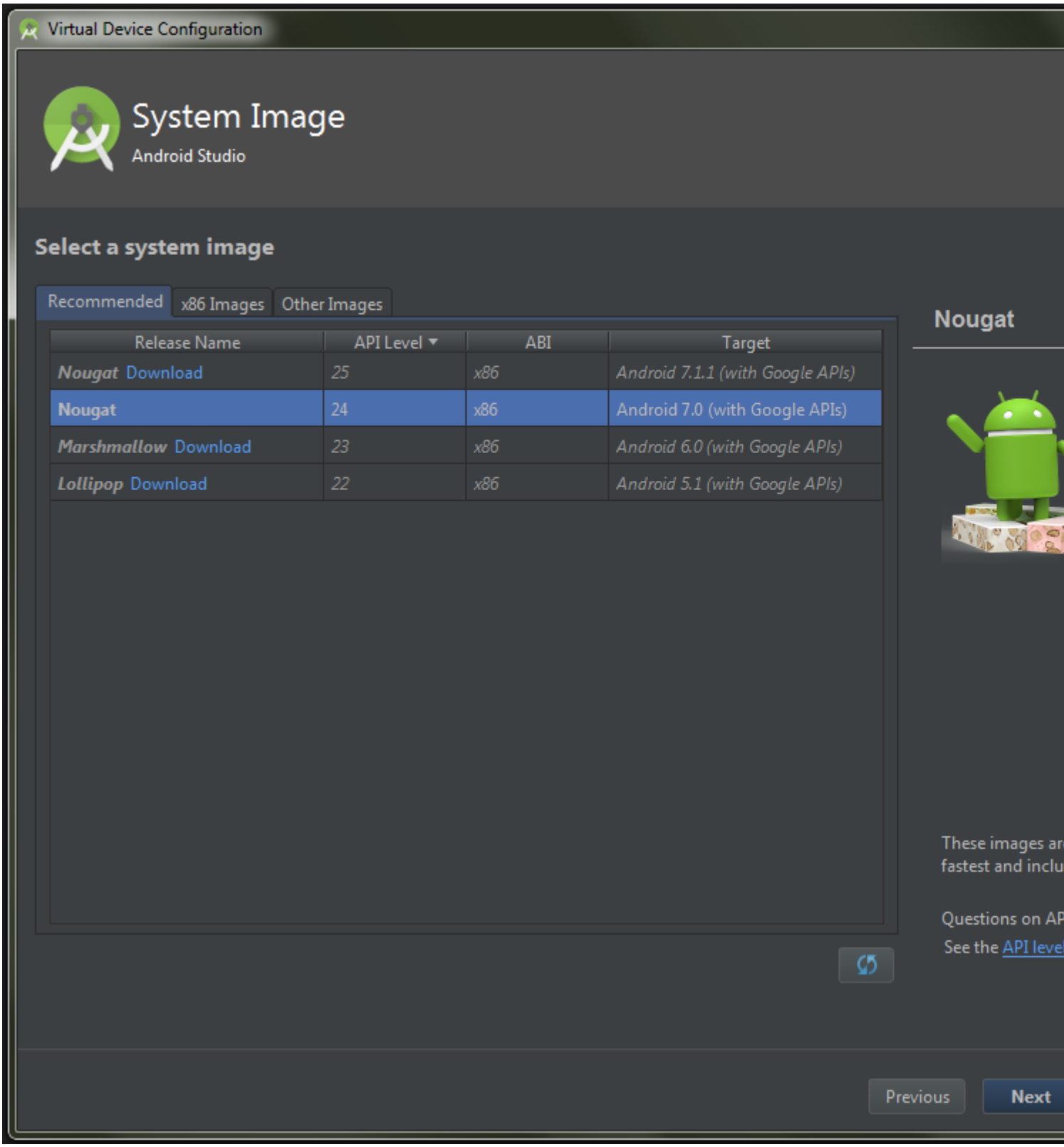
2. Dovresti vedere una finestra di dialogo come questa:



3. Ora fai clic sul pulsante `+ Create Virtual Device...` Questo farà apparire la finestra di dialogo di configurazione del dispositivo virtuale:



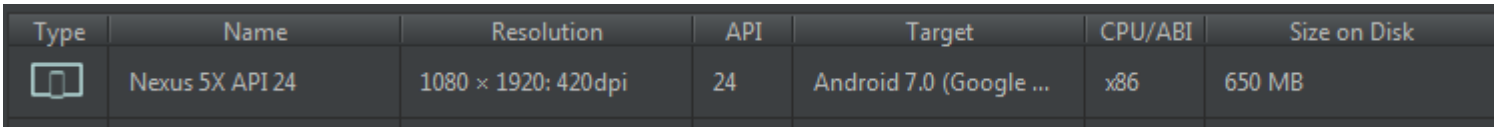
4. Selezionare qualsiasi dispositivo desiderato, quindi fare clic su `Next` :



5. Qui devi scegliere una versione per Android per il tuo emulatore. Potrebbe anche essere necessario scaricarlo prima facendo clic su `Download` . Dopo aver scelto una versione, fare clic su `Next` .

6. Qui, inserisci un nome per il tuo emulatore, l'orientamento iniziale e se vuoi visualizzare una cornice attorno ad esso. Dopo aver scelto tutti questi, fare clic su `Finish` .

7. Ora hai un nuovo AVD pronto per lanciare le tue app su di esso.



Leggi Iniziare con Android online: https://riptutorial.com/it/android/topic/85/iniziare-con-android

Capitolo 2: Accesso ai database SQLite utilizzando la classe ContentValues

Examples

Inserimento e aggiornamento di righe in un database SQLite

Innanzitutto, è necessario aprire il database SQLite, che può essere eseguito come segue:

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

Dopo aver aperto il database, puoi facilmente inserire o aggiornare le righe utilizzando la classe `ContentValues` . I seguenti esempi presumono che un nome sia dato da `str_edtfname` e un ultimo nome da `str_edtlname` . È inoltre necessario sostituire `table_name` con il nome della tabella che si desidera modificare.

Inserimento di dati

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

Aggiornamento dei dati

```
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});
```

Leggi Accesso ai database SQLite utilizzando la classe ContentValues online: https://riptutorial.com/it/android/topic/10154/accesso-ai-database-sqlite-utilizzando-la-classe-contentvalues

Capitolo 3: Account e AccountManager

Examples

Comprendere account / autenticazione personalizzati

L'esempio seguente è una copertura di alto livello dei concetti chiave e della configurazione scheletrica di base:

1. Raccoglie le credenziali dall'utente (solitamente da una schermata di accesso che hai creato)
2. Autentica le credenziali con il server (memorizza l'autenticazione personalizzata)
3. Memorizza le credenziali sul dispositivo

Estendi un abstractAccountAuthenticator *(utilizzato principalmente per recuperare l'autenticazione e riattivarli)*

```
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.
  }
}
```

Creare un servizio *(il framework di Account Manager si connette al AbstractAccountAuthenticator esteso tramite l'interfaccia del servizio)*

```
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();
    }
}
```

Configurazione XML dell'autenticatore *(il framework del gestore dell'account richiede. Questo è ciò che vedrai in Impostazioni -> Account in 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" />
```

Modifiche a AndroidManifest.xml *(riunisci tutti i concetti di cui sopra per renderlo utilizzabile programmaticamente tramite l'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>
```

Il prossimo esempio conterrà come utilizzare questa impostazione.

Leggi Account e AccountManager online: https://riptutorial.com/it/android/topic/7003/account-e-accountmanager

Capitolo 4: ACRA

Sintassi

- android: name = "ACRAHandler"
- ACRA.init (questo, config);
- la classe pubblica ACRAHandler estende l'applicazione {

Parametri

Parametro	Descrizione
@ReportCrashes	Definisce le impostazioni ACRA come dove deve essere segnalato, contenuto personalizzato, ecc
formUri	il percorso del file che riporta l'arresto anomalo

Osservazioni

- ACRA non supporta più i moduli di Google, quindi è necessario un back-end: https://github.com/ACRA/acra/wiki/Backends

Examples

ACRAHandler

Esempio di classe che estende l'applicazione per la gestione dei rapporti:

```
@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);


    }

}
```

Esempio manifest

```
<?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>
```

Installazione

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'
```

Leggi ACRA online: https://riptutorial.com/it/android/topic/1324/acra

Capitolo 5: ADB (Android Debug Bridge)

introduzione

ADB (Android Debug Bridge) è uno strumento da riga di comando che consente di comunicare con un'istanza dell'emulatore o un dispositivo Android connesso.

Panoramica di ADB

Una gran parte di questo argomento è stata suddivisa in adb shell

Osservazioni

Elenco di esempi spostati nella shell adb :

- Concessione e revoca delle autorizzazioni API 23+
- Invia testo, tasto premuto e tocca gli eventi sul dispositivo Android tramite ADB
- Elenca i pacchetti
- Registrazione del display
- Apri Opzioni sviluppatore
- Imposta data / ora tramite adb
- Modifica dei permessi dei file usando il comando chmod
- Generazione di una trasmissione "Boot Complete"
- Stampa i dati dell'applicazione
- Visualizza contenuto di archiviazione esterno / secondario
- http://stackoverflow.com/documentation/android/9408/adb-shell/29140/adb-shell
- uccidere un processo all'interno di un dispositivo Android

Examples

Stampa elenco dettagliato dei dispositivi collegati

Per ottenere un elenco dettagliato di tutti i dispositivi collegati ad `adb` , scrivi il seguente comando nel tuo terminale:

```
adb devices -l
```

Esempio di output

```
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 prima colonna è il numero di serie del dispositivo. Se inizia con l' `emulator-` , questo dispositivo è un emulatore.
- `usb:` il percorso del dispositivo nel sottosistema USB.
- `product:` il codice prodotto del dispositivo. Questo è molto specifico del produttore e, come puoi vedere nel caso del dispositivo Archos `A50PL` , può essere vuoto.
- `model:` il `model:` del dispositivo. Come il `product` , può essere vuoto.
- `device:` il codice del dispositivo. Questo è anche molto specifico del produttore e può essere vuoto.

Leggi le informazioni sul dispositivo

Scrivi il seguente comando nel tuo terminale:

```
adb shell getprop
```

Questo stamperà tutte le informazioni disponibili sotto forma di coppie chiave / valore.

Puoi solo leggere informazioni specifiche aggiungendo il nome di una chiave specifica al comando. Per esempio:

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

Ecco alcune informazioni interessanti che ottieni:

- `ro.product.model` : nome del modello del dispositivo (ad es. Nexus 6P)
- `ro.build.version.sdk` : Livello API del dispositivo (ad es. 23)
- `ro.product.brand` : marchio del dispositivo (ad esempio Samsung)

Pieno esempio di output

```
[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]
```

Collega ADB a un dispositivo tramite WiFi

La configurazione ADB standard prevede una connessione USB a un dispositivo fisico.
Se si preferisce, è possibile passare alla modalità TCP / IP e collegare invece ADB tramite WiFi.

Dispositivo non rootato

1. **Entra nella stessa rete:**
    - Assicurati che il tuo dispositivo e il tuo computer siano sulla stessa rete.
2. **Collegare il dispositivo al computer host con un cavo USB.**
3. **Collega `adb` al dispositivo tramite rete:**

    Mentre il dispositivo è connesso ad `adb` via USB, eseguire il seguente comando per ascoltare una connessione TCP / IP su una porta (predefinito 5555):

    - Digitare `adb tcpip <port>` (passare alla modalità TCP / IP).
    - Scollegare il cavo USB dal dispositivo di destinazione.
    - Digitare `adb connect <ip address>:<port>` (la porta è facoltativa; predefinita 5555).

    Per esempio:

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

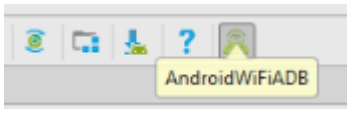
    Se non conosci l'IP del tuo dispositivo puoi:

    - controlla l'IP nelle impostazioni WiFi del tuo dispositivo.
    - usa ADB per scoprire IP (via USB):
        1. Collegare il dispositivo al computer tramite USB
        2. In una riga di comando, digita `adb shell ifconfig` e copia l'indirizzo IP del dispositivo

    Per **tornare al** debug tramite **USB,** utilizzare il seguente comando:

```
adb usb
```

Puoi anche collegare ADB tramite WiFi installando un plug-in su Android Studio. Per farlo, vai su *Impostazioni> Plugin* e Sfoglia repository, cerca *ADB WiFi* , installalo e riapri Android Studio. Verrà visualizzata una nuova icona nella barra degli strumenti come mostrato nell'immagine seguente. Collega il dispositivo al computer host tramite USB e fai clic sull'icona di *AndroidWiFiADB* . Visualizzerà un messaggio se il tuo dispositivo è connesso o meno. Una volta collegato, puoi scollegare l'USB.



Dispositivo rooted

Nota: alcuni dispositivi che **sono rootati** possono utilizzare l'app WiFi ADB dal Play Store per abilitarla in modo semplice. Inoltre, per alcuni dispositivi (specialmente quelli con CyanogenMod ROM) questa opzione è presente nelle Opzioni sviluppatore tra le Impostazioni. Abilitandolo ti darò l'indirizzo IP e il numero di porta necessari per connettersi ad `adb` semplicemente eseguendo `adb connect <ip address>:<port>` .

Quando si dispone di un dispositivo rooted ma non si ha accesso a un cavo USB

Il processo è spiegato in dettaglio nella seguente risposta: http://stackoverflow.com/questions/2604727/how-can-i-connect-to-android-with-adb-over-tcp/3623727#3623727 I comandi più importanti sono mostrati sotto.

Aprire un terminale nel dispositivo e digitare quanto segue:

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

Per esempio:

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

E sul tuo computer:

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

Per esempio:

```
adb connect 192.168.1.2:5555
```

Per disattivarlo:

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

Evitare il timeout

Per impostazione predefinita, `adb` terminerà il timeout dopo 5000 ms. Questo può accadere in alcuni casi, come il WiFi lento o l'APK di grandi dimensioni.

Un semplice cambiamento nella configurazione di Gradle può fare il trucco:

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

Tirare (spingere) i file da (a) il dispositivo

È possibile estrarre (scaricare) i file dal dispositivo eseguendo il seguente comando:

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

Per esempio:

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

Puoi anche inviare (caricare) file dal tuo computer al dispositivo:

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

Per esempio:

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

Esempio per recuperare il database dal dispositivo

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

Riavvia il dispositivo

È possibile riavviare il dispositivo eseguendo il seguente comando:

```
adb reboot
```

Esegui questo comando per riavviare nel bootloader:

```
adb reboot bootloader
```

Riavvia in modalità di ripristino:

```
adb reboot recovery
```

Tieni presente che il dispositivo non si spegne prima!

Accendi / spegni Wifi

Accendere:

```
adb shell svc wifi enable
```

Spegni:

```
adb shell svc wifi disable
```

Visualizza i dispositivi disponibili

Comando:

```
 adb devices
```

Esempio di risultato:

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

Prima colonna: numero di serie del dispositivo

Seconda colonna: stato della connessione

Documentazione di Android

Connetti dispositivo tramite IP

Inserisci questi comandi nel terminale del dispositivo Android

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

Dopo questo, è possibile utilizzare **CMD** e **ADB** per connettersi utilizzando il seguente comando

```
adb connect 192.168.0.101.5555
```

E puoi disabilitarlo e restituire ADB all'ascolto su USB con

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

Da un computer, se si dispone già dell'accesso USB (non è richiesta la root)

È ancora più semplice passare all'uso del Wi-Fi, se disponi già di USB. Da una riga di comando sul computer che ha il dispositivo connesso tramite USB, emettere i comandi

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

Sostituisci 192.168.0.101 con l'IP del dispositivo

Avvia / interrompi adb

Avvia ADB:

```
adb kill-server
```

Ferma ADB:

```
adb start-server
```

Visualizza logcat

È possibile eseguire `logcat` come comando adb o direttamente nel prompt della shell del proprio emulatore o dispositivo connesso. Per visualizzare l'output del registro utilizzando `adb` , accedere alla directory / strumenti della piattaforma SDK ed eseguire:

```
$ adb logcat
```

In alternativa, puoi creare una connessione shell a un dispositivo e quindi eseguire:

```
$ adb shell
$ logcat
```

Un comando utile è:

```
adb logcat -v threadtime
```

Visualizza la data, il tempo di chiamata, la priorità, il tag e il PID e il TID del thread che ha emesso il messaggio in un formato di messaggio lungo.

---

filtraggio

I log di Logcat hanno i cosiddetti livelli di log:

> **V** - Verbose, **D** - Debug, **I** - Info, **W** - Warning, **E** - Error, **F** - Fatal, **S** - Silent

Puoi anche filtrare logcat per livello di log. Ad esempio se vuoi solo emettere il livello di Debug:

```
adb logcat *:D
```

Logcat può essere filtrato dal nome di un pacchetto, ovviamente è possibile combinarlo con il filtro del livello di log:

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

È anche possibile filtrare il log utilizzando grep (ulteriori informazioni sull'output del filtro logcat qui):

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

In Windows, il filtro può essere utilizzato con findstr, ad esempio:

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

Per visualizzare il buffer di registro alternativo [main | events | radio], eseguire il `logcat` con l'opzione `-b` :

```
adb logcat -b radio
```

Salva l'output nel file:

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

Salva l'output nel file mentre lo guardi anche:

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

Pulizia dei registri:

```
adb logcat -c
```

Comando ADB diretto su dispositivo specifico in un'impostazione multi-dispositivo

1. Indirizzare un dispositivo per numero di serie

Utilizzare l'opzione `-s` seguita dal nome di un dispositivo per selezionare su quale dispositivo deve essere eseguito il comando `adb` . Le opzioni `-s` dovrebbero essere prima in linea, **prima** del comando.

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

Esempio:

```
adb devices

List of devices attached
emulator-5554       device
02157df2d1faeb33    device

adb -s emulator-5554 shell
```

Esempio # 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. Scegli come target un dispositivo, quando è connesso un solo tipo di dispositivo

Puoi scegliere come target l'unico emulatore in esecuzione con -e

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

Oppure puoi scegliere come target l'unico dispositivo USB collegato con -d

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

Acquisizione di uno screenshot e video (solo per kitkat) da un display del dispositivo

Schermata: Opzione 1 (adb puro)

Il comando `shell` adb ci consente di eseguire comandi utilizzando la shell incorporata di un dispositivo. Il comando `screencap` shell acquisisce il contenuto attualmente visibile su un dispositivo e lo salva in un determinato file immagine, ad es. `/sdcard/screen.png` :

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

È quindi possibile utilizzare il comando pull per scaricare il file dal dispositivo nella directory corrente sul proprio computer:

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

Schermata: Opzione 2 (più veloce)

Esegui il seguente one-liner:

(Marshmallow e precedenti):

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

(Torrone e successivo):

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

Il flag `-p` reindirizza l'output del comando `screencap` su stdout. L'espressione di Perl che viene convogliata per eliminare alcuni problemi di fine riga su Marshmallow e precedenti. Lo stream viene quindi scritto in un file denominato `screen.png` all'interno della directory corrente. Vedi questo articolo e questo articolo per maggiori informazioni.

video

questo funziona solo in KitKat e solo tramite ADB. Questo non funziona sotto Kitkat Per avviare la registrazione dello schermo del tuo dispositivo, esegui il seguente comando:

`adb shell screenrecord /sdcard/example.mp4` , questo comando inizierà a registrare lo schermo del tuo dispositivo usando le impostazioni predefinite e salva il video risultante in un file nel file `/sdcard/example.mp4` sul tuo dispositivo.

Quando hai finito di registrare, premi Ctrl + C (z in Linux) nella finestra del prompt dei comandi per interrompere la registrazione dello schermo. È quindi possibile trovare il file di registrazione dello schermo nella posizione specificata. Si noti che la registrazione dello schermo viene salvata nella memoria interna del dispositivo, non nel computer.

Le impostazioni predefinite servono per utilizzare la risoluzione dello schermo standard del dispositivo, codificare il video con un bitrate di 4 Mbps e impostare il tempo massimo di registrazione dello schermo su 180 secondi. Per ulteriori informazioni sulle opzioni della riga di comando che è possibile utilizzare, eseguire il seguente comando:

`adb shell screenrecord -help` , funziona senza fare il root del dispositivo. Spero che questo ti aiuti.

Cancella i dati dell'applicazione

È possibile cancellare i dati utente di un'app specifica utilizzando `adb` :

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

È lo stesso di navigare le impostazioni sul telefono, selezionare l'app e premere il pulsante Cancella dati.

- `pm` invoca il gestore pacchetti sul dispositivo
- `clear` cancella tutti i dati associati ad un pacchetto

Invio di trasmissione

È possibile inviare broadcast a `BroadcastReceiver` con `adb` .

In questo esempio, inviamo broadcast con action `com.test.app.ACTION` e stringa extra in bundle `'foo'='bar'` :

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

Puoi mettere in bundle qualsiasi altro tipo supportato, non solo le stringhe:

--ez - booleano
--ei - intero
--el - lungo
--ef - float
--eu - uri
--eia - array int (separato da ',')
--ela - array lungo (separato da ',')
--efa - array float (separato da ',')
--esa - array di stringhe (separato da ',')

Per inviare l'intento a specifici pacchetti / classi `-n` o `-p` parametro può essere usato.
Invio al pacchetto:

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

Invio a un componente specifico (`SomeReceiver` classe nel `com.test.app package`):

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

Esempi utili:

- Invio di una trasmissione "boot complete"
- Invio di una trasmissione "tempo modificato" dopo aver impostato l'ora tramite il comando adb

Installa ed esegui un'applicazione

Per installare un file APK , utilizzare il seguente comando:

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

o se l'app è esistente e vogliamo reinstallarla

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

Per disinstallare un'applicazione , dobbiamo specificare il suo pacchetto

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

Utilizzare il comando seguente per avviare un'app con un nome pacchetto fornito (o un'attività specifica in un'app):

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

Ad esempio, per avviare Waze:

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

di riserva

È possibile utilizzare il comando `adb backup` per eseguire il backup del dispositivo.

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

`-f <filename>` specifica nomefile **predefinito:** *crea backup.ab nella directory corrente*

`-apk|noapk` abilita / disabilita il backup di .apks **: default:** *-noapk*

`-obb|noobb` abilita / disabilita il backup di file aggiuntivi **default:** *-noobb*

`-shared|noshared` di memoria condivisa del dispositivo di backup `-shared|noshared` / **predefinito della** scheda SD **:** *-noshared*

`-all` backup di tutte le applicazioni installate

`-system|nosystem` include le applicazioni di sistema **predefinite:** *-sistema*

`<packages>` un elenco di pacchetti di cui eseguire il backup (es. com.example.android.myapp) (non necessario se `-all` è specificato)

Per un backup completo del dispositivo, incluso tutto, utilizzare

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

Nota: l' esecuzione di un backup completo può richiedere molto tempo.

Per ripristinare un backup, utilizzare

```
adb restore backup.ab
```

Installa ADB su sistema Linux

Come installare Android Debugging Bridge (ADB) su un sistema Linux con il terminale usando i repository della tua distro.

Installa sul sistema Ubuntu / Debian tramite apt:

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

Installa nel sistema Fedora / CentOS via yum:

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

Installa sul sistema Gentoo con portage:

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

Installa sul sistema openSUSE con zypper:

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

Installa su Arch system con pacman:

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

Elencare tutte le autorizzazioni che richiedono la concessione di runtime dagli utenti su Android 6.0

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

Visualizza i dati interni di un'app (dati / dati /) su un dispositivo

Innanzitutto, assicurati che la tua app possa essere sottoposta a backup in `AndroidManifest.xml` , ad esempio `android:allowBackup` non è `false` .

Comando di backup:

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

Creare un tar con il 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
```

Estrai il tar:

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

È quindi possibile visualizzare il contenuto estratto.

Visualizza lo stack delle attività

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

Molto utile se utilizzato insieme al comando `watch` unix:

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

Visualizza e tira i file di cache di un'app

È possibile utilizzare questo comando per elencare i file per il proprio apk di debugging:

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

E questo script per estrarre dalla cache, copiare prima il contenuto in sdcard, estrarlo e quindi rimuoverlo alla fine:

```
#!/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'"
```

Quindi puoi estrarre un file dalla cache in questo modo:

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

Ottieni file di database tramite ADB

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

Leggi ADB (Android Debug Bridge) online: https://riptutorial.com/it/android/topic/1051/adb--android-debug-bridge-

Capitolo 6: Adb shell

introduzione

`adb shell` apre una shell Linux in un dispositivo o emulatore di destinazione. È il modo più potente e versatile per controllare un dispositivo Android tramite `adb` .

Questo argomento è stato diviso da ADB (Android Debug Bridge) a causa del raggiungimento del limite di esempi, molti dei quali riguardavano il comando della `adb shell` .

Sintassi

- adb shell [-e escape] [-n] [-Tt] [-x] [comando]

Parametri

Parametro	Dettagli
-e	scegli il carattere di escape o "none"; predefinito '~'
-n	non leggere da stdin
-T	disabilitare l'allocazione PTY
-t	forzare l'allocazione del PTY
-X	disabilita i codici di uscita remoti e la separazione stdout / stderr

Examples

Invia testo, tasto premuto e tocca gli eventi sul dispositivo Android tramite ADB

eseguire il seguente comando per inserire il testo in una vista con un focus (se supporta l'immissione di testo)

6.0

Invia il testo su SDK 23+

```
adb shell "input keyboard text 'Paste text on Android Device'"
```

Se sei già connesso al tuo dispositivo tramite `adb` :

```
input text 'Paste text on Android Device'
```

6.0

Invia il testo prima di SDK 23

```
adb shell "input keyboard text 'Paste%stext%son%sAndroid%sDevice'"
```

Gli spazi non sono accettati come input, sostituirli con% s.

Invia eventi

Per simulare la pressione del tasto di accensione dell'hardware

```
adb shell input keyevent 26
```

o in alternativa

```
adb shell input keyevent POWER
```

Anche se non si dispone di una chiave hardware, è comunque possibile utilizzare un `keyevent` per eseguire l'azione equivalente

```
adb shell input keyevent CAMERA
```

Invia evento touch come input

```
adb shell input tap Xpoint Ypoint
```

Invia evento di scorrimento come input

```
adb shell input swipe Xpoint1 Ypoint1 Xpoint2 Ypoint2 [DURATION*]
```

* DURATION è facoltativo, impostazione predefinita = 300ms. fonte

Ottieni punti X e Y abilitando la posizione del puntatore nell'opzione sviluppatore.

Script di shell di esempio ADB

> Per eseguire uno script in Ubuntu, creare script.sh, fare clic con il pulsante destro del mouse sul file e aggiungere il permesso di lettura / scrittura e selezionare **Consenti il file in esecuzione come programma** .
>
> Aprire l'emulatore di terminale ed eseguire il 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
```

Per un elenco completo dei numeri degli eventi

- elenco di alcuni eventi interessanti ADB Shell Input Events
- documentazione di riferimento https://developer.android.com/reference/android/view/KeyEvent.html#KEYCODE_POWER .

Elenca i pacchetti

Stampa tutti i pacchetti, opzionalmente solo quelli il cui nome del pacchetto contiene il testo in <FILTER>.

```
adb shell pm list packages [options] <FILTER>

All <FILTER>

adb shell pm list packages
```

attributi:

`-f` per vedere il loro file associato.

`-i` Vedi l'installer per i pacchetti.

`-u` includere anche i pacchetti disinstallati.

`-u` Includono anche pacchetti disinstallati.

Attributi che filtrano:

`-d` per pacchetti disabilitati.

`-e` per i pacchetti abilitati.

`-s` per pacchetti di sistema.

`-3` per pacchetti di terze parti.

`--user <USER_ID>` per uno spazio utente specifico da interrogare.

Concessione e revoca delle autorizzazioni API 23+

Un one-liner che aiuta a concedere o revocare le autorizzazioni vulnerabili.

- **rilascio**

```
adb shell pm grant <sample.package.id> android.permission.<PERMISSION_NAME>
```

- **revoca**

```
adb shell pm revoke <sample.package.id> android.permission.<PERMISSION_NAME>
```

- **Concessione di tutte le autorizzazioni di runtime alla volta durante l'installazione (-g)**

```
adb install -g /path/to/sample_package.apk
```

Stampa i dati dell'applicazione

Questo comando stampa tutti i dati rilevanti dell'applicazione:

- codice di versione
- nome della versione
- autorizzazioni concesse (API Android 23+)
- eccetera..

```
adb shell dumpsys package <your.package.id>
```

Registrazione del display

4.4

Registrazione del display di dispositivi con Android 4.4 (livello API 19) e versioni successive:

```
adb shell screenrecord [options] <filename>
adb shell screenrecord /sdcard/demo.mp4
```

(premi Ctrl-C per interrompere la registrazione)

Scarica il file dal dispositivo:

```
adb pull /sdcard/demo.mp4
```

> Nota: interrompere la registrazione dello schermo premendo Ctrl-C, altrimenti la registrazione si interrompe automaticamente a tre minuti o il limite di tempo impostato da `--time-limit` .

```
adb shell screenrecord --size <WIDTHxHEIGHT>
```

Imposta la dimensione del video: 1280x720. Il valore predefinito è la risoluzione di visualizzazione nativa del dispositivo (se supportato), 1280x720 in caso contrario. Per risultati ottimali, utilizza una dimensione supportata dal codificatore AVC (Advanced Video Coding) del tuo dispositivo.

```
adb shell screenrecord --bit-rate <RATE>
```

Imposta la velocità in bit del video per il video, in megabit al secondo. Il valore predefinito è 4Mbps. È possibile aumentare la velocità in bit per migliorare la qualità del video, ma in questo modo si ottengono file di film più grandi. L'esempio seguente imposta il bit rate di registrazione su 5Mbps:

```
adb shell screenrecord --bit-rate 5000000 /sdcard/demo.mp4
```

```
adb shell screenrecord --time-limit <TIME>
```

Imposta il tempo di registrazione massimo, in secondi. Il valore predefinito e massimo è 180 (3 minuti).

```
adb shell screenrecord --rotate
```

Ruota l'uscita di 90 gradi. Questa funzionalità è sperimentale.

```
adb shell screenrecord --verbose
```

Visualizza le informazioni del registro sullo schermo della riga di comando. Se non si imposta questa opzione, l'utilità non visualizza alcuna informazione durante l'esecuzione.

> Nota: questo potrebbe non funzionare su alcuni dispositivi.

4.4

Il comando di registrazione dello schermo non è compatibile con le versioni di Android pre 4.4

> Il comando screenrecord è un'utilità shell per registrare la visualizzazione di dispositivi con Android 4.4 (livello API 19) e versioni successive. L'utilità registra l'attività dello schermo su un file MPEG-4.

Modifica dei permessi dei file usando il comando chmod

Si noti che, per modificare le autorizzazioni di file, il dispositivo deve essere rootato, *`su` binary non viene fornito con i dispositivi spediti in fabbrica!*

Convenzione:

```
adb shell su -c "chmod <numeric-permisson> <file>"
```

Autorizzazione numerica costruita da sezioni utente, gruppo e mondo.

Ad esempio, se vuoi cambiare il file per essere leggibile, scrivibile ed eseguibile da tutti, questo sarà il tuo comando:

```
adb shell su -c "chmod 777 <file-path>"
```

O

```
adb shell su -c "chmod 000 <file-path>"
```

se intendi negare qualsiasi autorizzazione ad esso.

1a cifra: specifica il permesso dell'utente, **2a cifra** : specifica il permesso del gruppo, **3a cifra** , specifica l'autorizzazione del mondo (altri).

Permessi di accesso:

```
--- :   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)
```

Imposta data / ora tramite adb

6.0

Il formato SET predefinito è `MMDDhhmm[[CC]YY][.ss]` , cioè (2 cifre ciascuno)

Ad esempio, per impostare il 17 luglio alle 10:10, senza modificare l'anno corrente, digitare:

```
adb shell 'date 07171010.00'
```

Suggerimento 1: la modifica della data non si rifletterà immediatamente e una modifica notevole avverrà solo dopo che l'orologio di sistema avanza al minuto successivo.
È possibile forzare un aggiornamento allegando una trasmissione di intenti `TIME_SET` alla chiamata, in questo modo:

```
adb shell 'date 07171010.00 ; am broadcast -a android.intent.action.TIME_SET'
```

Suggerimento 2: per sincronizzare l'orologio di Android con la tua macchina locale:

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"
```

Entrambi i suggerimenti insieme:

```
adb shell 'date `date +%m%d%H%M%G.%S` ; am broadcast -a android.intent.action.TIME_SET'
```

6.0

Il formato SET predefinito è 'YYYYMMDD.HHmmss'

```
adb shell 'date -s 20160117.095930'
```

Suggerimento: per sincronizzare l'orologio di Android con la tua macchina locale (basata su Linux):

```
adb shell date -s `date +%G%m%d.%H%M%S`
```

Apri Opzioni sviluppatore

```
adb shell am start -n com.android.settings/.DevelopmentSettings
```

Navigherà il tuo dispositivo / emulatore nella sezione `Developer Options` .

Generazione di una trasmissione "Boot Complete"

Questo è rilevante per le app che implementano un `BootListener` . Prova la tua app uccidendo la tua app e poi prova con:

```
adb shell am broadcast -a android.intent.action.BOOT_COMPLETED -c android.intent.category.HOME
-n your.app/your.app.BootListener
```

(sostituire `your.package/your.app.BootListener` con valori corretti).

Visualizza contenuto di archiviazione esterno / secondario

Visualizza contenuto:

```
adb shell ls \$EXTERNAL_STORAGE
adb shell ls \$SECONDARY_STORAGE
```

Visualizza percorso:

```
adb shell echo \$EXTERNAL_STORAGE
adb shell echo \$SECONDARY_STORAGE
```

uccidere un processo all'interno di un dispositivo Android

A volte il logcat di Android funziona in modo infinito con errori derivanti da un processo non di tua proprietà, esaurendo la batteria o rendendo difficile il debug del codice.

Un modo conveniente per risolvere il problema senza riavviare il dispositivo è individuare e

terminare il processo che causa il problema.

Da Logcat

```
03-10 11:41:40.010 1550-1627/? E/SomeProcess: ....
```

nota il numero del processo: 1550

Ora possiamo aprire una shell e uccidere il processo. Nota che non possiamo eliminare il processo di `root` .

```
adb shell
```

all'interno della shell possiamo controllare di più sul processo usando

```
ps -x | grep 1550
```

e uccidilo se vogliamo:

```
kill -9 1550
```

Leggi Adb shell online: https://riptutorial.com/it/android/topic/9408/adb-shell

Capitolo 7: AdMob

Sintassi

- compila 'com.google.firebase: firebase-ads: 10.2.1' // NOTA: SET SU VERSIONE PIÙ NUOVA SE DISPONIBILE
- `<uses-permission android:name="android.permission.INTERNET" />` Necessario per recuperare l'annuncio
- AdRequest adRequest = new AdRequest.Builder (). Build (); // Banner pubblicitario
- AdView mAdView = (AdView) findViewById (R.id.adView); // Banner pubblicitario
- mAdView.loadAd (adRequest); // Banner pubblicitario

Parametri

Param	Dettagli
annunci: AdUnitId = "@ string / main_screen_ad"	L'ID del tuo annuncio. Ottieni il tuo ID dal sito di AdMob. *"Anche se non è un requisito, la conservazione dei valori ID dell'unità pubblicitaria in un file di risorse è una buona pratica: man mano che la tua app cresce e le tue esigenze di pubblicazione degli annunci sono mature, potrebbe essere necessario modificare i valori ID. file, non devi mai cercare attraverso il tuo codice per cercarli. "* . [1]

Osservazioni

- Richiede un account Admob valido
- Leggi la politica di AdMob . Assicurati di non fare nulla che possa far sospendere il tuo account AdMob

Examples

Implementazione

Nota: questo esempio richiede un account Admob valido e un codice dell'annuncio Admob valido.

Build.gradle a livello di app

Passare alla versione più recente se esistente:

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

Manifesto

È richiesto l'accesso a Internet per accedere ai dati degli annunci. Nota che questa autorizzazione non deve essere richiesta (utilizzando API 23+) poiché è un permesso normale e non pericoloso:

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

XML

Il seguente esempio XML mostra un banner pubblicitario:

```
<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" />
```

Per il codice di altri tipi, consulta la Guida di Google AdMob .

Giava

Il seguente codice è per l'integrazione di banner pubblicitari. Tieni presente che altri tipi di annunci potrebbero richiedere un'integrazione diversa:

```
// 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);
```

Aggiungi i metodi del ciclo di vita `AdView` nei metodi `onResume()` , `onPause()` e `onDestroy()` della tua attività:

```
@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();
}
```

Leggi AdMob online: https://riptutorial.com/it/android/topic/5334/admob

Capitolo 8: Affresco

introduzione

Fresco è un potente sistema per la visualizzazione di immagini in applicazioni Android.

In Android 4.xe versioni precedenti, Fresco inserisce le immagini in una regione speciale della **memoria Android** (chiamata ashmem). Ciò consente alla tua applicazione di funzionare più velocemente e subisce il temuto OutOfMemoryError molto meno spesso.

Affresco supporta anche lo streaming di file JPEG.

Osservazioni

Come impostare le dipendenze nel file build.gradle a livello di app:

```
dependencies {
    // Your app's other dependencies.
    compile 'com.facebook.fresco:fresco:0.14.1' // Or a newer version if available.
}
```

Maggiori informazioni possono essere trovate qui .

Examples

Iniziare con Affresco

Per prima cosa aggiungi Fresco al tuo `build.gradle` come mostrato nella sezione Note:

Se sono necessarie funzionalità aggiuntive, come GIF animate o supporto WebP, è necessario aggiungere anche gli artefatti da affresco corrispondenti.

L'affresco deve essere inizializzato. Dovresti farlo solo 1 volta, quindi posizionare l'inizializzazione nella tua `Application` è una buona idea. Un esempio per questo sarebbe:

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        Fresco.initialize(this);
    }
}
```

Se vuoi caricare immagini remote da un server, la tua app necessita dell'autorizzazione interna. Basta aggiungerlo al tuo `AndroidManifest.xml` :

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

Quindi, aggiungi un `SimpleDraweeView` al tuo layout XML. Affresco non supporta `wrap_content` per le dimensioni dell'immagine poiché potresti avere più immagini con dimensioni diverse (immagine segnaposto, immagine di errore, immagine reale, ...).

Quindi puoi aggiungere un `SimpleDraweeView` con dimensioni fisse (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" />
```

Oppure fornisci *proporzioni* per la tua immagine:

```
<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" />
```

Infine, puoi impostare l'URI dell'immagine in Java:

```
SimpleDraweeView draweeView = (SimpleDraweeView) findViewById(R.id.my_image_view);
draweeView.setImageURI("http://yourdomain.com/yourimage.jpg");
```

Questo è tutto! Dovresti vedere il tuo segnaposto disegnabile fino a quando l'immagine di rete è stata recuperata.

Usando OkHttp 3 con Affresco

Innanzitutto, oltre alla normale dipendenza di Fresco Gradle, devi aggiungere la dipendenza di OkHttp 3 al tuo `build.gradle` :

```
compile "com.facebook.fresco:imagepipeline-okhttp3:1.2.0" // Or a newer version.
```

Quando si inizializza Fresco (di solito nell'implementazione `Application` personalizzata), ora è possibile specificare il client 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 affresco utilizzando DraweeController

Questo esempio presume che tu abbia già aggiunto Fresco alla tua app (vedi questo esempio):

```
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.
```

Leggi Affresco online: https://riptutorial.com/it/android/topic/5217/affresco

Capitolo 9: Aggiunta di un FuseView a un progetto Android

introduzione

Esportare un Fuse.View da fusetools e utilizzarlo all'interno di un progetto Android esistente.

Il nostro obiettivo è quello di esportare l'intera applicazione del campione hikr e utilizzarlo all'interno di `Activity` .

Il lavoro finale può essere trovato @ lucamtudor / hikr-fuse-view

Examples

hikr app, solo un altro android.view.View

Prerequisiti

- dovresti avere un fusibile installato (https://www.fusetools.com/downloads)
- avresti dovuto fare il tutorial introduttivo
- nel terminale: `fuse install android`
- nel terminale: `uno install Fuse.Views`

Passo 1

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

Passaggio 2 : aggiungi il riferimento del pacchetto a `Fuse.Views`

Trova il file `hikr.unoproj` all'interno della cartella principale del progetto e aggiungi `"Fuse.Views"` alla matrice `"Packages"` .

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

Passaggio 3 : rendere il componente `HikrApp` per contenere l'intera app

3.1 Nella cartella principale del progetto crea un nuovo file chiamato `HikrApp.ux` e incolla il contenuto di `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 In `HikrApp.ux`

- sostituire i tag `<App>` con `<Page>`
- aggiungi `ux:Class="HikrApp"` all'apertura `<Page>`
- rimuovere `<ClientPanel>` , non dobbiamo più preoccuparci della barra di stato o dei pulsanti di navigazione in basso

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 Utilizzare il componente `HikrApp` appena creato all'interno di `MainView.ux`

Sostituisci il contenuto del file `MainView.ux` con:

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

La nostra app è tornata al suo normale comportamento, ma ora l'abbiamo estratta in un componente separato chiamato `HikrApp`

Passaggio 4 All'interno di `MainView.ux` sostituire i tag `<App>` con `<ExportedViews>` e aggiungere `ux:Template="HikrAppView"` a `<HikrApp />`

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

Ricorda il modello `HikrAppView` , perché ne avremo bisogno per ottenere un riferimento alla nostra vista da Java.

Nota Dai documenti del fusibile:

> `ExportedViews` si comporterà come `App` quando `fuse preview` normale e `uno build`

Non vero. Si otterrà questo errore durante l'anteprima da Fuse Studio:

> Errore: impossibile trovare un tag app in uno dei file UX inclusi. Hai dimenticato di includere il file UX che contiene il tag app?

Passaggio 5 Avvolgere il `<DockPanel>` Wrap `SplashPage.ux` in 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>
```

Passaggio 6 Esportare il progetto di fusibile come libreria di aar

- nel terminale, nella cartella del progetto root: `uno clean`
- nel terminale, nella cartella del progetto root: `uno build -t=android -DLIBRARY`

Passaggio 7 Prepara il tuo progetto Android

- copia l'aar da `.../rootHikeProject/build/Android/Debug/app/build/outputs/aar/app-debug.aar` a `.../androidRootProject/app/libs`
- aggiungi `flatDir { dirs 'libs' }` al file 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'
        }
    }
}

...
```

- aggiungi `compile(name: 'app-debug', ext: 'aar')` alle dipendenze in `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'
}
```

- aggiungi le seguenti proprietà all'attività all'interno di `AndroidManifest.xml`

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

Il tuo `AndroidManifest.xml` sarà simile a questo:

```
<?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>
```

Passaggio 8 : mostra `Fuse.View HikrAppView` nella tua `Activity`

- nota che la tua `Activity` deve ereditare `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

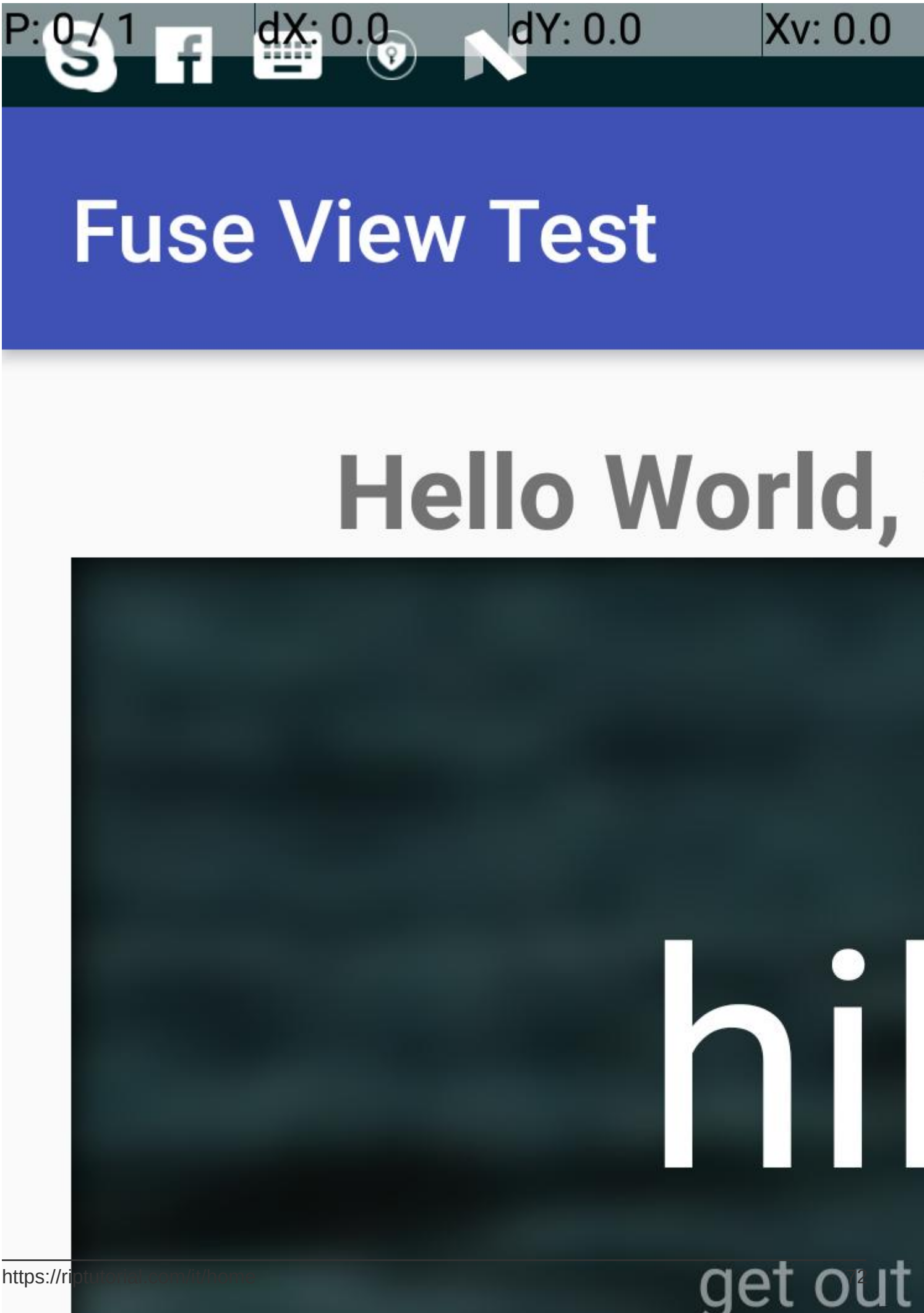
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    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
Quando premi il pulsante Indietro, su Android, l'app si arresta in modo anomalo. Puoi seguire il problema sul forum dei fusibili .

```
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
```

E il risultato finale è qualcosa di simile. Puoi anche trovare una breve clip su github .


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

https://riptutorial.com/it/android/topic/10052/aggiunta-di-un-fuseview-a-un-progetto-android

Capitolo 10: AIDL

introduzione

AIDL è la lingua di definizione dell'interfaccia Android.

Che cosa? Perché? Come ?

Che cosa? È un servizio limitato. Questo servizio AIDL sarà attivo fino a quando uno dei client è esistente. Funziona sulla base del concetto di marshalling e unmarshaling.

Perché? Le applicazioni remote possono accedere al servizio + Multi Threading (richiesta dell'applicazione remota).

Come? Creare il file .aidl Implementare l'interfaccia Esporre l'interfaccia ai client

Examples

Servizio 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;
        }
    };

}
```

Connessione di servizio

```
 // 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);
        }
    };
```

Leggi AIDL online: https://riptutorial.com/it/android/topic/9504/aidl

Capitolo 11: AlarmManager

Examples

Esegui un intento in un secondo momento

1. Crea un ricevitore. Questa classe riceverà l'intento e gestirà come desideri.

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

2. Dare un intento a AlarmManager. Questo esempio attiverà l'intenzione di essere inviata ad AlarmReceiver dopo 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 );
```

Come annullare un allarme

Se si desidera annullare un allarme e non si dispone di un riferimento al PendingIntent originale utilizzato per impostare l'allarme, è necessario ricreare un PendingIntent esattamente com'era quando è stato originariamente creato.

Un intento è considerato uguale da AlarmManager :

> se la loro azione, i dati, il tipo, la classe e le categorie sono gli stessi. Questo non confronta alcun dato extra incluso negli intenti.

Di solito il codice di richiesta per ogni allarme è definito come una costante:

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

Quindi, per un semplice allarme impostato in questo modo:

```
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);
```

Ecco come si creerebbe un nuovo riferimento PendingIntent che è possibile utilizzare per annullare l'allarme con un nuovo riferimento 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);
}
```

Creazione di allarmi esatti su tutte le versioni di Android

Con l'aumentare del numero di ottimizzazioni della batteria nel sistema Android nel tempo, anche i metodi di `AlarmManager` sono stati modificati in modo significativo (per consentire tempi più lievi). Tuttavia, per alcune applicazioni è ancora necessario essere il più precisi possibile su tutte le versioni di Android. Il seguente helper utilizza il metodo più accurato disponibile su tutte le piattaforme per pianificare 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);
    }
}
```

La modalità API2 + Doze interferisce con AlarmManager

Android 6 (API23) ha introdotto la modalità Doze che interferisce con AlarmManager. Usa determinate finestre di manutenzione per gestire gli allarmi, quindi anche se hai usato `setExactAndAllowWhileIdle()` non puoi fare in modo che il tuo allarme si `setExactAndAllowWhileIdle()` nel momento desiderato.

È possibile disattivare questo comportamento per l'app utilizzando le impostazioni del telefono (`Settings/General/Battery & power saving/Battery usage/Ignore optimizations` o simili)

All'interno della tua app puoi controllare questa impostazione ...

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

... e alla fine mostra la rispettiva finestra di dialogo delle impostazioni:

```
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);
```

Leggi AlarmManager online: https://riptutorial.com/it/android/topic/1361/alarmmanager

Capitolo 12: AlertDialog Box animato

introduzione

Finestra di avviso animata che mostra alcuni effetti di animazione .. Puoi ottenere alcune animazioni per finestre di dialogo come Fadein, Slideleft, Slidetop, SlideBottom, Slideright, Fall, Newspager, Fliph, Flipv, RotateBottom, RotateLeft, Slit, Shake, Sidefill per rendere il tuo applicazione attraente ..

Examples

Inserisci sotto codice per la finestra di dialogo animata ...

```
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();
    }
}
```

Aggiungi le linee sottostanti in build.gradle per includere NifyBuilder (CustomView)

build.gradle

```
 dependencies {

     compile 'com.nineoldandroids:library:2.4.0'

     compile 'com.github.sd6352051.niftydialogeffects:niftydialogeffects:1.0.0@aar'

 }
```

Link di riferimento: https://github.com/sd6352051/NiftyDialogEffects

Leggi AlertDialog Box animato online: https://riptutorial.com/it/android/topic/10654/alertdialog-box-animato

Capitolo 13: Android Java Native Interface (JNI)

introduzione

JNI (Java Native Interface) è un potente strumento che consente agli sviluppatori Android di utilizzare l'NDK e utilizzare il codice nativo C ++ nelle loro applicazioni. Questo argomento descrive l'utilizzo dell'interfaccia <-> C ++ Java.

Examples

Come chiamare le funzioni in una libreria nativa tramite l'interfaccia JNI

Java Native Interface (JNI) consente di chiamare funzioni native da codice Java e viceversa. Questo esempio mostra come caricare e chiamare una funzione nativa tramite JNI, non entra nell'accesso ai metodi e ai campi Java dal codice nativo utilizzando le funzioni JNI .

Supponiamo di avere una libreria nativa chiamata `libjniexample.so` nella `libjniexample.so project/libs/<architecture>` e si desidera chiamare una funzione dalla classe Java `JNITest` all'interno del pacchetto `com.example.jniexample` .

Nella classe JNITest, dichiara la funzione in questo modo:

```
public native int testJNIfunction(int a, int b);
```

Nel tuo codice nativo, definisci la funzione in questo modo:

```
#include <jni.h>

JNIEXPORT jint JNICALL Java_com_example_jniexample_JNITest_testJNIfunction(JNIEnv *pEnv,
jobject thiz, jint a, jint b)
{
    return a + b;
}
```

L'argomento `pEnv` è un puntatore all'ambiente JNI che è possibile passare alle funzioni JNI per accedere a metodi e campi di oggetti e classi Java. Il puntatore `thiz` è un riferimento di `jobject` all'oggetto Java su cui è stato chiamato il metodo nativo (o la classe se si tratta di un metodo statico).

Nel tuo codice Java, in `JNITest` , carica la libreria in questo modo:

```
static{
    System.loadLibrary("jniexample");
}
```

Notare la `lib` all'inizio e il `.so` alla fine del nome del file sono omessi.

Chiama la funzione nativa da Java in questo modo:

```
JNITest test = new JNITest();
int c = test.testJNIfunction(3, 4);
```

Come chiamare un metodo Java dal codice nativo

Java Native Interface (JNI) consente di chiamare le funzioni Java dal codice nativo. Ecco un semplice esempio di come farlo:

Codice Java:

```
package com.example.jniexample;
public class JNITest {
    public static int getAnswer(bool) {
        return 42;
    }
}
```

Codice 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 metodo JNI in tipo Java:

Firma JNI	Tipo Java
Z	booleano
B	byte
C	carbonizzare
S	corto
io	int

Firma JNI	Tipo Java
J	lungo
F	galleggiante
D	Doppio
L classe pienamente qualificata;	fully-qualified di classe
[genere	genere[]

Quindi per il nostro esempio abbiamo usato (Z) I - il che significa che la funzione ottiene un valore booleano e restituisce un int.

Metodo di utilità nel livello JNI

Questo metodo aiuterà a ottenere la stringa Java dalla stringa 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;
    }
```

Questo metodo ti aiuterà a convertire jbyteArray in 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;
}
```

Leggi Android Java Native Interface (JNI) online:
https://riptutorial.com/it/android/topic/8674/android-java-native-interface--jni-

Capitolo 14: Android Vk Sdk

Examples

Inizializzazione e login

1. Crea una nuova applicazione qui: crea un'applicazione
2. Scegli l'applicaton standalone e conferma la creazione dell'app tramite SMS.
3. Compila il nome del **pacchetto per Android** come nome del pacchetto corrente. *È possibile ottenere il nome del pacchetto all'interno del file manifest Android, all'inizio.*
4. Ottieni l' **impronta digitale del certificato** eseguendo questo comando nella tua shell / cmd:

```
keytool -exportcert -alias androiddebugkey -keystore path-to-debug-or-production-keystore -
list -v
```

Puoi anche ottenere questa impronta digitale dall'SDK stesso:

```
String[] fingerprints = VKUtil.getCertificateFingerprint(this, this.getPackageName());
Log.d("MainActivity", fingerprints[0]);
```

5. Aggiungi l'impronta digitale ricevuta nell'impronta digitale del **certificato di firma per Android:** campo nelle impostazioni dell'app Vk (dove hai inserito il nome del pacchetto)

6. Quindi aggiungi questo al tuo file gradle:

```
compile 'com.vk:androidsdk:1.6.5'
```

8. Inizializza l'SDK all'avvio usando il seguente metodo. Il modo migliore è chiamarlo nel metodo Applications 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);
}
```

Questo è il modo migliore per inizializzare VKSdk. Non utilizzare il methid in cui VK_ID deve essere inserito all'interno di strings.xml perché api non funzionerà correttamente dopo di esso.

9. Il passaggio finale è accedere utilizzando 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();
            }
        });
    }
```

Leggi Android Vk Sdk online: https://riptutorial.com/it/android/topic/6046/android-vk-sdk

Capitolo 15: Android-x86 in VirtualBox

introduzione

L'idea di questa sezione è di spiegare come installare e utilizzare VirtualBox con Android-x86 a scopo di debug. Questo è un compito difficile perché ci sono differenze tra le versioni. Per il momento ho intenzione di coprire 6.0, che è quello con cui ho dovuto lavorare e quindi dovremo trovare somiglianze.

Non copre in dettaglio VirtualBox o Linux, ma mostra i comandi che ho usato per farlo funzionare.

Examples

Configurazione della macchina virtuale

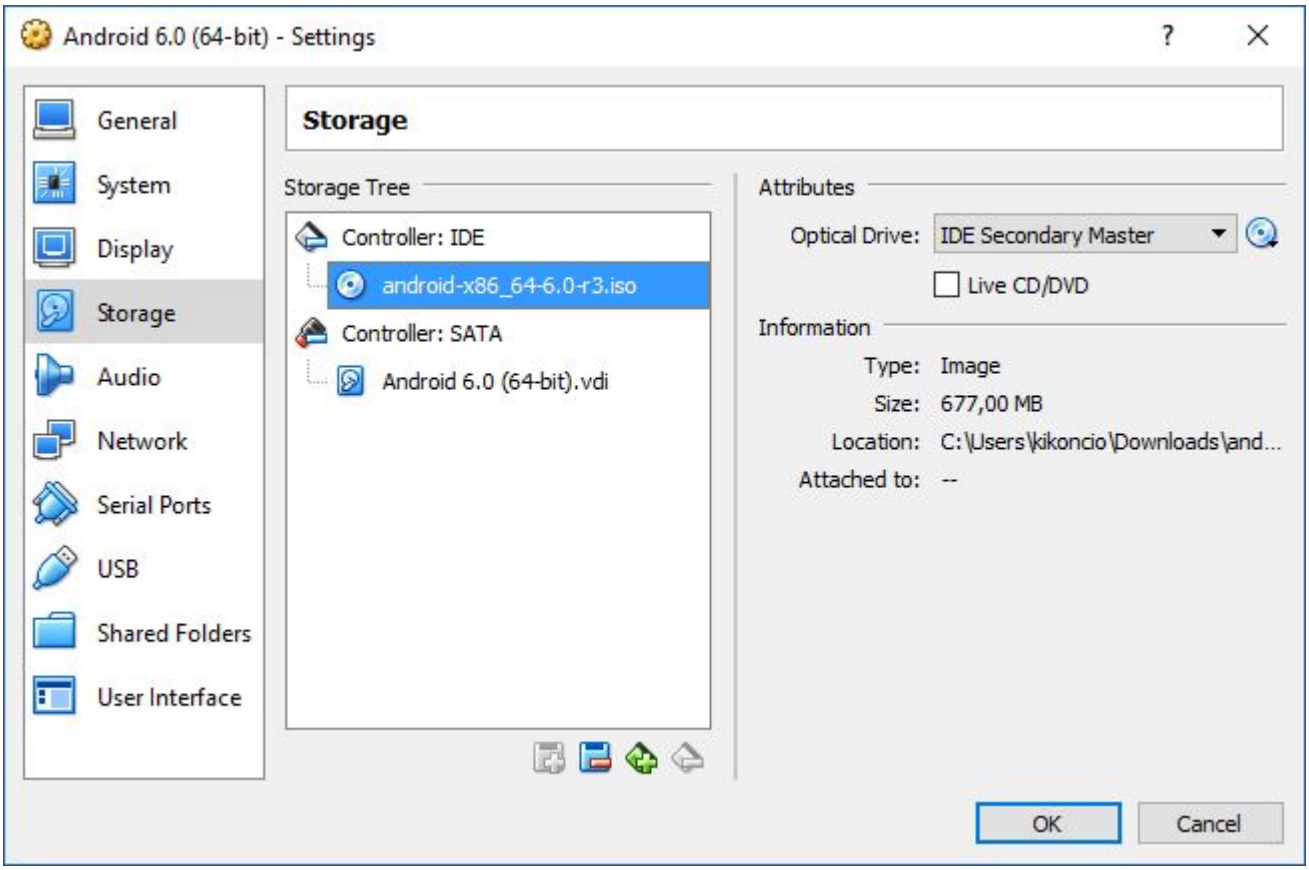
Queste sono le mie impostazioni VirtualBox:

- Tipo di sistema operativo: Linux 2.6 (Ho utente a 64 bit perché il mio computer può supportarlo)
- Dimensione del disco rigido virtuale: 4 GB
- Ram Memory: 2048
- Memoria video: 8M
- Dispositivo audio: Sound Blaster 16.
- Dispositivo di rete: PCnet-Fast III, collegato a NAT. È anche possibile utilizzare l'adattatore a ponte, ma è necessario un server DHCP nel proprio ambiente.

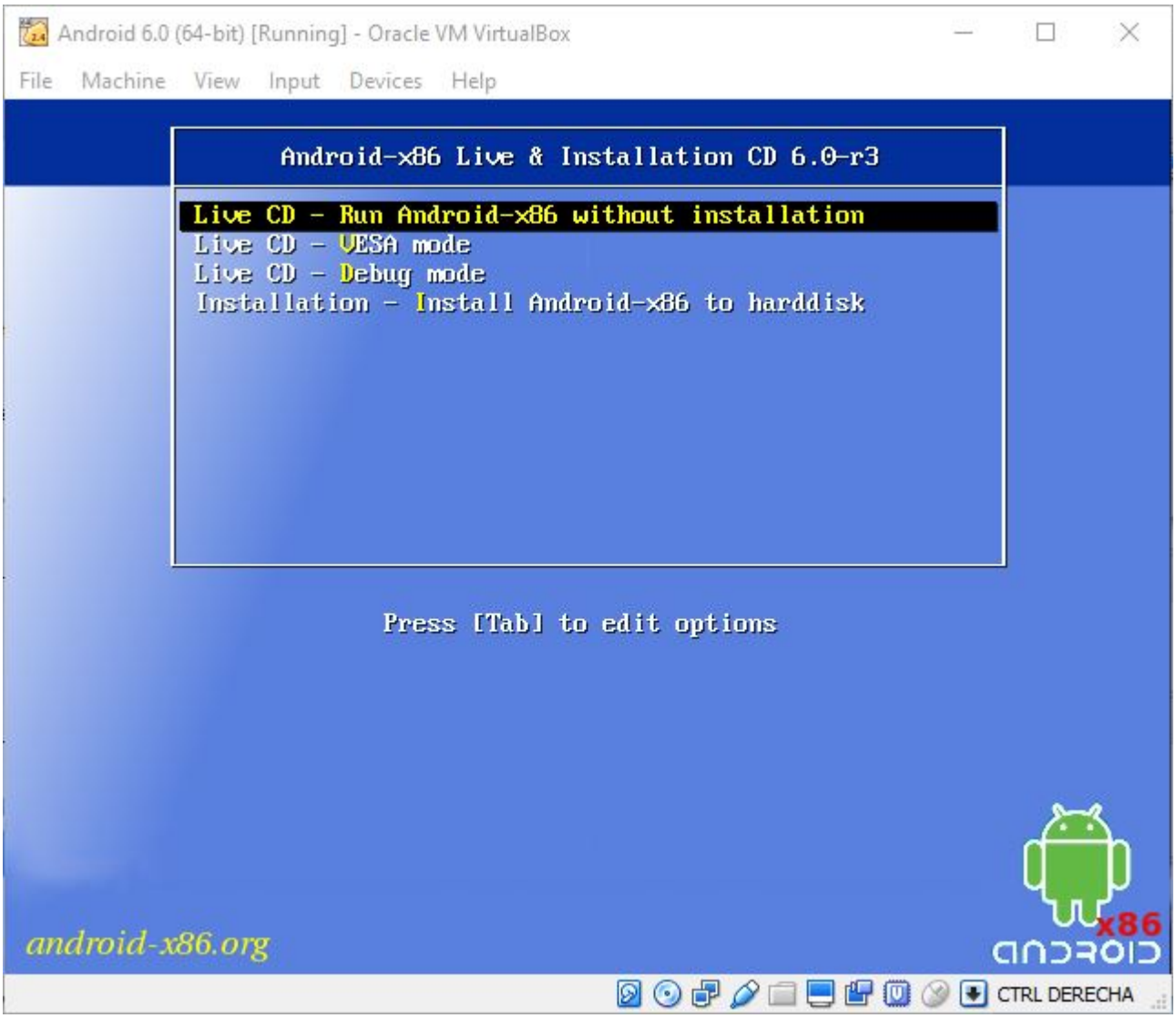
L'immagine utilizzata con questa configurazione è stata Android-x86_64-6.0-r3.iso (è 64 bit) scaricato da http://www.android-x86.org/download . Suppongo che funzioni anche con la versione a 32 bit.

Configurazione del disco rigido virtuale per supporto SDCARD

Con il disco rigido virtuale appena creato, avviare la macchina virtuale con l'immagine di android-x86 nell'unità ottica.



Una volta avviato, puoi vedere il menu di grub del Live CD



Scegli l'opzione Modalità debug, quindi dovresti vedere il prompt della shell. Questa è una shell busybox. È possibile ottenere più shell passando dalla console virtuale Alt-F1 / F2 / F3.

Creare due partizioni da fdisk (alcune altre versioni utilizzerebbero cfdisk). Formattali su ext3. Quindi riavviare:

```
# fdisk /dev/sda
```

Quindi digitare:

"n" (nuova partizione)

"p" (partizione primaria)

"1" (prima partizione)

"1" (primo cilindro)

"261" (scegli un cilindro, lasceremo il 50% del disco per una seconda partizione)

"2" (2a partizione)

"262" (262 ° cilindro)

"522" (scegli l'ultimo cilindro)

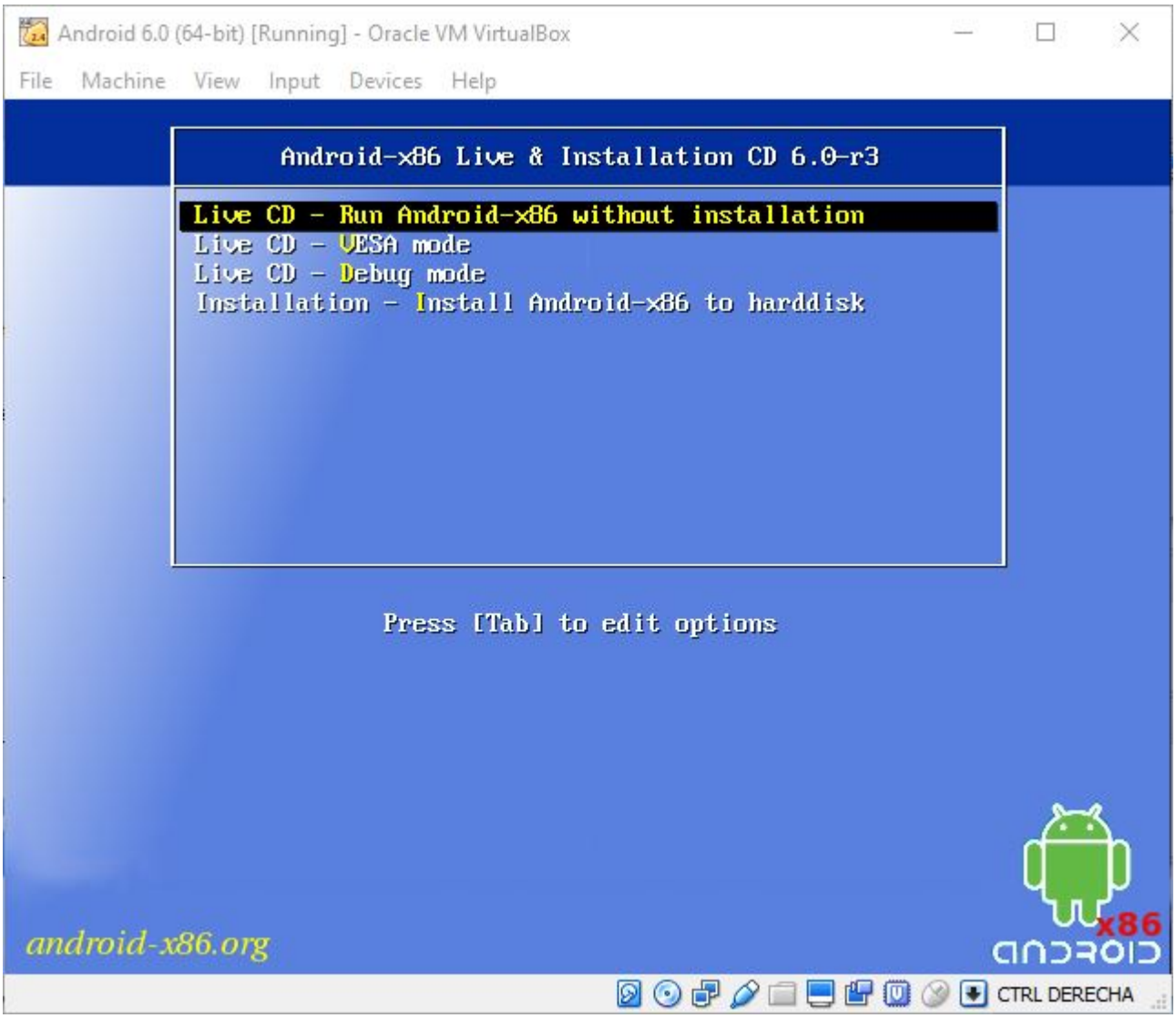
"w" (scrivi la partizione)

```
#mdev -s
#mke2fs -j -L DATA /dev/sda1
#mke2fs -j -L SDCARD /dev/sda2
#reboot -f
```

Quando si riavvia la macchina virtuale, viene visualizzato il menu Grub e si può modificare la linea di avvio del kernel in modo da poter aggiungere `DATA=sda1 SDCARD=sda2` opzioni `DATA=sda1 SDCARD=sda2` per puntare alla sdcard o alla partizione dati.

Installazione nella partizione

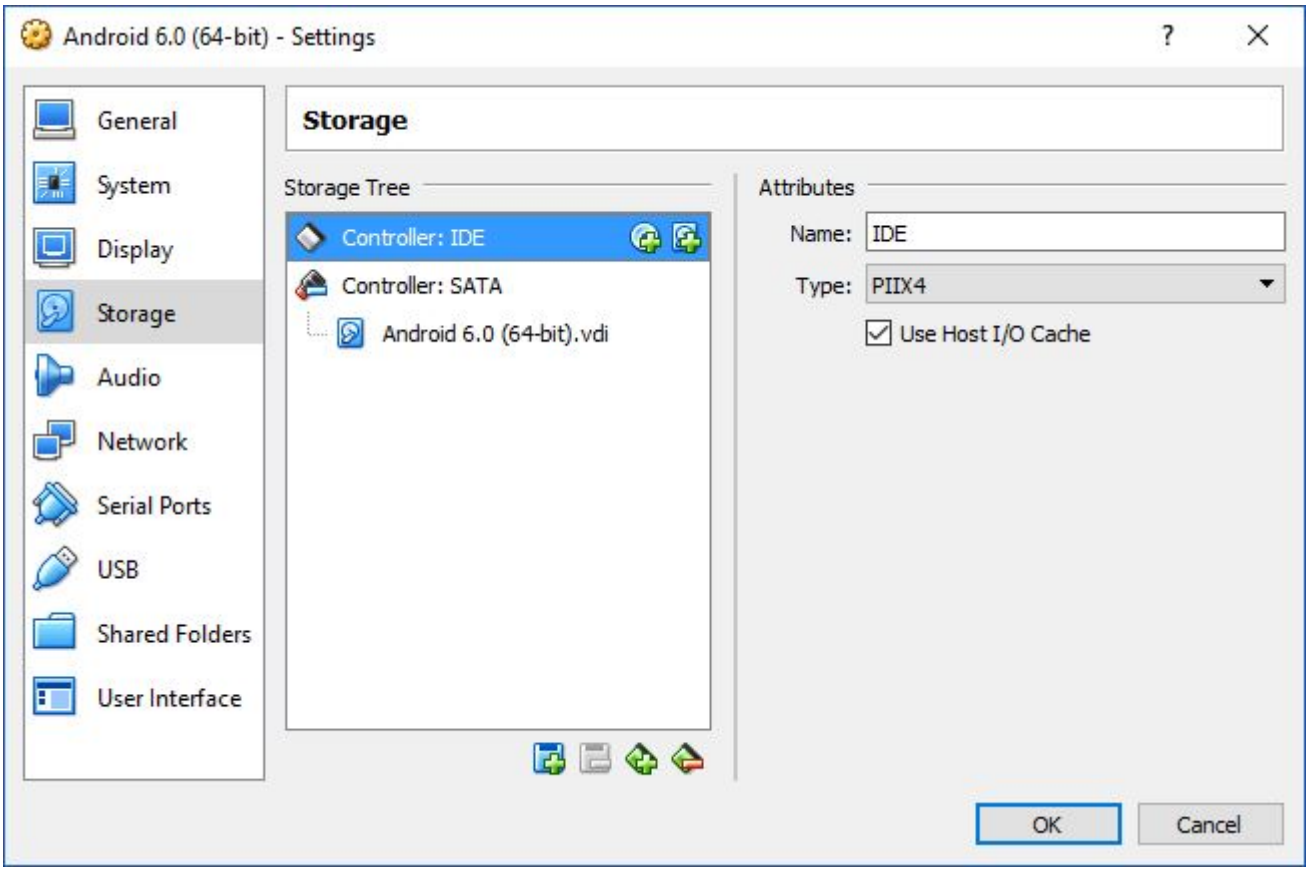
Con il disco rigido virtuale appena creato, avviare la macchina virtuale con l'immagine di android-x86 come unità ottica.



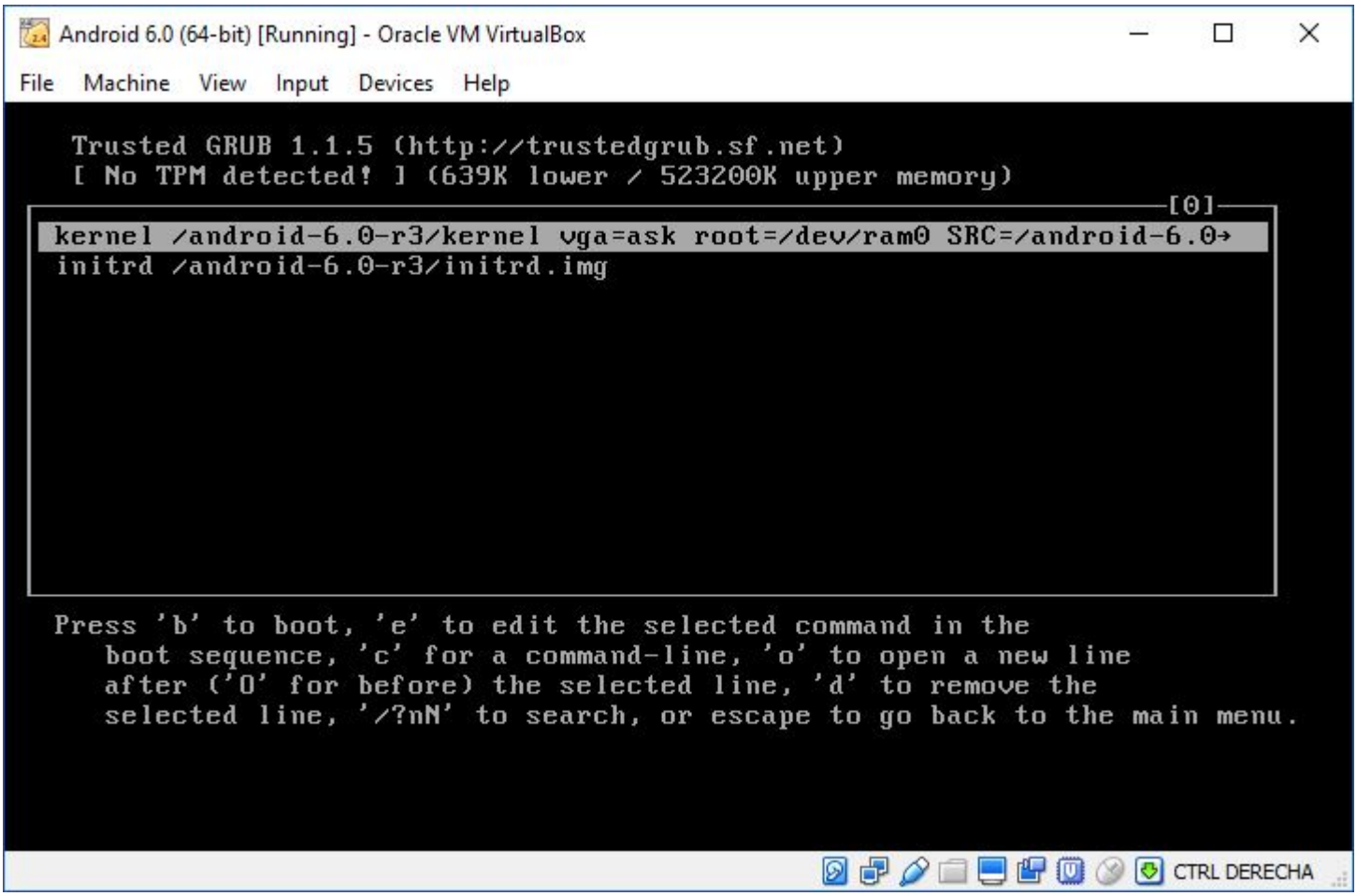
Nelle opzioni di avvio del Live CD scegli "Installazione - Installa Android su disco rigido"

Scegli la partizione sda1 e installa Android e installeremo grub.

Riavvia la macchina virtuale ma assicurati che l'immagine non si trovi nell'unità ottica in modo che possa essere riavviata dal disco rigido virtuale.



Nel menu di grub dobbiamo modificare il kernel come nell'opzione "Android-x86 6.0-r3", quindi premi e.

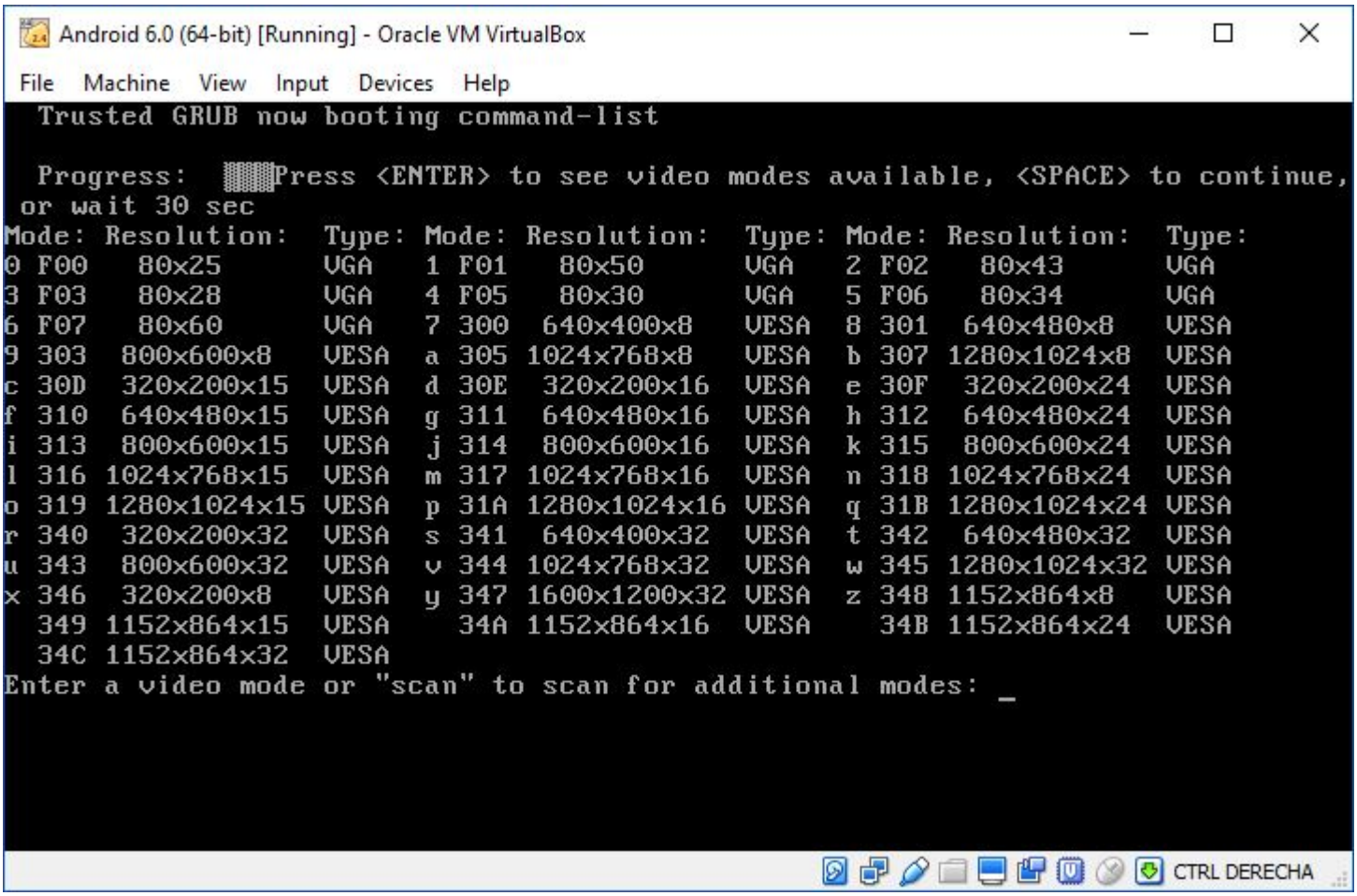


Quindi sostituiamo "quiet" con "vga = ask" e aggiungiamo l'opzione "SDCARD = sda2"

Nel mio caso, la linea del kernel appare così dopo la modifica:

```
kenel /android-6.0-r3/kernel vga=ask root=ram0 SRC=/android-6/android-6.0-r3 SDCARD=sda2
```

Premi b per avviare, quindi sarai in grado di scegliere le dimensioni dello schermo premendo INVIO (l'opzione `vga=ask`)



Una volta avviata la procedura guidata di installazione, scegli la lingua. Potrei scegliere inglese (Stati Uniti) e spagnolo (Stati Uniti) e ho avuto problemi a scegliere un altro.

Leggi Android-x86 in VirtualBox online: https://riptutorial.com/it/android/topic/9903/android-x86-in-virtualbox

Capitolo 16: animatori

Examples

Agitare l'animazione di un ImageView

Nella cartella res, crea una nuova cartella chiamata "anim" per memorizzare le risorse di animazione e inseriscila in quella cartella.

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" />
```

Crea un'attività vuota denominata Landing

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>
```

E il metodo per animare l'imageview su 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);
    }
```

Animazione di dissolvenza in entrata / uscita

Per ottenere una vista per dissolvenza in `ObjectAnimator` o in `ObjectAnimator` , utilizzare un `ObjectAnimator` . Come visto nel codice sottostante, imposta una durata usando `.setDuration(millis)` dove il parametro `millis` è la durata (in millisecondi) dell'animazione. Nel codice seguente, le viste si dissolvono in / out per oltre 500 millisecondi o 1/2 secondo. Per avviare `ObjectAnimator` di animazione 's, chiamare `.start()` . Una volta completata l'animazione, viene chiamato `onAnimationEnd(Animator animation)` . Ecco un buon punto per impostare la visibilità della vista su `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();
}
```

Animazione TransitionDrawable

Questo esempio mostra una transazione per una visualizzazione di immagine con solo due

immagini (può usare più immagini una dopo l'altra per le posizioni di primo e secondo livello dopo ogni transazione come un ciclo)

- aggiungi un array di immagini 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 un modo semplice per animare un valore (di un particolare tipo, ad esempio `int` , `float` , ecc.).

Il solito modo di usarlo è:

1. Crea un `ValueAnimator` che animerà un valore da `min` a `max`
2. Aggiungi un `UpdateListener` in cui utilizzerai il valore animato calcolato (che puoi ottenere con `getAnimatedValue()`)

Esistono due modi per creare `ValueAnimator` :

(il codice di esempio anima un `float` da `20f` a `40f` in `250ms`)

1. Da `xml` (mettilo in `/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. Dal codice:

```
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` è una sottoclasse di `ValueAnimator` con la possibilità aggiunta di impostare il valore calcolato sulla proprietà di una `View` `target` .

Proprio come in `ValueAnimator` , ci sono due modi per creare `ObjectAnimator` :

(il codice di esempio anima un `alpha` di una `View` da `0.4f` a `0.2f` in `250ms`)

1. Da `xml` (mettilo in `/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. Dal codice:

```
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` è un modo semplificato e ottimizzato per animare le proprietà di una `View` .

Ogni singola `View` ha un oggetto `ViewPropertyAnimator` disponibile tramite il metodo `animate()` . Puoi usarlo per animare più proprietà contemporaneamente con una semplice chiamata. Ogni singolo metodo di un oggetto `ViewPropertyAnimator` specifica il valore di **destinazione** di un parametro specifico a cui deve essere animato `ViewPropertyAnimator` .

```
View exampleView = ...;
exampleView.animate()
        .alpha(0.6f)
        .translationY(200)
        .translationXBy(10)
        .scaleX(1.5f)
        .setDuration(250)
        .setInterpolator(new FastOutLinearInInterpolator());
```

Nota: la chiamata `start()` su un oggetto `ViewPropertyAnimator` **NON** è obbligatoria. Se non lo fai, stai semplicemente lasciando che la piattaforma gestisca l'avvio dell'animazione nel momento appropriato (passaggio successivo per la gestione dell'animazione). Se lo fai effettivamente (chiamata `start()`) ti stai assicurando che l'animazione sia avviata immediatamente.

Espandi e comprimi l'animazione della 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);
        }
    }
```

Leggi animatori online: https://riptutorial.com/it/android/topic/1829/animatori

Capitolo 17: Annotazioni Typedef: @IntDef, @StringDef

Osservazioni

Il pacchetto di annotazioni include un numero di annotazioni di metadati utili con cui puoi decorare il tuo codice, per aiutare a cogliere i bug.

Basta aggiungere la dipendenza nel file `build.gradle` .

```
dependencies {
    compile 'com.android.support:support-annotations:25.3.1'
}
```

Examples

Annotazioni IntDef

Questa annotazione garantisce che vengano utilizzate solo le costanti integer valide che ci si aspetta.
Il seguente esempio illustra i passaggi per creare un'annotazione:

```
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;
    }
}
```

Inoltre, consentono il completamento del codice per offrire automaticamente le costanti consentite.

Quando si genera questo codice, viene generato un avviso se il parametro type non fa riferimento a una delle costanti definite.

Combinare le costanti con le bandiere

Utilizzando l' `IntDef#flag()` impostato su `true` , è possibile combinare più costanti.

Utilizzando lo stesso esempio in questo argomento:

```
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)

    .....

}
```

Gli utenti possono combinare le costanti permesse con un flag (come `|` , `&` , `^`).

Leggi Annotazioni Typedef: @IntDef, @StringDef online:
https://riptutorial.com/it/android/topic/4505/annotazioni-typedef---intdef---stringdef

Capitolo 18: API Android Places

Examples

Inserisci esempio di utilizzo del selettore

Place Picker è un widget UI davvero semplice fornito da Places API. Fornisce una mappa integrata, posizione corrente, luoghi vicini, capacità di ricerca e completamento automatico.

Questo è un esempio di utilizzo del widget dell'interfaccia utente di Selettore dei luoghi.

```
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()));
        }
    }
}
```

Ottenere luoghi attuali utilizzando l'API di Places

Puoi ottenere la posizione corrente e i luoghi locali dell'utente utilizzando l' API di Google Places .

Per prima cosa, dovresti chiamare il metodo `PlaceDetectionApi.getCurrentPlace()` per recuperare attività commerciali locali o altri luoghi. Questo metodo restituisce un oggetto `PlaceLikelihoodBuffer` che contiene un elenco di oggetti `PlaceLikelihood` . Quindi, puoi ottenere un oggetto `Place` chiamando il metodo `PlaceLikelihood.getPlace()` .

Importante: devi richiedere e ottenere l'autorizzazione `ACCESS_FINE_LOCATION` per consentire alla tua app di accedere a informazioni precise sulla posizione.

```
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());
}
```

Inserire l'integrazione del completamento automatico

La funzione di completamento automatico dell'API di Google Places per Android fornisce le previsioni dei luoghi all'utente. Mentre l'utente digita nella casella di ricerca, il completamento automatico mostra i posti in base alle query dell'utente.

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"/>
```

Aggiunta di più di un'attività completa automatica di Google.

```
 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.
           }
        }
    }
```

Impostazione dei filtri dei tipi di luogo per PlaceAutocomplete

In alcuni scenari, potremmo voler restringere i risultati mostrati da **PlaceAutocomplete** a un paese specifico o magari mostrare solo regioni. Questo può essere ottenuto impostando un **filtro di completamento automatico** sull'intento. Ad esempio, se voglio cercare solo i luoghi di tipo REGION e solo appartenenti all'India, farei quanto segue:

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>
```

PlaceAutocomplete si avvierà automaticamente e potrai quindi selezionare un luogo dai risultati che saranno solo del tipo **REGION** e apparterranno solo al paese specificato. L'intento può anche essere lanciato con un clic di un pulsante.

Leggi API Android Places online: https://riptutorial.com/it/android/topic/4111/api-android-places

Capitolo 19: API di Awareness di Google

Osservazioni

Ricorda che l' API Snapshot viene utilizzata per richiedere lo stato corrente mentre l' API Fence verifica continuamente lo stato specificato e invia i callback quando un'app non è in esecuzione.

Nel complesso, ci sono alcuni passaggi di base per utilizzare l'API Snapshot o l'API Fence:

- Ottieni una chiave API dalla Google Developers Console
- Aggiungi le autorizzazioni necessarie e la chiave API al manifest:

```
<!-- 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"/>
```

- Initalizzare `GoogleApiClient` da qualche parte, preferibilmente nel metodo onCreate () della tua attività.

```
GoogleApiClient client = new GoogleApiClient.Builder(context)
    .addApi(Awareness.API)
    .build();
client.connect();
```

- Chiama l'API di tua scelta
- Pare risultato

Un modo semplice per verificare l'autorizzazione utente necessaria è un metodo come questo:

```
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

Ottieni attività utente corrente utilizzando l'API Snapshot

Per le richieste una tantum e non costanti relative all'attività fisica dell'utente, utilizza l'API Snapshot:

```
// 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());
        }
    });
```

Ottieni lo stato delle cuffie con l'API Snapshot

```
// 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));
        }
    });
```

Ottieni la posizione corrente utilizzando l'API Snapshot

```
// 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());
        }
    });
```

Ottieni luoghi nelle vicinanze utilizzando l'API Snapshot

```
// 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");
            }
        }
    });
```

Per quanto riguarda il recupero dei dati in quei luoghi, ecco alcune opzioni:

```
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()));
```

Ottieni meteo attuali usando l'API Snapshot

```
// 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());
            }
        }
    });
```

Ottieni cambiamenti nell'attività dell'utente con l'API di Fence

Se si desidera rilevare quando l'utente inizia o termina un'attività come camminare, correre o qualsiasi altra attività della classe `DetectedActivityFence` , è possibile creare una fence per l'attività che si desidera rilevare e ricevere una notifica all'avvio dell'utente / termina questa attività. Usando un `BroadcastReceiver` , otterrai un `Intent` con i dati che contengono l'attività:

```
// 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);
                }
            }
        });
    }
}
```

Ora puoi ricevere l'intento con un `BroadcastReceiver` per ottenere i callback quando l'utente cambia l'attività:

```
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;
        }
    }
}
```

Ottieni le modifiche per la posizione all'interno di un determinato intervallo utilizzando l'API Fence

Se vuoi rilevare quando il tuo utente entra in una posizione specifica, puoi creare una recinzione per la posizione specifica con un raggio che desideri e ricevere una notifica quando il tuo utente entra o esce dalla posizione.

```
// 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);
                }
            }
        });
    }
}
```

Ora creare un BroadcastReciver per ricevere gli aggiornamenti nello stato utente:

```
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;
        }
    }
}
```

Leggi API di Awareness di Google online: https://riptutorial.com/it/android/topic/3361/api-di-awareness-di-google

Capitolo 20: API di Google Drive

introduzione

Google Drive è un servizio di file hosting creato da **Google** . Fornisce il servizio di archiviazione di file e consente all'utente di caricare file nel cloud e condividere anche con altre persone. Utilizzando l'API di Google Drive, possiamo sincronizzare i file tra computer o dispositivo mobile e Google Drive Cloud.

Osservazioni

legale

Se utilizzi l'API Android di Google Drive nella tua applicazione, devi includere il testo di attribuzione dei servizi di Google Play come parte di una sezione "Note legali" nella tua applicazione.

Si consiglia di includere le note legali come voce di menu indipendente o come parte di una voce di menu "Informazioni".

È possibile effettuare una chiamata a `GooglePlayServicesUtil.getOpenSourceSoftwareLicenseInfo()` per ottenere il testo di attribuzione in fase di esecuzione.

Examples

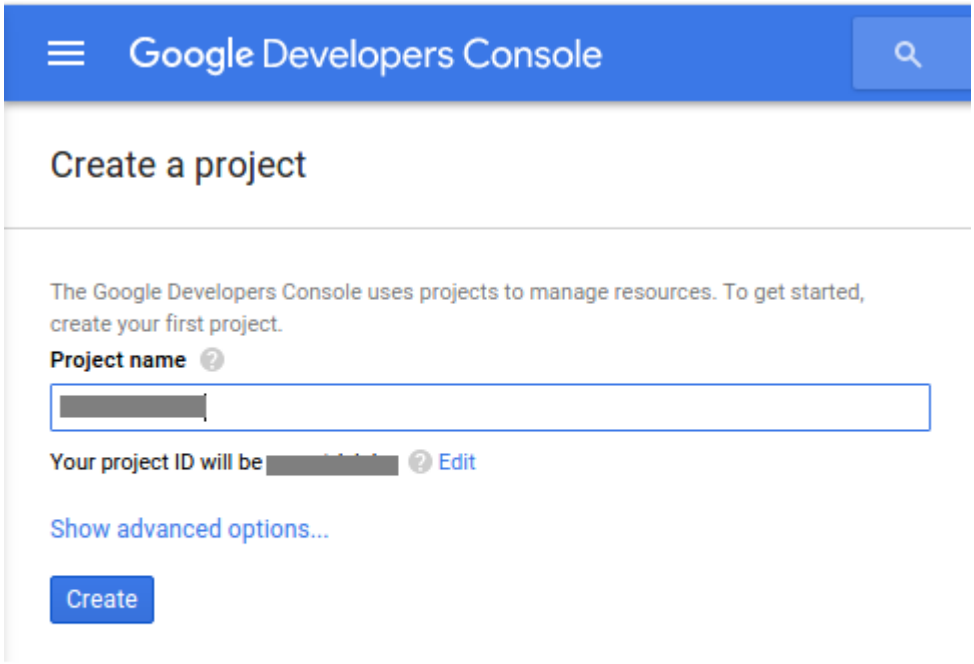
Integra Google Drive in Android

Crea un nuovo progetto su Google Developer Console

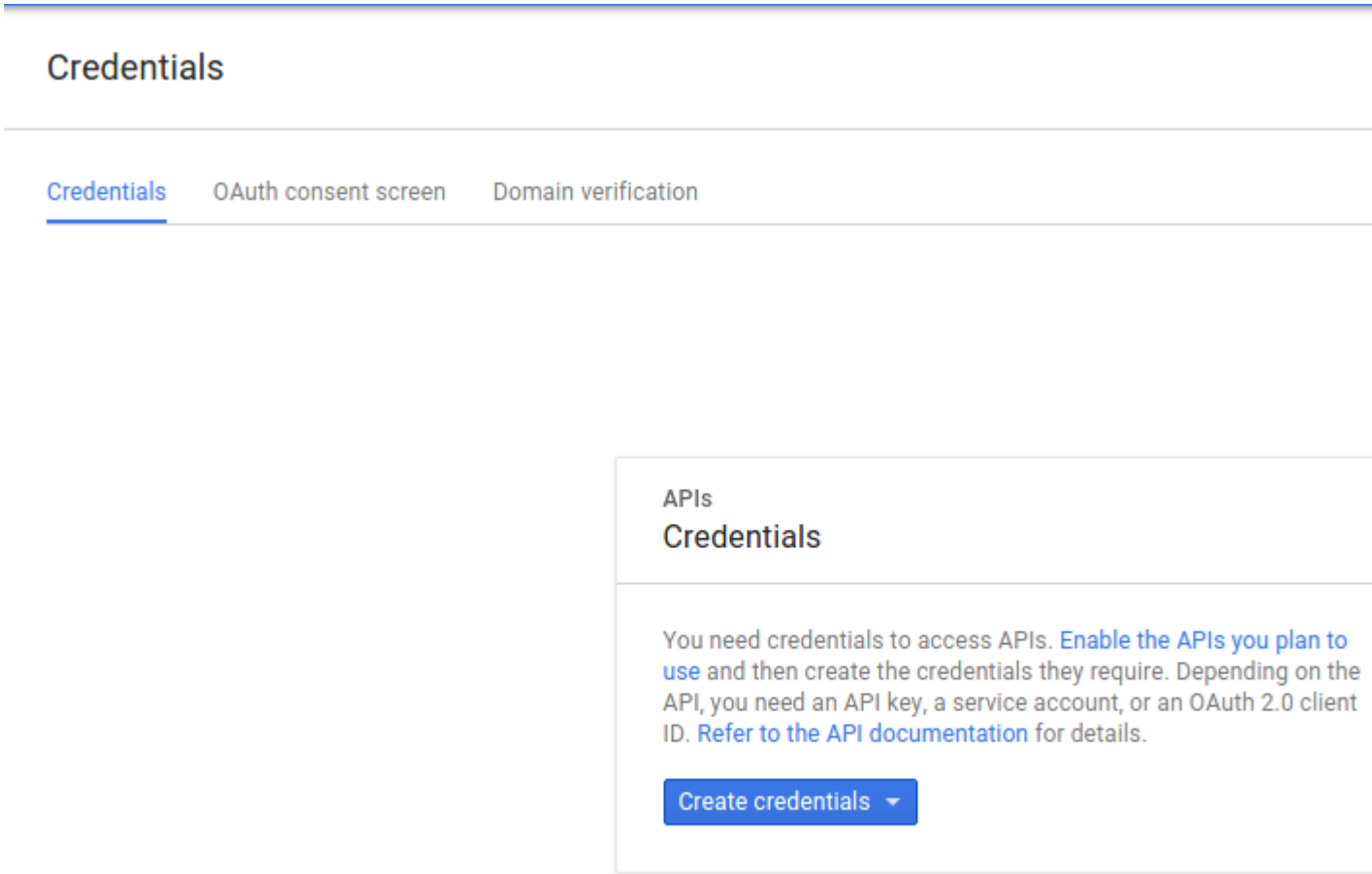
Per integrare l'applicazione Android con Google Drive, crea le credenziali del progetto nella Google Developers Console. Pertanto, dobbiamo creare un progetto sulla Console per gli sviluppatori di Google.

Per creare un progetto su Google Developer Console, procedi nel seguente modo:

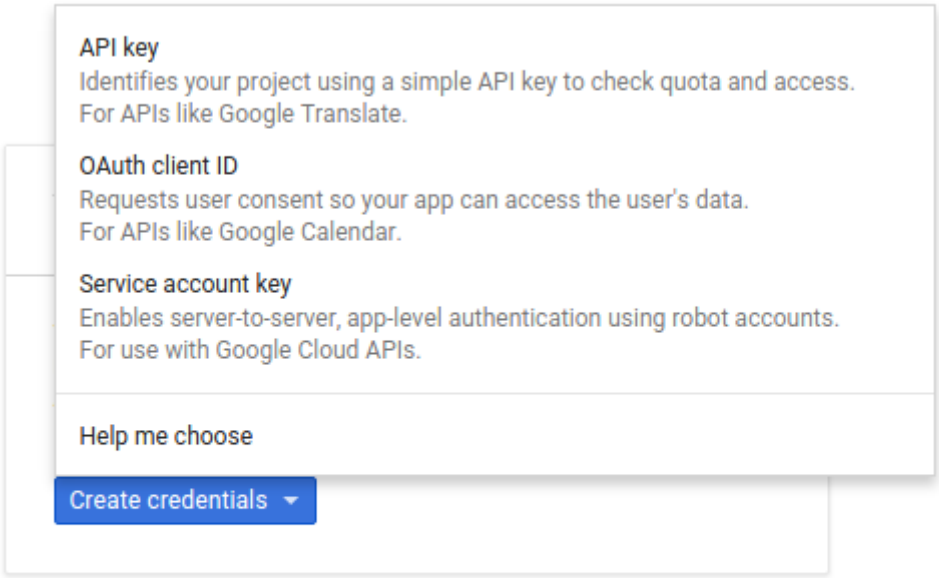
- Vai a Google Developer Console per Android. Riempi il tuo **nome del progetto** nel campo di immissione e fare clic sul pulsante **Crea** per creare un nuovo progetto su console per sviluppatori Google.



- Dobbiamo creare credenziali per accedere all'API. Quindi, fare clic sul pulsante **Crea credenziali** .

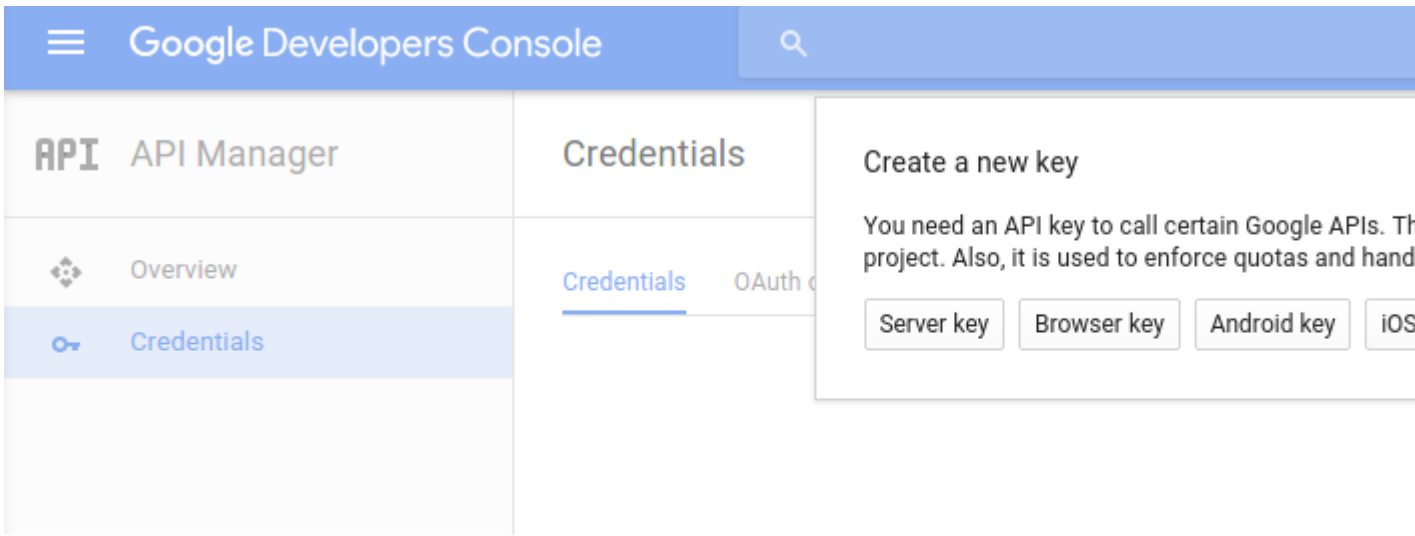


- Ora, si aprirà una finestra pop. Fare clic sull'opzione **Chiave API** nell'elenco per creare la

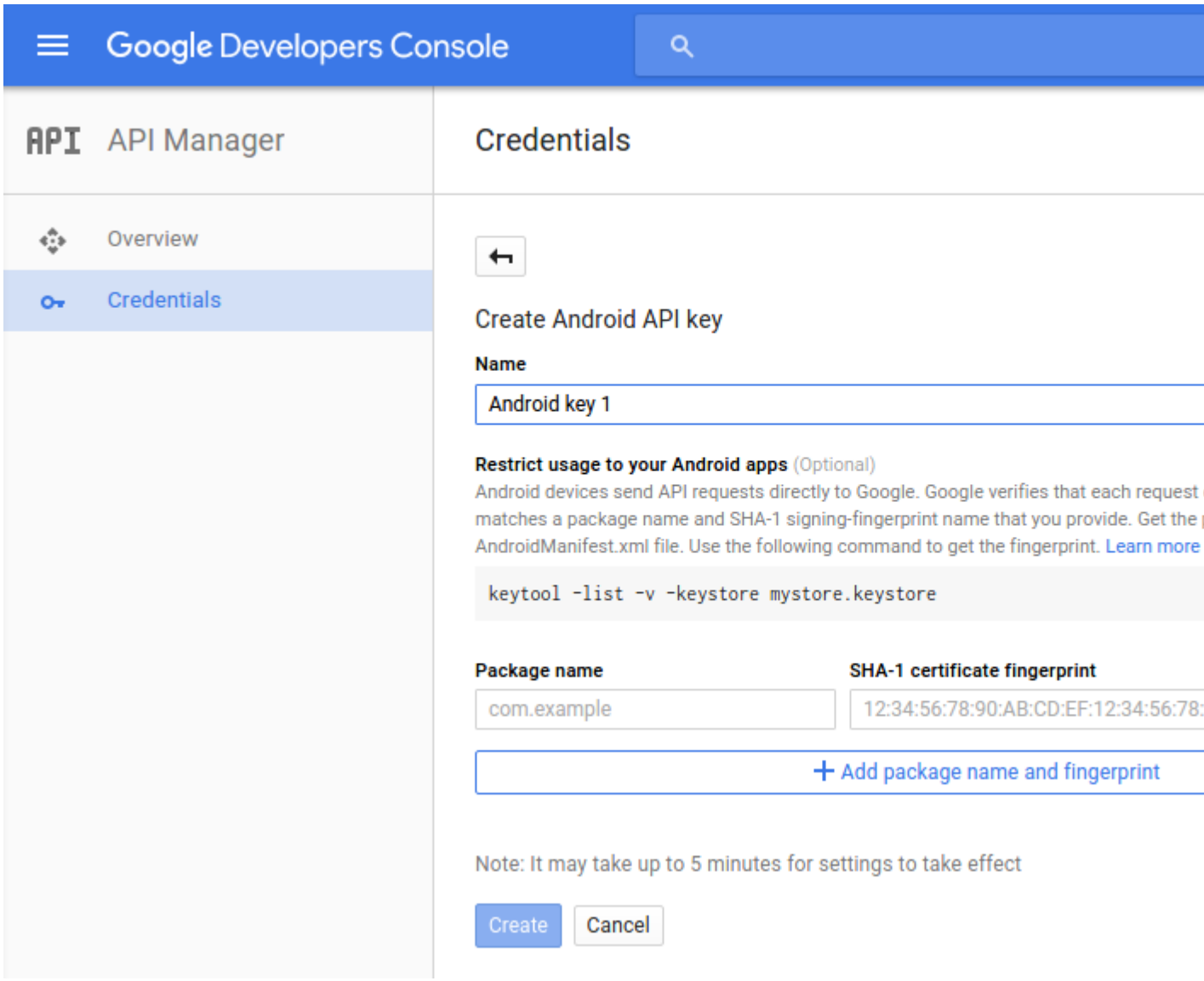


chiave API.

- Abbiamo bisogno di una chiave API per chiamare le API di Google per Android. Quindi, fai clic sulla **chiave Android** per identificare il tuo progetto Android.



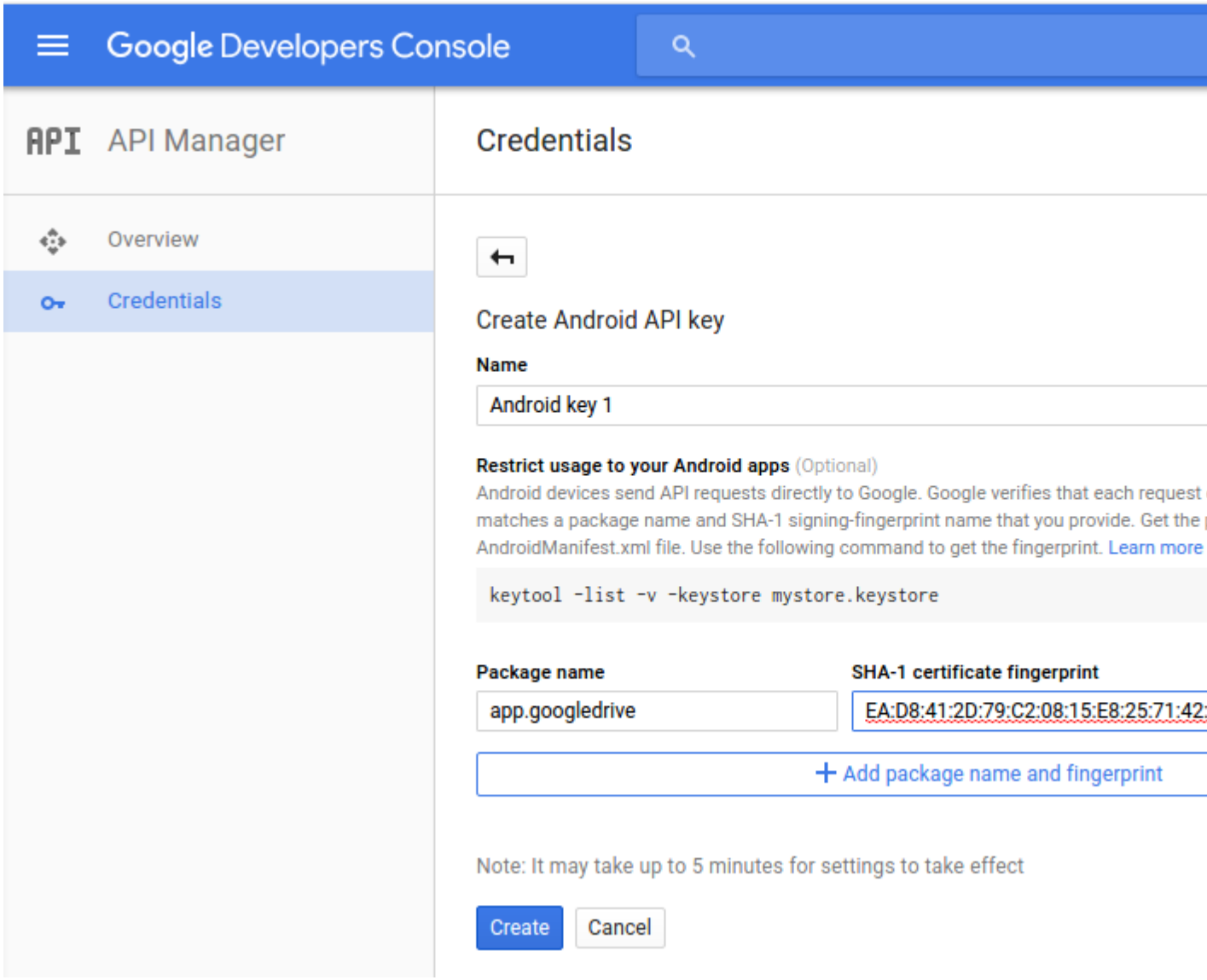
- Successivamente, dobbiamo aggiungere il nome del pacchetto del progetto Android e l' **impronta digitale SHA-1** nei campi di input per creare la chiave API.



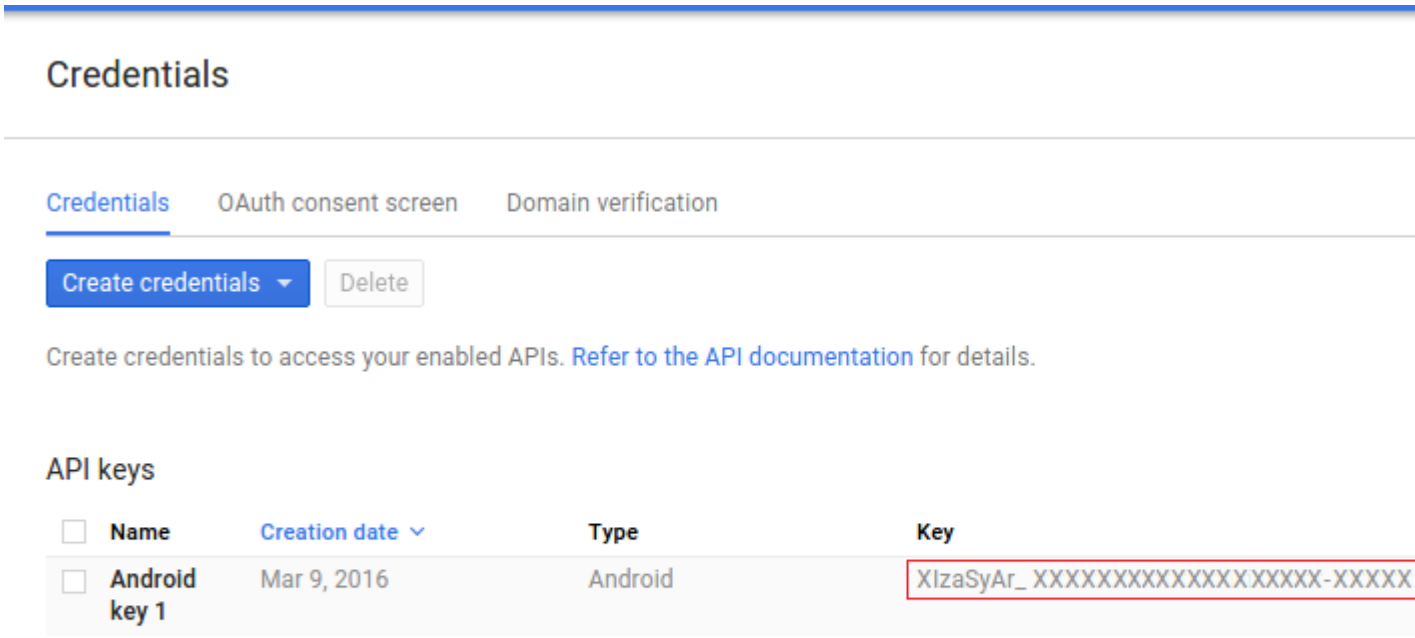
- Abbiamo bisogno di generare **un'impronta digitale SHA-1** . Quindi, apri il tuo terminale ed esegui l' **utilità Keytool** per ottenere l'impronta digitale SHA1. Durante l'esecuzione dell'utilità Keytool, è necessario fornire la **password del keystore** . La password keytool di sviluppo predefinita è **"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
]
]
```

- Ora aggiungi il **nome del pacchetto** e l' **impronta digitale SHA-1** nei campi di inserimento nella pagina delle credenziali. Infine, fai clic sul pulsante Crea per creare la chiave API.



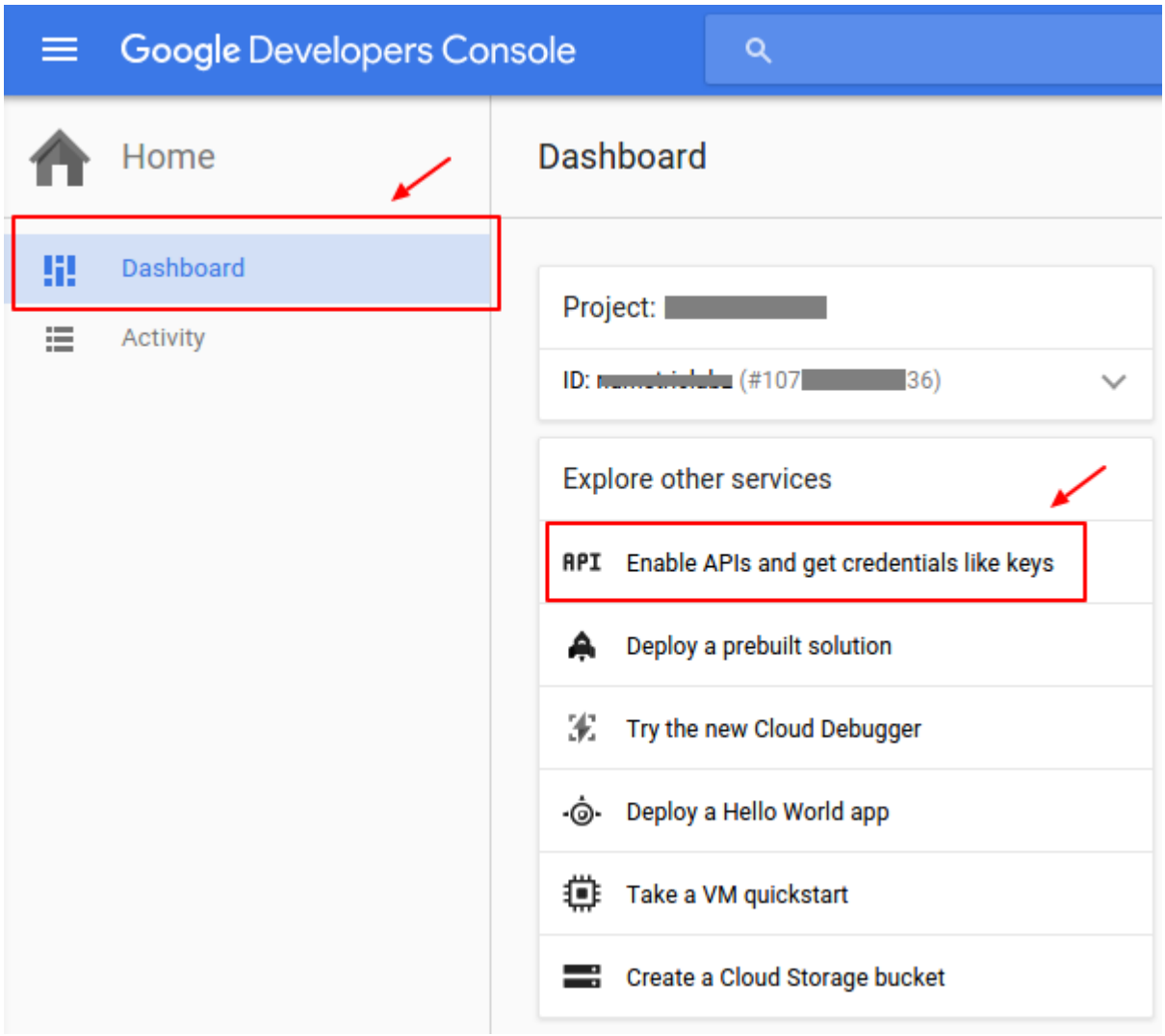
- Questo creerà la chiave API per Android. Utilizzeremo questa chiave API per integrare l'applicazione Android con Google Drive.



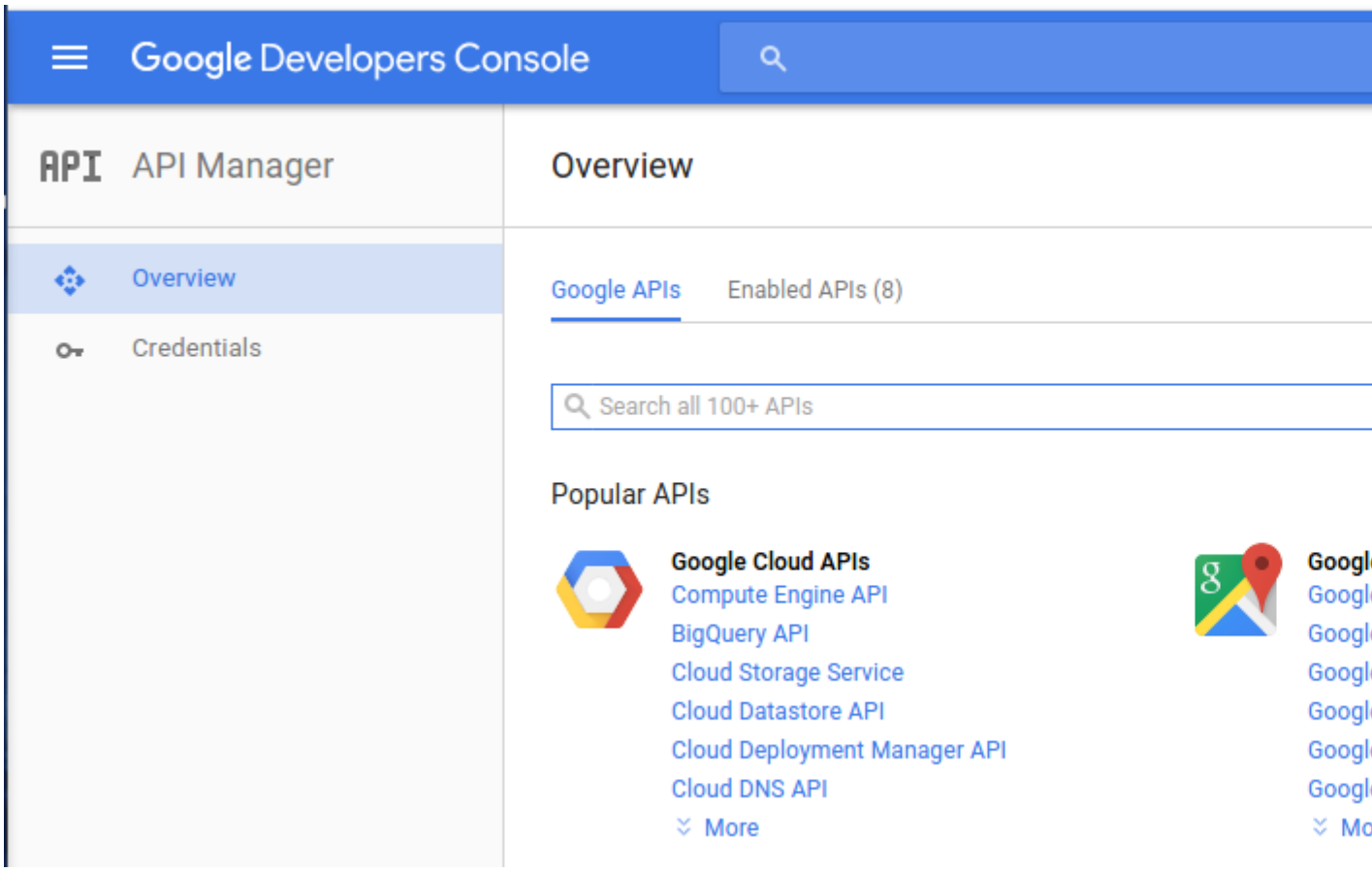
Abilita l'API di Google Drive

Dobbiamo abilitare Google Drive Api per accedere ai file memorizzati su Google Drive dall'applicazione Android. Per abilitare l'API di Google Drive, procedi nel seguente modo:

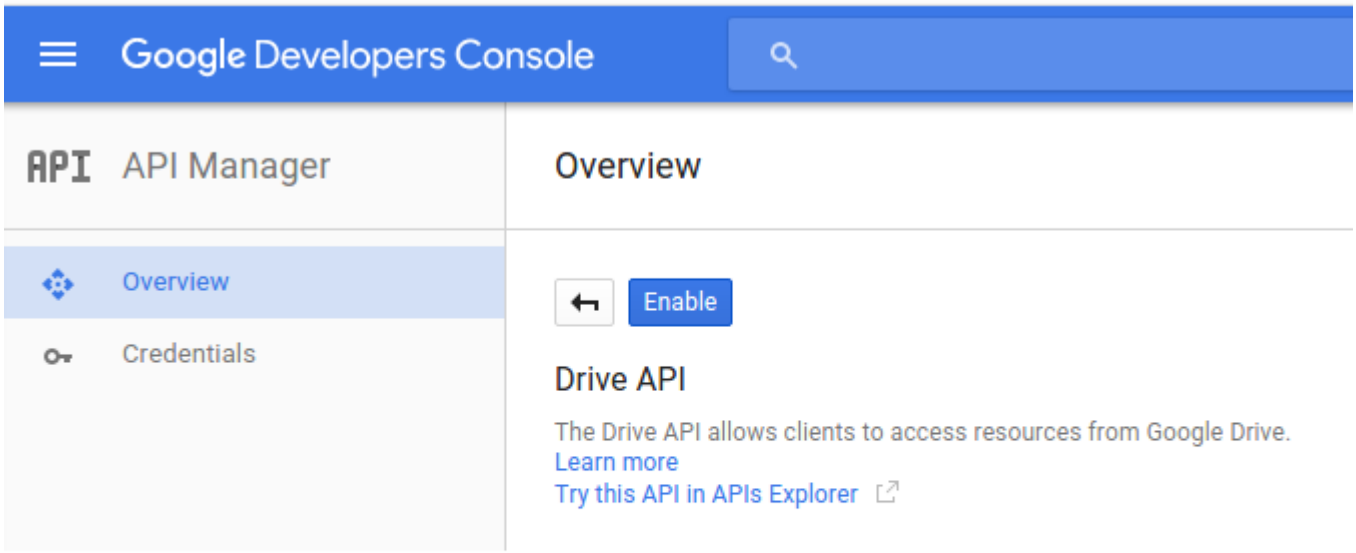
- Vai alla Dashboard della tua Console per gli sviluppatori di Google e fai clic su **Abilita le API per ottenere credenziali come le chiavi,** quindi vedrai le API di Google più diffuse.



- Fai clic sul link **Drive API** per aprire la pagina di panoramica dell'API di Google Drive.



- Fai clic sul pulsante Abilita per abilitare l'API di Google Drive. Consente l'accesso client a Google Drive.



Aggiungi autorizzazione Internet

L'app ha bisogno di accesso a **Internet I** file di Google Drive. Utilizzare il seguente codice per

impostare le autorizzazioni Internet nel file AndroidManifest.xml:

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

Aggiungi Google Play Services

Useremo **Google Play Services API** che include l' **API di Google Drive per Android** . Pertanto, dobbiamo configurare l'SDK dei servizi di gioco di Google nell'applicazione Android. Apri il tuo file `build.gradle` (modulo app) e aggiungi l'SDK di Google Play Services come dipendenze.

```
dependencies {
  ....
    compile 'com.google.android.gms:play-services:<latest_version>'
  ....
}
```

Aggiungi la chiave API nel file manifest

Per utilizzare l'API di Google nell'applicazione Android, è necessario aggiungere la chiave API e la versione del servizio Google Play nel file AndroidManifest.xml. Aggiungi i tag di metadati corretti all'interno del tag del file AndroidManifest.xml.

Collega e autorizza l'API Android di Google Drive

Dobbiamo autenticare e connettere l' **API Android di Google Drive** con l'applicazione Android. L'autorizzazione **dell'API Android** di **Google Drive** è gestita da **GoogleApiClient** . Useremo **GoogleApiClient** all'interno del metodo **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();
    }
```

Disconnetti API Google Deive Android

Quando l'attività si interrompe, disconnetteremo la connessione dell'API Android di Google Drive con l'applicazione Android chiamando il metodo **disconnect ()** all'interno del metodo **onStop ()** dell'attività.

```
@Override
protected void onStop() {
    super.onStop();
    if (mGoogleApiClient != null) {

         // disconnect Google Android Drive API connection.
         mGoogleApiClient.disconnect();
    }
    super.onPause();
}
```

Implementare le chiamate di connessione e il listener non riuscito alla connessione

Implementeremo Connection Callbacks e Connection Failed Listener del client API di Google nel file MainActivity.java per conoscere lo stato della connessione del client API di Google. Questi listener forniscono il **metodo onConnected (), onConnectionFailed (), onConnectionSuspended ()** per gestire i problemi di connessione tra app e Drive.

Se l'utente ha autorizzato l'applicazione, viene invocato il metodo **onConnected ()** . Se l'utente non ha autorizzato l'applicazione, viene invocato il metodo **onConnectionFailed ()** e all'utente viene visualizzata una finestra di dialogo in cui l'app non è autorizzata ad accedere a Google Drive. Nel caso in cui la connessione sia sospesa, viene chiamato il metodo **onConnectionSuspended ()** .

È necessario implementare **ConnectionCallbacks** e **OnConnectionFailedListener** nella propria attività. Usa il seguente codice nel tuo file 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");
    }
```

Crea un file su Google Drive

Aggiungeremo un file su Google Drive. Useremo il metodo `createFile()` di un oggetto `Drive` per creare un file a livello di codice su Google Drive. In questo esempio stiamo aggiungendo un nuovo file di testo nella cartella principale dell'utente. Quando viene aggiunto un file, è necessario specificare il set iniziale di metadati, contenuto del file e cartella principale.

Dobbiamo creare un metodo di callback `CreateMyFile()` e all'interno di questo metodo, utilizzare l'oggetto `Drive` per recuperare una risorsa `DriveContents` . Quindi passiamo il client API all'oggetto `Drive` e richiamiamo il metodo callback `driveContentsCallback` per gestire il risultato di `DriveContents` .

Una risorsa `DriveContents` contiene una copia temporanea del flusso binario del file, disponibile solo per l'applicazione.

```
public void CreateMyFile(){
    fileOperation = true;
    // Create new contents resource.
    Drive.DriveApi.newDriveContents(mGoogleApiClient)
                  .setResultCallback(driveContentsCallback);
}
```

Gestore dei risultati di DriveContents

Per gestire la risposta è necessario verificare se la chiamata è andata a buon fine o meno. Se la

chiamata ha avuto successo, possiamo recuperare la risorsa `DriveContents` .

Creeremo un gestore di risultati di `DriveContents` . All'interno di questo metodo, chiamiamo il metodo `CreateFileOnGoogleDrive()` e passiamo il risultato di `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);
                    }
                }
            }
        };
```

Crea file a livello di programmazione

Per creare file, è necessario utilizzare un oggetto `MetadataChangeSet` . Usando questo oggetto, impostiamo il titolo (nome del file) e il tipo di file. Inoltre, dobbiamo utilizzare il metodo `createFile()` della classe `DriveFolder` e passare l'API del client Google, l'oggetto `MetaDataChangeSet` e il `driveContents` per creare un file. Chiamiamo il callback del gestore di risultati per gestire il risultato del file creato.

Usiamo il seguente codice per creare un nuovo file di testo nella cartella principale dell'utente:

```
/**
 * 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();
}
```

Gestire il risultato del file creato

Il codice seguente creerà un metodo di callback per gestire il risultato del file creato:

```
/**
 * 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;
            }
        };
```

Leggi API di Google Drive online: https://riptutorial.com/it/android/topic/10646/api-di-google-drive

Capitolo 21: API di Google Maps v2 per Android

Parametri

Parametro	Dettagli
Google Map	`GoogleMap` è un oggetto che viene ricevuto su un evento `onMapReady()`
MarkerOptions	`MarkerOptions` è la classe builder di un `Marker` e viene utilizzata per aggiungere un marker a una mappa.

Osservazioni

Requisiti

1. L'SDK di Google Play Services è installato.
2. Un account Google Console.
3. Una chiave API di Google Maps ottenuta in Google Console.

Examples

Attività predefinita di Google Maps

Questo codice attività fornirà funzionalità di base per includere una mappa di Google utilizzando un SupportMapFragment.

L'API di Google Maps V2 include un modo completamente nuovo di caricare mappe.

Le attività ora devono implementare l'interfaccia **OnMapReadyCallBack** , che viene fornita con un metodo **overMate su onMapReady ()** che viene eseguito ogni volta che viene eseguito **SupportMapFragment** . **getMapAsync (OnMapReadyCallback)** ; e la chiamata è stata completata con successo.

Mappe usare **marcatori** , **poligoni** e **polilinee** per visualizzare le informazioni interattivo per l'utente.

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));
    }
}
```

Si noti che il codice sopra gonfia un layout, che ha un SupportMapFragment nidificato all'interno del layout del contenitore, definito con un ID di `R.id.map` . Il file di layout è mostrato di seguito:

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>
```

Stili di Google Maps personalizzati

Stile della mappa

Google Maps include una serie di stili diversi da applicare, utilizzando questo codice:

```
// Sets the map type to be "hybrid"
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

I diversi stili di mappa sono:

Normale

```
map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
```

Tipica mappa stradale. Vengono mostrate strade, alcune caratteristiche create dall'uomo e importanti caratteristiche naturali come i fiumi. Sono inoltre visibili le etichette stradali e delle funzionalità.

Ibrido

```
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

Dati di fotografie satellitari con mappe stradali aggiunte. Sono inoltre visibili le etichette stradali e delle funzionalità.

Satellitare

```
map.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
```

Dati di fotografia satellitare Le etichette stradali e le caratteristiche non sono visibili.

Terreno

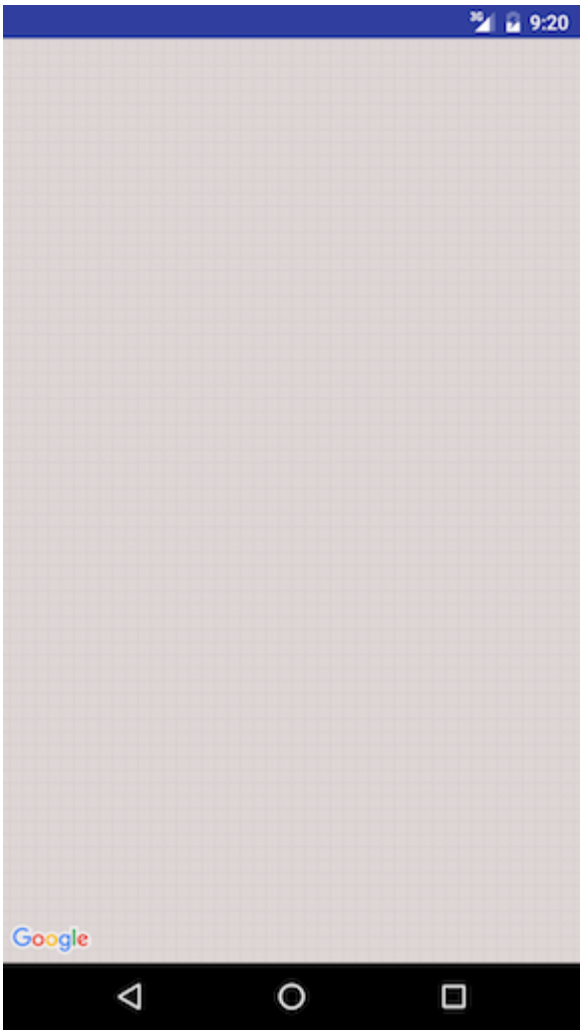
```
map.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

Dati topografici La mappa include i colori, le linee di contorno e le etichette e l'ombreggiatura prospettica. Sono anche visibili alcune strade ed etichette.

Nessuna

```
map.setMapType(GoogleMap.MAP_TYPE_NONE);
```

Nessuna tessera. La mappa verrà visualizzata come una griglia vuota senza tessere caricate.



ALTRE OPZIONI DI STILE

Mappe dell'interno

A livelli di zoom elevati, la mappa mostrerà le planimetrie per gli spazi interni. Queste sono chiamate mappe interne e sono visualizzate solo per i tipi di mappe "normali" e "satellite".

per abilitare o disabilitare le mappe interne, ecco come è fatto:

```
GoogleMap.setIndoorEnabled(true).
GoogleMap.setIndoorEnabled(false).
```

Possiamo aggiungere stili personalizzati alle mappe.

Nel metodo onMapReady aggiungi il seguente snippet di codice

```
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);
    }
```

sotto la cartella *res* creare un nome di cartella *raw* e aggiungere il file json di stile. Esempio di file style.json

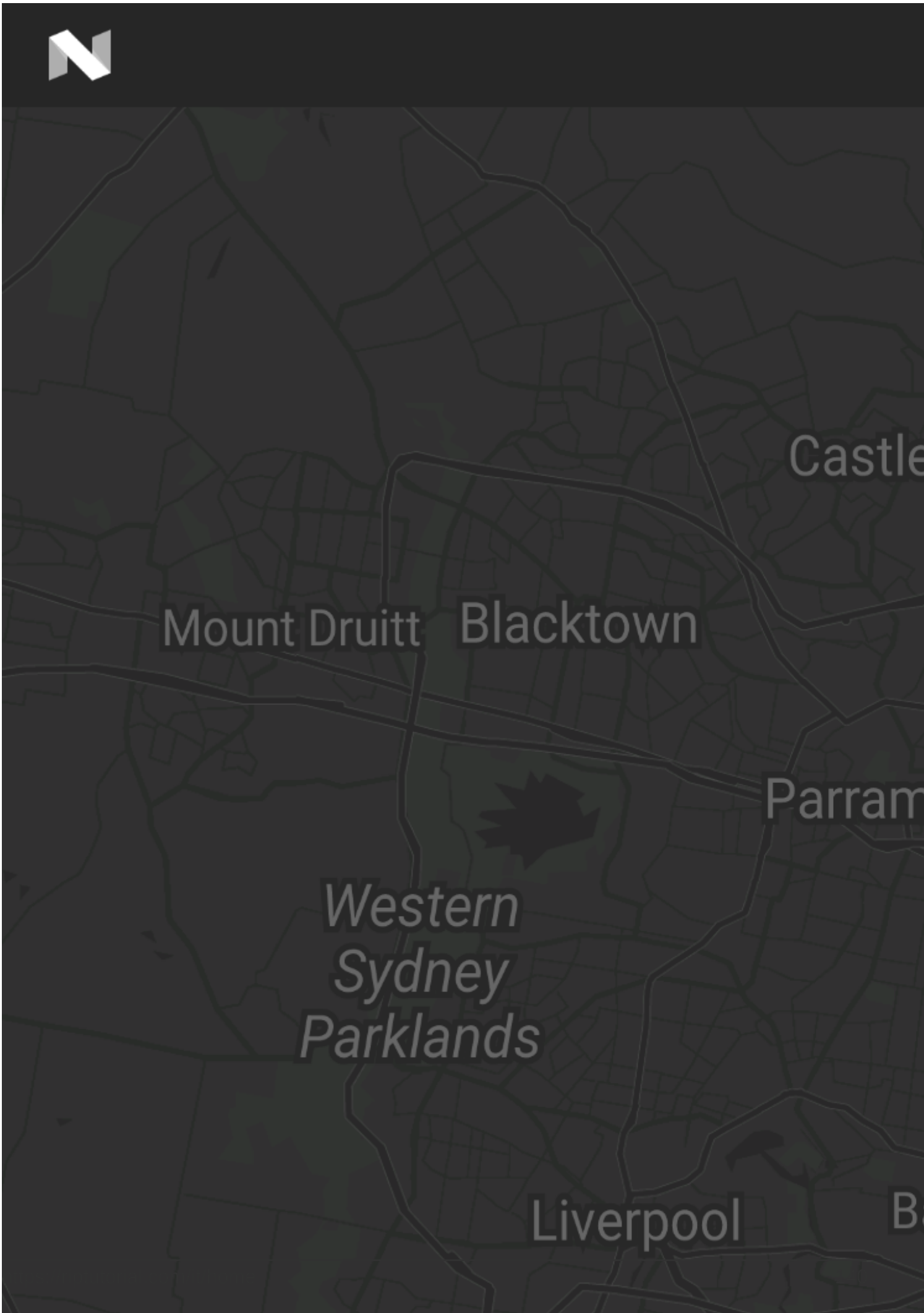
```
  [
{
  "featureType": "all",
  "elementType": "geometry",
  "stylers": [
    {
      "color": "#242f3e"
    }
  ]
},
{
  "featureType": "all",
  "elementType": "labels.text.stroke",
  "stylers": [
    {
      "lightness": -80
    }
  ]
},
{
  "featureType": "administrative",
  "elementType": "labels.text.fill",
  "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
      }
    ]
  }
]
```

Per generare stili file json, fare clic su questo collegamento


Castle
Mount Druitt
Blacktown
Parram
Western
Sydney
Parklands
Liverpool

Objects, possiamo farlo in questo modo.

La classe titolare di `MyLocation` :

```
public class MyLocation {
  LatLng latLng;
  String title;
  String snippet;
}
```

Ecco un metodo che richiede un elenco di oggetti `MyLocation` e inserisce un indicatore per ognuno:

```
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: ai fini di questo esempio, `mMap` è una variabile membro della classe dell'attività, in cui l'abbiamo assegnata al riferimento mappa ricevuto nella sovrascrittura `onMapReady()` .

MapView: incorporare un GoogleMap in un layout esistente

È possibile trattare un GoogleMap come una vista Android se utilizziamo la classe MapView fornita. Il suo utilizzo è molto simile a MapFragment.

Nel tuo layout usa MapView come segue:

```
<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 -->
    />
```

La tua attività deve implementare l'interfaccia OnMapReadyCallback per funzionare:

```
/**
* 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);
    }
}
```

Mostra posizione corrente in una mappa di Google

Ecco una classe di attività completa che colloca un indicatore nella posizione corrente e sposta la telecamera nella posizione corrente.

Ci sono alcune cose che succedono in sequenza qui:

- Verifica l'autorizzazione di posizione
- Una volta concesso il permesso di posizione, chiama `setMyLocationEnabled()` , crea GoogleApiClient e collegalo
- Una volta connesso GoogleApiClient, richiedi aggiornamenti sulla posizione

```
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_M
AGENTA));
        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) {
```

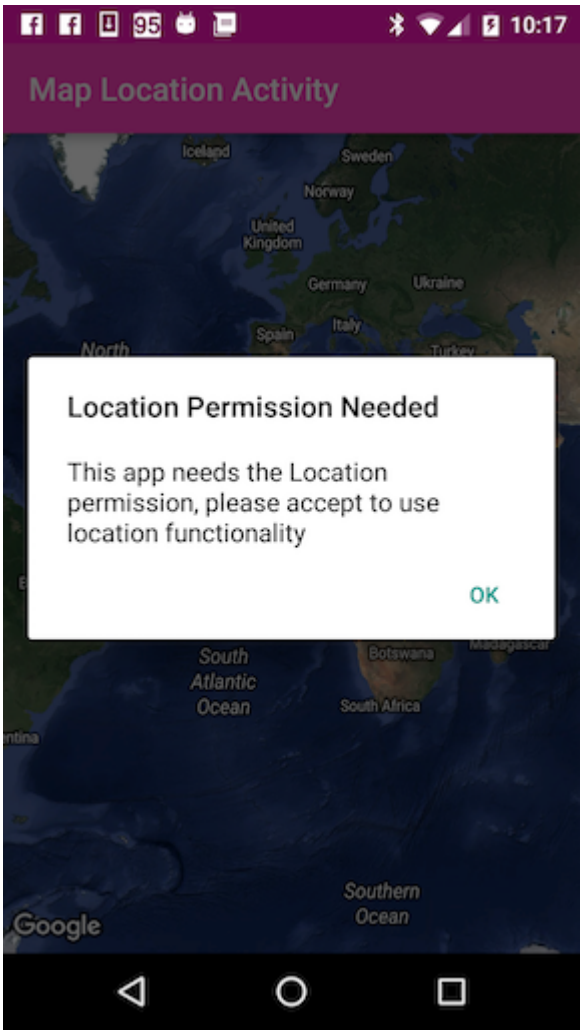
```
                    // permission was granted, yay! Do the
                    // location-related task you need to do.
                    if (ContextCompat.checkSelfPermission(this,
                            Manifest.permission.ACCESS_FINE_LOCATION)
                            == PackageManager.PERMISSION_GRANTED) {

                        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:

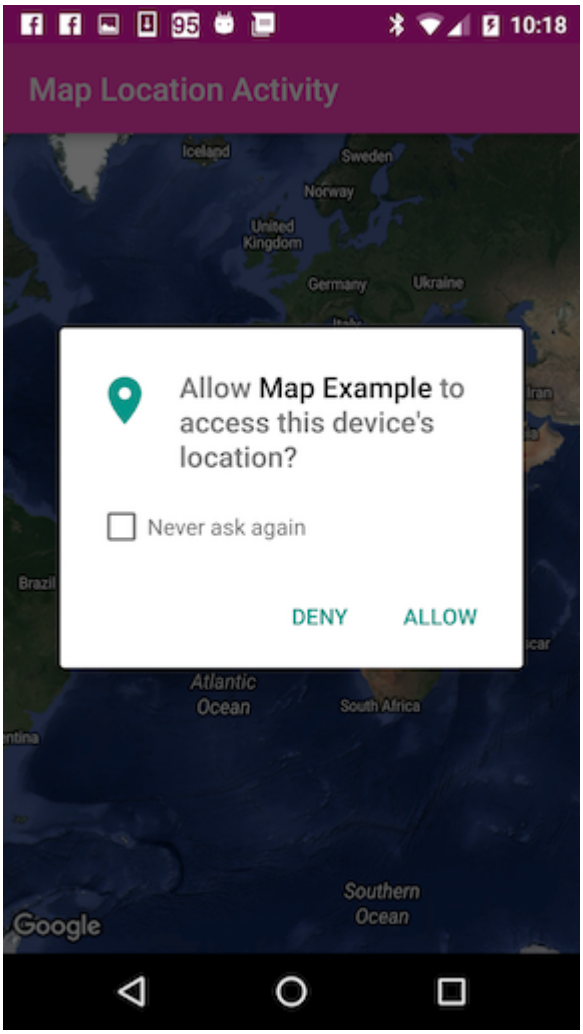
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <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>
```

Risultato:

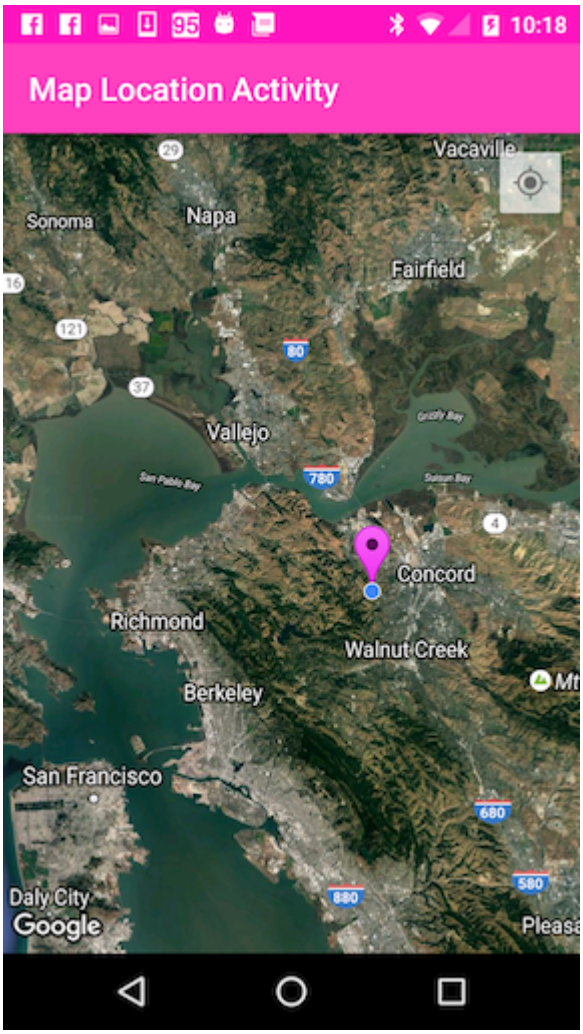
Mostra spiegazione se necessario su Marshmallow e Nougat utilizzando un AlertDialog (questo caso si verifica quando l'utente ha precedentemente negato una richiesta di autorizzazione o ha concesso l'autorizzazione e successivamente lo ha revocato nelle impostazioni):



Richiedi all'utente l'autorizzazione di posizione su Marshmallow e Nougat chiamando `ActivityCompat.requestPermissions()` :



Sposta la videocamera nella posizione corrente e posiziona il marcatore quando viene concessa l'autorizzazione di posizione:



Ottenimento dell'impronta SH1 del file del keystore del certificato

Per ottenere una chiave API di Google Maps per il tuo certificato, devi fornire alla console API l'impronta SH1 del tuo keystore di debug / release.

È possibile ottenere il keystore utilizzando il programma **keytool di JDK** come descritto qui nella documentazione.

Un altro approccio è ottenere l'impronta digitale a livello di codice eseguendo questo frammento con l'app firmata con il certificato di debug / release e stampando l'hash nel log.

```
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());
}
```

Non lanciare Google Maps quando si fa clic sulla mappa (modalità lite)

Quando una mappa di Google viene visualizzata in modalità lite, facendo clic su una mappa si aprirà l'applicazione Google Maps. Per disabilitare questa funzionalità è necessario chiamare `setClickable(false)` su `MapView` , *ad esempio* :

```
final MapView mapView = (MapView)view.findViewById(R.id.map);
mapView.setClickable(false);
```

UISettings

Utilizzando `UISettings` , è possibile modificare l'aspetto di Google Map.

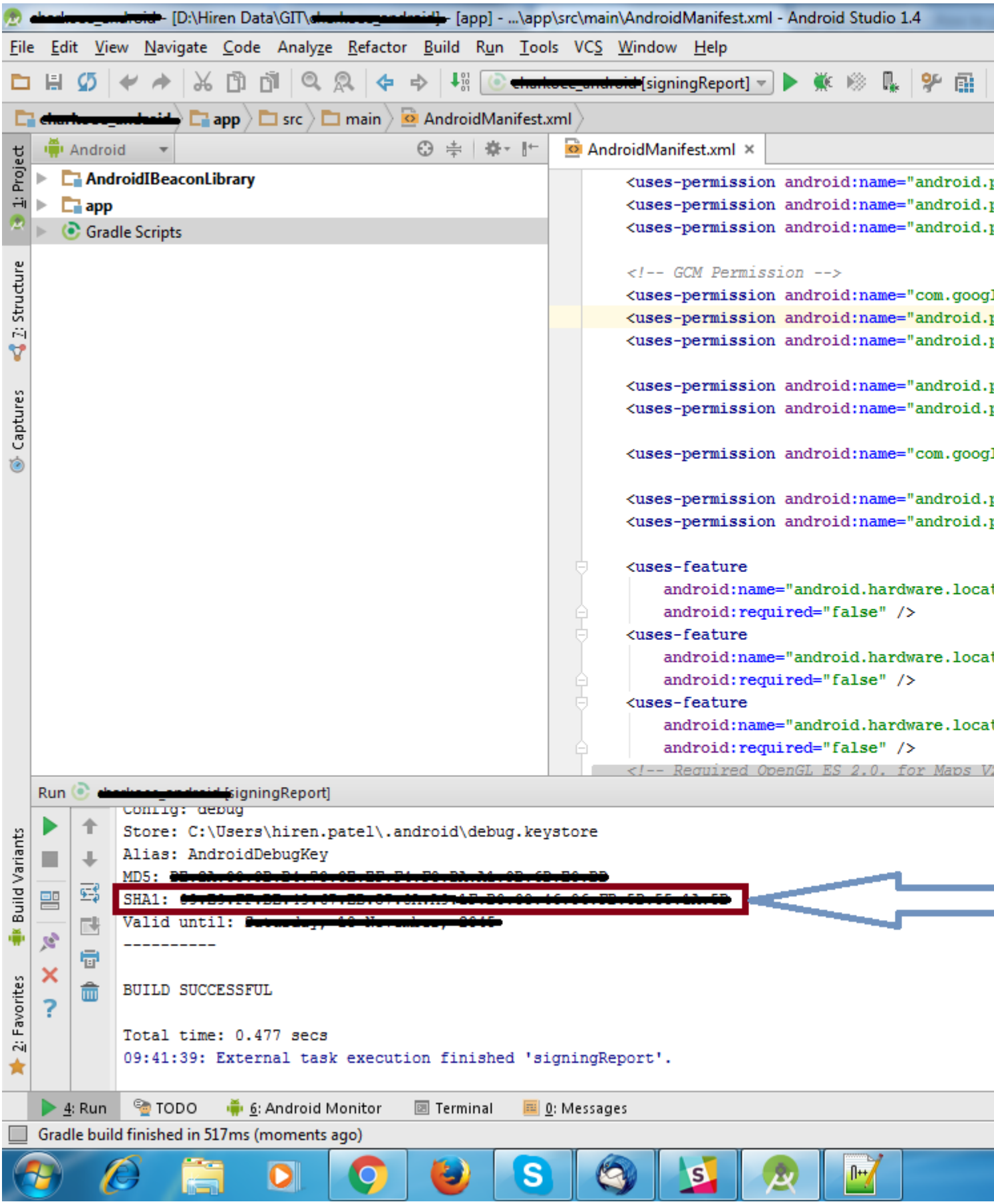
Ecco un esempio di alcune impostazioni comuni:

```
    mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
    mGoogleMap.getUiSettings().setMapToolbarEnabled(true);
    mGoogleMap.getUiSettings().setZoomControlsEnabled(true);
    mGoogleMap.getUiSettings().setCompassEnabled(true);
```

Risultato:

Ottieni l'impronta digitale SHA1 di debug

1. Apri Android Studio
2. Apri il tuo progetto
3. Clicca su Gradle (dal pannello laterale destro, vedrai **Gradle Bar**)
4. Fai clic su Aggiorna (fai clic su Aggiorna dalla **barra del gradino** , vedrai gli script del Gradle della **lista** del tuo progetto)
5. Clicca sul tuo progetto (il tuo **elenco dei** nomi dei progetti (root))
6. Clicca su Attività
7. Clicca su Android
8. Doppio clic su signingReport (otterrai **SHA1** e **MD5** nella **barra di esecuzione**)



Listener del clic di InfoWindow

Ecco un esempio di come definire un'azione diversa per ogni evento clickWindow di ogni Marker.

Utilizzare una HashMap in cui l'ID del marker è la chiave e il valore è l'azione corrispondente che deve essere eseguita quando si fa clic su InfoWindow.

Quindi, utilizzare `OnInfoWindowClickListener` per gestire l'evento di un utente facendo clic su InfoWindow e utilizzare HashMap per determinare l'azione da intraprendere.

In questo semplice esempio verrà aperta un'attività diversa in base alla quale è stato fatto clic su InfoWindow di Marker.

Dichiarare HashMap come variabile di istanza dell'attività o Frammento:

```
 //Declare HashMap to store mapping of marker to Activity
HashMap<String, String> markerMap = new HashMap<String, String>();
```

Quindi, ogni volta che aggiungi un marcatore, crea una voce in HashMap con l'ID marcatore e l'azione che deve compiere quando si fa clic su InfoWindow.

Ad esempio, aggiungendo due marcatori e definendo un'azione da intraprendere per ciascuno:

```
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");
```

Nel listener dei clic di InfoWindow, ottieni l'azione da HashMap e apri l'attività corrispondente in base all'azione del Marcatore:

```
 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 Se il codice si trova in un frammento, sostituire MainActivity.this con getActivity ().

Modifica offset

Cambiando MapPoint valori *xey* di cui hai bisogno è possibile cambiare possition offset del google map, per impostazione predefinita, sarà al centro della mappa. Chiama sotto il metodo dove vuoi cambiarlo! Meglio usarlo all'interno di `onLocationChanged` come
`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

        }
```

Leggi API di Google Maps v2 per Android online: https://riptutorial.com/it/android/topic/170/api-di-google-maps-v2-per-android

Capitolo 22: API di impronte digitali in Android

Osservazioni

Guarda anche

Progetto di esempio Github

Blogspot per sviluppatori Android

Sito per sviluppatori Android

Examples

Aggiunta di Fingerprint Scanner nell'applicazione Android

Android supporta l'impronta delle impronte digitali da Android 6.0 (Marshmallow) SDK 23

> Per utilizzare questa funzione nella tua app, aggiungi innanzitutto l'autorizzazione USE_FINGERPRINT nel tuo manifest.

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

Ecco la procedura da seguire

> Innanzitutto è necessario creare una chiave simmetrica nel Key Store di Android utilizzando KeyGenerator che può essere utilizzata solo dopo che l'utente si è autenticato con l'impronta digitale e passa 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();
```

> Impostando KeyGenParameterSpec.Builder.setUserAuthenticationRequired su true, è possibile consentire l'uso della chiave solo dopo che l'utente l'ha autenticato, incluso quando è stato autenticato con l'impronta digitale dell'utente.

```
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);
```

> Quindi iniziare ad ascoltare un'impronta digitale sul sensore dell'impronta digitale chiamando FingerprintManager.authenticate con un Cipher inizializzato con la chiave simmetrica creata. In alternativa, puoi ricorrere alla password verificata sul lato server come autenticatore.

Crea e inizializza `FingerprintManger` da `fingerprintManger.class`

```
getContext().getSystemService(FingerprintManager.class)
```

Per autenticare usa `FingerprintManger` api e crea sottoclasse usando

`FingerprintManager.AuthenticationCallback` e sovrascrive i metodi

```
onAuthenticationError
onAuthenticationHelp
onAuthenticationSucceeded
onAuthenticationFailed
```

Iniziare

Per iniziare Lasciando il metodo di autenticazione della chiamata evento fingerPrint con cripto

```
fingerprintManager
              .authenticate(cryptoObject, mCancellationSignal, 0 , this, null);
```

Annulla

per smettere di ascoltare la chiamata scanner

```
android.os.CancellationSignal;
```

> Una volta verificata l'impronta digitale (o la password), viene richiamato il callback di FingerprintManager.AuthenticationCallback # onAuthenticationSucceeded ().

```
@Override

public void onAuthenticationSucceeded(AuthenticationResult result) {

            }
```

Come utilizzare Android Fingerprint API per salvare le password degli utenti

Questo esempio di classe helper interagisce con il gestore di impronte digitali ed esegue la crittografia e la decrittografia della password. Si noti che il metodo utilizzato per la crittografia in

questo esempio è AES. Questo non è l'unico modo per crittografare e esistono altri esempi . In questo esempio i dati vengono crittografati e decrittografati nel modo seguente:

crittografia:

1. L'utente assegna l'helper alla password non crittografata desiderata.
2. L'utente è tenuto a fornire un'impronta digitale.
3. Una volta autenticato, l'helper ottiene una chiave dal `KeyStore` e crittografa la password utilizzando un `Cipher` .
4. La password e il sale IV (IV viene ricreato per ogni crittografia e non viene riutilizzato) vengono salvati nelle preferenze condivise per essere utilizzati successivamente nel processo di decrittografia.

decrittazione:

1. L'utente richiede di decrittografare la password.
2. L'utente è tenuto a fornire un'impronta digitale.
3. L'helper crea un `Cipher` utilizzando l'IV e una volta che l'utente è autenticato, il KeyStore ottiene una chiave dal `KeyStore` e decifra la password.

```
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());
            }
        }
    }
}
```

Questa attività sotto è un esempio molto semplice di come ottenere una password salvata dall'utente e di interagire con l'helper.

```
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);
            }
        };
    }
}
```

Leggi API di impronte digitali in Android online: https://riptutorial.com/it/android/topic/7523/api-di-impronte-digitali-in-android

Capitolo 23: API di Twitter

Examples

Creazione di login con il pulsante twitter e allegare una richiamata ad esso

1. All'interno del layout, aggiungi un pulsante Accedi con il seguente codice:

```
<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. Nell'attività o nel frammento che visualizza il pulsante, è necessario creare e allegare una richiamata al pulsante di accesso come segue:

```
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. Passa il risultato dell'attività di autenticazione al pulsante:

```
@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);
}
```

Nota, se si utilizza TwitterLoginButton in un frammento, utilizzare invece i seguenti passaggi:

```
 @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. Aggiungi le seguenti linee alle dipendenze **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;
 }
```

Leggi API di Twitter online: https://riptutorial.com/it/android/topic/4801/api-di-twitter

Capitolo 24: Architettura MVP

introduzione

Questo argomento fornirà l'architettura Model-View-Presenter (MVP) di Android con vari esempi.

Osservazioni

Esistono molti modi per progettare un'app Android. Ma non tutti sono testabili e ci permettono di strutturare il nostro codice in modo che l'app sia facile da testare. L'idea chiave di un'architettura testabile è la separazione di parti dell'applicazione che le rende più facili da mantenere, estendere e testare separatamente l'una dall'altra.

Definizione MVP

Modello

In un'applicazione con una buona architettura a livelli, questo modello sarebbe solo il gateway per il livello di dominio o la logica aziendale. Vedi come il fornitore dei dati che vogliamo visualizzare nella vista.

vista

The View, di solito attuato da `Activity` o di `Fragment` , conterrà un riferimento al *presentatore.* L'unica cosa che farà la vista è chiamare un metodo dal Presenter ogni volta che c'è un'azione di interfaccia.

Presentatore

Il Presenter è responsabile di agire da intermediario tra Visualizza e Modello. Recupera i dati dal modello e lo restituisce formattato nella vista. Ma a differenza del tipico MVC, decide anche cosa succede quando interagisci con la vista.

* Definizioni dell'articolo di Antonio Leiva.

Struttura dell'app *consigliata* (non richiesta)

L'app dovrebbe essere strutturata per pacchetto *per funzionalità* . Ciò migliora la leggibilità e modularizza l'app in modo che parti di esso possano essere modificate indipendentemente l'una dall'altra. Ogni caratteristica chiave dell'app è nel proprio pacchetto Java.

Examples

Esempio di accesso nel modello Model View Presenter (MVP)

Vediamo MVP in azione utilizzando una semplice schermata di accesso. Ci sono due `Button` s-one per l'azione di login e un'altra per una schermata di registrazione; due `EditText` s-one per l'e-mail e l'altro per la password.

LoginFragment (The View)

```
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 (The Presenter)

```
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 (il modello)

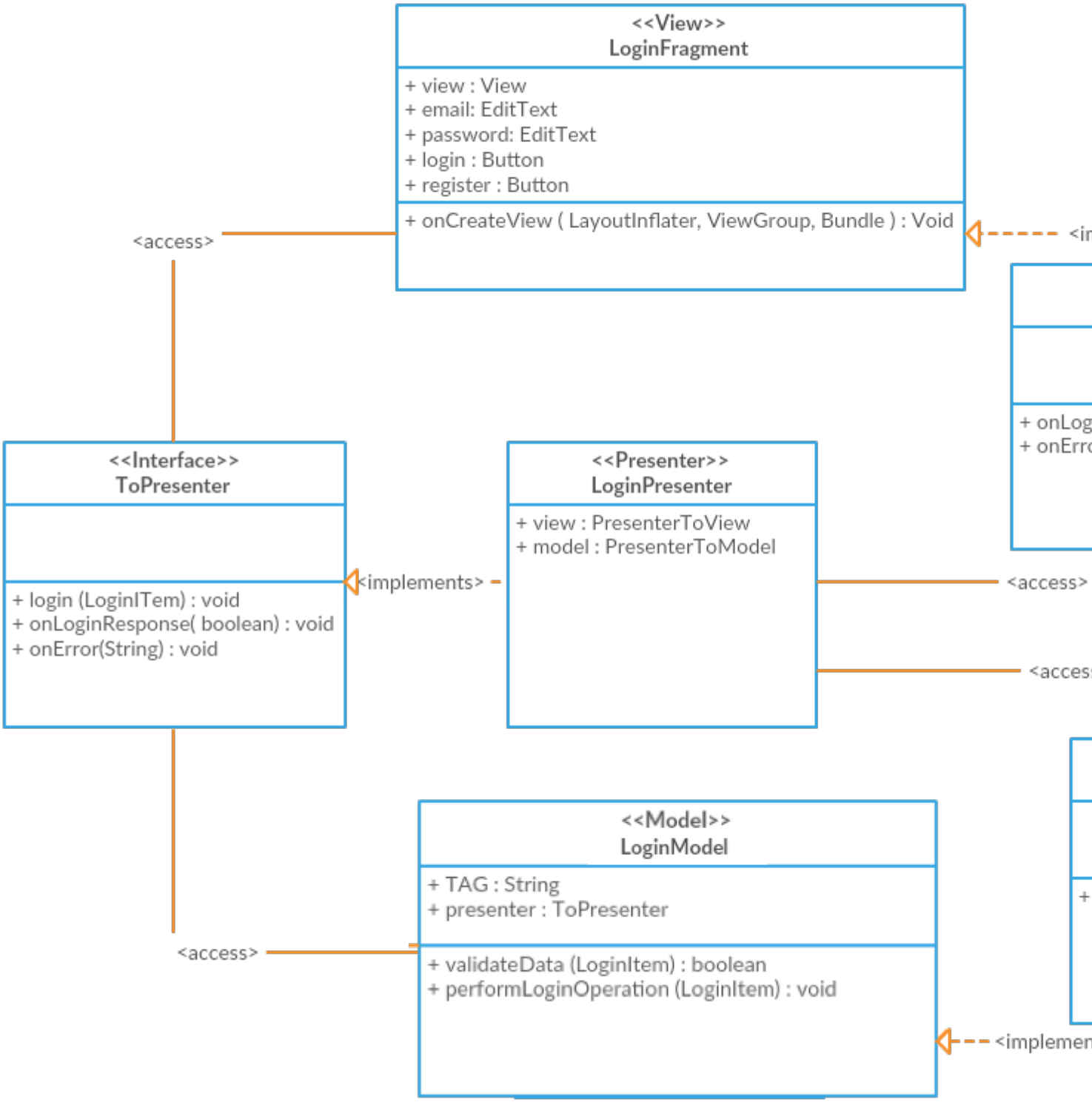
```
public class LoginModel implements LoginContract.PresenterToModel,
```

```
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);
    }
}
```

Diagramma di classe

Vediamo l'azione sotto forma di diagramma di classe.

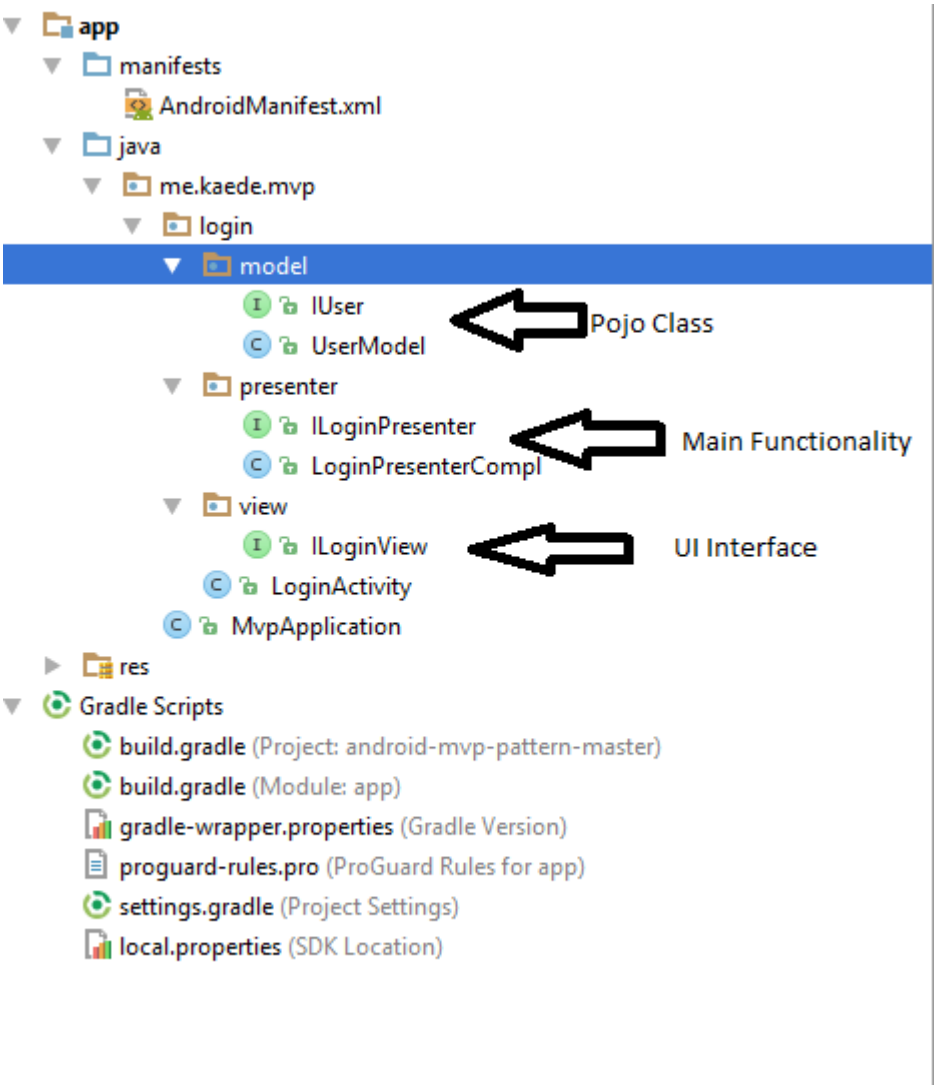


Gli appunti:

- Questo esempio utilizza Volley per le comunicazioni di rete, ma questa libreria non è richiesta per MVP
- `UrlUtils` è una classe che contiene tutti i collegamenti per i miei endpoint API
- `ResponseErrorListener.ErrorListener` è `interface` che ascolta l'errore in `ErrorResponse` che `implements Response.ErrorListener` di Volley; queste classi non sono incluse qui in quanto non fanno direttamente parte di questo esempio

Esempio di accesso semplice in MVP

Struttura del pacchetto richiesta



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>
```

Classe di attività 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);
    }
}
```

Creazione di un'interfaccia ILoginView

Creare un'interfaccia `ILoginView` per le informazioni di aggiornamento dalla cartella di visualizzazione di Presenter come segue:

```
public interface ILoginView {
    public void onClearText();
    public void onLoginResult(Boolean result, int code);
    public void onSetProgressBarVisibility(int visibility);
}
```

Creazione di un'interfaccia ILoginPresenter

Creare un'interfaccia `ILoginPresenter` per comunicare con `LoginActivity` (Visualizzazioni) e creare la classe `LoginPresenterCompl` per gestire la funzionalità di accesso e riferire all'Attività. La classe `LoginPresenterCompl` implementa l'interfaccia `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");
    }
}
```

Creazione di un UserModel

Crea un `UserModel` che è come una classe `LoginActivity` per `LoginActivity` . Creare un'interfaccia `IUser` per le convalide 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;
}
```

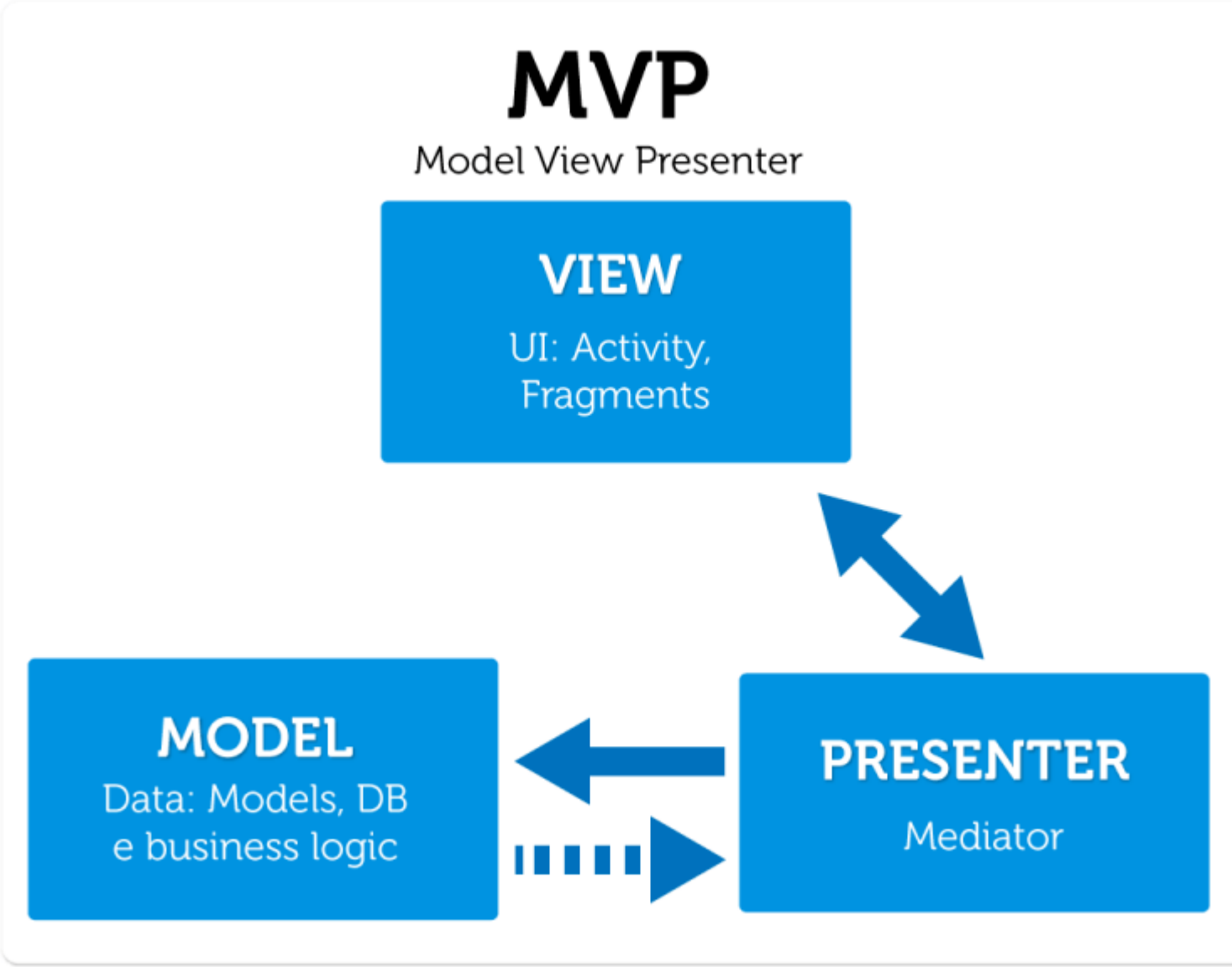
IUser.class

```
public interface IUser {
    String getName();

    String getPasswd();

    int checkUserValidity(String name, String passwd);
}
```

MVP

Un Model-view-presenter (MVP) è una derivazione del modello architettonico model-view-controller (MVC). Viene utilizzato principalmente per la creazione di interfacce utente e offre i seguenti vantaggi:

- Le viste sono più separate dai modelli. Il Presentatore è il mediatore tra Modello e Vista.
- È più facile creare test unitari.
- Generalmente, esiste una mappatura uno-a-uno tra View e Presenter, con la possibilità di utilizzare più Presenter per viste complesse.



Leggi Architettura MVP online: https://riptutorial.com/it/android/topic/4615/architettura-mvp

Capitolo 25: AsyncTask

Parametri

Parametro	Dettagli
Parametri	il tipo di parametri inviati all'attività al momento dell'esecuzione.
Progresso	il tipo di unità di progresso pubblicate durante il calcolo dello sfondo
Risultato	il tipo del risultato del calcolo dello sfondo.

Examples

Uso di base

In Attività e servizi Android, la maggior parte dei callback vengono eseguiti sul thread principale . Ciò semplifica l'aggiornamento dell'interfaccia utente, ma l'esecuzione di attività gravose di processore o I / O sul thread principale può causare la sospensione dell'interfaccia utente e la mancata risposta (documentazione ufficiale su ciò che accade).

Puoi rimediare mettendo queste attività più pesanti su un thread in background.

Un modo per farlo è usare AsyncTask , che fornisce un framework per facilitare l'uso di un thread in background e anche eseguire attività Thread UI prima, durante e dopo lo sfondo Thread ha completato il suo lavoro.

Metodi che possono essere sovrascritti durante l'estensione di `AsyncTask` :

- `onPreExecute()` : richiamato sul **thread UI** prima `onPreExecute()`
- `doInBackground()` : richiamato sul **thread in background** immediatamente dopo che `onPreExecute()` termina l'esecuzione.
- `onProgressUpdate()` : richiamato sul **thread dell'interfaccia utente** dopo una chiamata a `publishProgress(Progress...)` .
- `onPostExecute()` : richiamato sul **thread dell'interfaccia utente al** termine del calcolo dello sfondo

Esempio

```
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 semplicemente:

```
new MyCustomAsyncTask().execute(userSuppliedFilename);
```

Nota

Quando `AsyncTask` un `AsyncTask` possiamo passare tre tipi tra parentesi `< >` .
Definito come `<Params, Progress, Result>` (vedere la **sezione Parametri**)

Nell'esempio precedente abbiamo utilizzato i tipi `<File, Void, String>` :

```
AsyncTask<File, Void, String>
// Params has type File
// Progress has unused type
// Result has type String
```

`Void` viene utilizzato quando si desidera contrassegnare un tipo come non usato.

Si noti che non è possibile passare i tipi primitivi (cioè `int` , `float` e altri 6) come parametri. In questi casi, dovresti passare le loro classi wrapper , ad esempio `Integer` anziché `int` , o `Float` invece di `float` .

Il ciclo di vita AsyncTask e Activity

AsyncTasks non segue il ciclo di vita delle istanze di attività. Se si avvia un AsyncTask all'interno di un'attività e si ruota il dispositivo, l'attività verrà distrutta e verrà creata una nuova istanza. Ma AsyncTask non morirà. Continuerà a vivere fino al suo completamento.

Soluzione: AsyncTaskLoader

Una sottoclasse di Caricatori è AsyncTaskLoader. Questa classe svolge la stessa funzione dell'AsyncTask, ma molto meglio. Può gestire le modifiche alla configurazione delle attività più facilmente e si comporta nei cicli di vita di Fragments and Activities. La cosa bella è che AsyncTaskLoader può essere utilizzato in qualsiasi situazione in cui viene utilizzato AsyncTask. Ogni volta che i dati devono essere caricati in memoria per l'attività / frammento da gestire, AsyncTaskLoader può fare meglio il lavoro.

Annullare AsyncTask

```
YourAsyncTask task = new YourAsyncTask();
task.execute();
task.cancel();
```

Questo non ferma il tuo compito se fosse in corso, imposta semplicemente il flag cancellato che può essere controllato controllando il valore restituito da `isCancelled()` (assumendo che il tuo codice sia correntemente in esecuzione) facendo così:

```
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

Se un AsyncTask viene cancellato mentre `doInBackground(Params... params)` è ancora in esecuzione, allora il metodo `onPostExecute(Result result)` **NON** verrà chiamato dopo il `doInBackground(Params... params)` . AsyncTask richiamerà invece `onCancelled(Result result)` per indicare che l'attività è stata annullata durante l'esecuzione.

Pubblicazione dei progressi

A volte, abbiamo bisogno di aggiornare il progresso del calcolo fatto da un `AsyncTask` . Questo progresso potrebbe essere rappresentato da una stringa, un intero, ecc. Per fare ciò, dobbiamo usare due funzioni. Per prima cosa, dobbiamo impostare la funzione `onProgressUpdate` cui tipo di parametro è uguale al secondo parametro di tipo del nostro `AsyncTask` .

```
class YourAsyncTask extends AsyncTask<URL, Integer, Long> {
    @Override
    protected void onProgressUpdate(Integer... args) {
        setProgressPercent(args[0])
    }
}
```

In secondo luogo, dobbiamo usare la funzione `publishProgress` necessariamente sulla funzione `doInBackground` , e questo è tutto, il metodo precedente farà tutto il lavoro.

```
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;
 }
```

Scarica immagine usando AsyncTask in Android

Questo tutorial spiega come scaricare l'immagine usando AsyncTask in Android. L'esempio seguente scarica l'immagine mentre mostri la barra di avanzamento durante il download.

Comprendere Android AsyncTask

L'attività asincrona consente di implementare MultiThreading senza sporcare le mani nei thread. AsyncTask consente l'uso corretto e semplice del thread dell'interfaccia utente. Permette di eseguire operazioni in background e di passare i risultati sul thread dell'interfaccia utente. Se stai facendo qualcosa di isolato relativo all'interfaccia utente, ad esempio il download di dati da presentare in un elenco, andare avanti e utilizzare AsyncTask.

- AsyncTasks dovrebbe idealmente essere utilizzato per operazioni brevi (pochi secondi al massimo).
- Un'attività asincrona è definita da 3 tipi generici, denominati Params, Progress e Result e 4 passaggi, chiamati `onPreExecute()` , `doInBackground()` , `onProgressUpdate()` e `onPostExecute()` .
- In `onPreExecute()` è possibile definire il codice, che deve essere eseguito prima dell'inizio dell'elaborazione in background.
- doInBackground ha un codice che deve essere eseguito in background, qui in `doInBackground()` possiamo inviare i risultati più volte al thread degli eventi con il metodo publishProgress (), per notificare che l'elaborazione in background è stata completata possiamo restituire risultati semplicemente.
- `onProgressUpdate()` metodo `onProgressUpdate()` riceve gli aggiornamenti di avanzamento dal metodo `doInBackground()` , che viene pubblicato tramite il metodo `publishProgress()` e questo metodo può utilizzare questo aggiornamento di avanzamento per aggiornare il thread di eventi
- `onPostExecute()` metodo `onPostExecute()` gestisce i risultati restituiti dal metodo `doInBackground()` .
- I tipi generici usati sono
    - Params, il tipo di parametri inviati all'attività dopo l'esecuzione
    - Progresso, il tipo di unità di progresso pubblicate durante il calcolo dello sfondo.
    - Risultato, il tipo del risultato del calcolo dello sfondo.
- Se un'attività asincrona non utilizza alcun tipo, può essere contrassegnata come tipo Void.
- Un'attività asincrona in esecuzione può essere annullata chiamando il metodo `cancel(boolean)` .

Download di immagini tramite Android AsyncTask

il tuo layout .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>
```

.java class

```
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;
        }
    }
}
```

Dato che al momento non esiste un campo di commento per gli esempi (o non l'ho trovato o non ne ho il permesso) ecco alcuni commenti a riguardo:

Questo è un buon esempio di cosa si può fare con AsyncTask.

Tuttavia, l'esempio ha attualmente problemi con

- possibili perdite di memoria
- arresto anomalo dell'app se si verificava una rotazione dello schermo poco prima del completamento dell'attività asincrona.

Per dettagli vedi:

- Passa Attività come Debolezza di Risanamento per evitare perdite di memoria
- http://stackoverflow.com/documentation/android/117/asynctask/5377/possible-problems-with-inner-async-tasks
- Evitare le perdite di attività con AsyncTask

Passa Attività come Debolezza di Risanamento per evitare perdite di memoria

È normale che un AsyncTask richieda un riferimento all'attività che lo ha chiamato.

Se AsyncTask è una classe interna dell'attività, puoi fare riferimento direttamente a qualsiasi variabile / metodo membro.

Se, tuttavia, AsyncTask non è una classe interna dell'attività, sarà necessario passare un riferimento attività a AsyncTask. Quando si esegue questa operazione, un potenziale problema che potrebbe verificarsi è che AsyncTask manterrà il riferimento dell'attività fino a quando AsyncTask non avrà completato il proprio lavoro nel suo thread in background. Se l'attività è terminata o terminata prima che venga eseguito il lavoro sul thread in background di AsyncTask, AsyncTask avrà ancora il suo riferimento all'attività e, pertanto, non può essere sottoposto a Garbage Collection.

Di conseguenza, ciò causerà una perdita di memoria.

Per evitare che ciò accada, fai uso di un WeakReference in AsyncTask invece di avere un riferimento diretto all'Attività.

Ecco un esempio AsyncTask che utilizza un 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();
        }
    }
}
```

Chiamare AsyncTask da un'attività:

```
new MyAsyncTask(this).execute("param1", "param2");
```

Chiamando il AsyncTask da un frammento:

```
new MyAsyncTask(getActivity()).execute("param1", "param2");
```

Ordine di esecuzione

Quando sono stati introdotti per la prima volta, `AsyncTasks` stato eseguito in serie su un singolo thread in background. A partire da `DONUT` , questo è stato modificato in un pool di thread che consente a più attività di operare in parallelo. A partire da `HONEYCOMB` , le attività vengono eseguite su un singolo thread per evitare errori di applicazione comuni causati dall'esecuzione parallela.

Se si desidera realmente l'esecuzione parallela, è possibile richiamare `executeOnExecutor(java.util.concurrent.Executor, Object[])` con `THREAD_POOL_EXECUTOR` .

> SERIAL_EXECUTOR -> Un Executor che esegue le attività una alla volta in ordine seriale.
>
> THREAD_POOL_EXECUTOR -> Un Executor che può essere utilizzato per eseguire attività in parallelo.

campione :

```
Task task = new Task();
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB)
    task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR, data);
else
    task.execute(data);
```

AsyncTask: esecuzione seriale ed esecuzione parallela dell'attività

AsyncTask è una classe astratta e non eredita la classe `Thread` . Ha un metodo **astratto** `doInBackground(Params... params)` , che viene sovrascritto per eseguire l'operazione. Questo metodo è chiamato da `AsyncTask.call()` .

Executor fa parte del pacchetto `java.util.concurrent` .

Inoltre, AsyncTask contiene 2 `Executor`

THREAD_POOL_EXECUTOR

Utilizza i thread di lavoro per eseguire le attività parallelamente.

```
public static final Executor THREAD_POOL_EXECUTOR = new ThreadPoolExecutor(CORE_POOL_SIZE,
MAXIMUM_POOL_SIZE, KEEP_ALIVE, TimeUnit.SECONDS, sPoolWorkQueue, sThreadFactory);
```

SERIAL_EXECUTOR

Esegue il compito in serie, cioè uno per uno.

```
private static class SerialExecutor implements Executor { }
```

Entrambi gli `Executor` sono **statici** , quindi `THREAD_POOL_EXECUTOR` solo un `THREAD_POOL_EXECUTOR` e un oggetto `SerialExecutor` , ma è possibile creare diversi oggetti `AsyncTask` .

Pertanto, se si tenta di eseguire più attività in background con l'Executor predefinito (`SerialExecutor`), queste attività saranno `SerialExecutor` ed eseguite in serie.

Se si tenta di eseguire più attività in background con `THREAD_POOL_EXECUTOR` , verranno eseguite parallelamente.

Esempio:

```
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");
        }
    }
}
```

Esegui Fare clic sul pulsante più volte per avviare un'attività e vedere il risultato.

Attività eseguita nel pool di thread (1)

Ogni operazione richiede 1000 ms per essere completata.

A t = 36 s, le attività 2, 3 e 4 sono messe in coda e hanno iniziato l'esecuzione anche perché sono in esecuzione parallela.

```
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
----------
```

Commento `Task Executed in thread pool` `Task executed Serially` `Task Executed in thread pool` (1) e uncomment `Task executed Serially` (2).

Esegui Fare clic sul pulsante più volte per avviare un'attività e vedere il risultato.

Sta eseguendo l'attività in modo seriale quindi ogni attività viene avviata dopo che l'attività corrente ha completato l'esecuzione. Di conseguenza, quando l'esecuzione del task 1 viene completata, solo l'attività 2 inizia a essere eseguita in background. Vice versa.

```
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
```

Leggi AsyncTask online: https://riptutorial.com/it/android/topic/117/asynctask

Capitolo 26: Attività

introduzione

Un'attività rappresenta una singola schermata con un'interfaccia utente **(UI)** . Un'app per Android può avere più di un'attività, ad esempio, un'app di posta elettronica può avere un'attività per elencare tutte le e-mail, un'altra attività per mostrare i contenuti della posta elettronica, un'altra attività per comporre una nuova e-mail. Tutte le attività in un'app interagiscono per creare un'esperienza utente perfetta.

Sintassi

- void onCreate (Bundle savedInstanceState) // Chiamato all'avvio dell'attività.
- void onPostCreate (Bundle savedInstanceState) // Chiamato quando l'avvio dell'attività è completo (dopo che onStart () e onRestoreInstanceState (Bundle) sono stati chiamati).
- void onStart () // Chiamato dopo onCreate (Bundle) - o after onRestart () quando l'attività è stata interrotta, ma ora viene nuovamente visualizzata all'utente.
- void onResume () // Chiamato dopo onRestoreInstanceState (Bundle), onRestart () o onPause (), affinché l'attività inizi a interagire con l'utente.
- void onPostResume () // Chiamato quando l'attività riprende è completa (dopo che onResume () è stato chiamato).
- void onRestart () // Chiamato dopo onStop () quando l'attività corrente viene nuovamente visualizzata all'utente (l'utente è tornato indietro).
- void onPause () // Chiamato come parte del ciclo di vita dell'attività quando un'attività sta andando in background, ma non è (ancora) stata uccisa.
- void onStop () // Chiamato quando non sei più visibile all'utente.
- void onDestroy () // Esegue qualsiasi pulizia finale prima che un'attività venga distrutta.
- void onNewIntent (Intent intent) // Questo è chiamato per le attività che impostano launchMode su "singleTop" nel loro pacchetto, o se un client usa il flag FLAG_ACTIVITY_SINGLE_TOP quando chiama startActivity (Intent).
- void onSaveInstanceState (Bundle outState) // Chiamato per recuperare lo stato per-instance da un'attività prima di essere ucciso in modo che lo stato possa essere ripristinato in onCreate (Bundle) o onRestoreInstanceState (Bundle) (il pacchetto popolato da questo metodo verrà passato ad entrambi).
- void onRestoreInstanceState (Bundle savedInstanceState) // Questo metodo viene chiamato dopo onStart () quando l'attività viene reinizializzata da uno stato precedentemente salvato,

indicato qui in savedInstanceState.

Parametri

Parametro	Dettagli
Intento	Può essere utilizzato con startActivity per avviare un'attività
impacchettare	Un mapping da chiavi String a vari valori Parcelable .
Contesto	Interfaccia con informazioni globali su un ambiente applicativo.

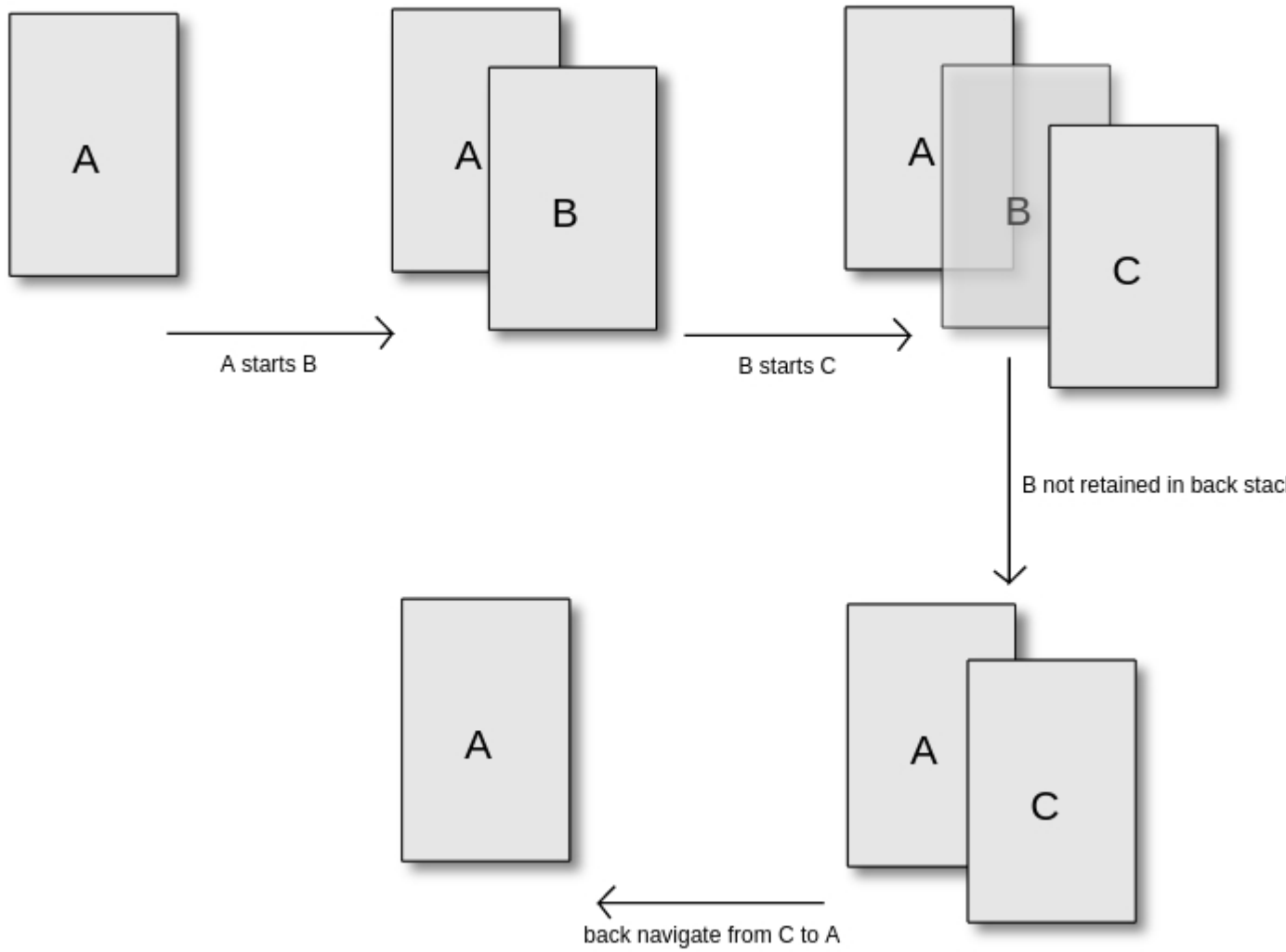
Osservazioni

Un'attività è un componente software che fornisce una schermata con cui gli utenti possono interagire al fine di fare qualcosa, come ad esempio comporre il telefono, scattare una foto, inviare una e-mail o visualizzare una mappa. Ogni attività viene data una finestra in cui disegnare la sua interfaccia utente. La finestra in genere riempie lo schermo, ma potrebbe essere più piccola dello schermo e galleggiare sopra altre finestre.

Examples

Escludere un'attività dalla cronologia dello stack

Lascia che ci sia l'attività `B` che può essere aperta e può avviare ulteriori attività. Ma l'utente non dovrebbe incontrarlo durante la navigazione nelle attività dell'attività.



La soluzione più semplice è impostare l'attributo `noHistory` su `true` per quel tag `<activity>` in `AndroidManifest.xml` :

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

Questo stesso comportamento è anche possibile dal codice se `B` chiama `finish()` prima di iniziare la prossima attività:

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

L'uso tipico del flag `noHistory` è con "Splash Screen" o attività di accesso.

Spiegazione del LifeCycle dell'attività Android

Assumi un'applicazione con MainActivity che può chiamare l'attività successiva utilizzando un clic

del pulsante.

```
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);
    } }
```

e

```
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");
    } }
```

Quando viene creata per la prima volta l'app
D / MainActivity: call onCreate da MainActivity
D / MainActivity: chiamando onStart da MainActivity
D / MainActivity: chiamata su Riduci da MainActivity
sono chiamati

Quando lo schermo dorme
08: 11: 03.142 D / MainActivity: chiamata onPause da MainActivity
08: 11: 03.192 D / MainActivity: chiamata onStop da MainActivity
sono chiamati. E ancora quando si sveglia
08: 11: 55.922 D / MainActivity: chiamata onRestart da MainActivity
08: 11: 55.962 D / MainActivity: chiamata onStart da MainActivity
08: 11: 55.962 D / MainActivity: chiamata su Riduci da MainActivity
sono chiamati

Caso 1: quando viene chiamata l'attività successiva dall'attività principale
D / MainActivity: chiamata Attività successiva
D / MainActivity: chiamata onPause da MainActivity

D / NextActivity: chiamando onCreate dalla prossima attività
D / NextActivity: chiamata onStart da Attività successiva
D / NextActivity: chiamata su Riduci dall'attività successiva
D / MainActivity: chiamata onStop da MainActivity

Quando si ritorna all'attività principale dalla prossima attività usando il pulsante Indietro
D / NextActivity: chiamata onPause da attività successiva
D / MainActivity: chiamando onRestart da MainActivity
D / MainActivity: chiamando onStart da MainActivity
D / MainActivity: chiamata su Riduci da MainActivity
D / NextActivity: chiamata onStop dall'attività successiva
D / NextActivity: chiama onDestroy dall'attività successiva

Caso 2: Quando l'attività è parzialmente oscurata (quando viene premuto il pulsante di panoramica) o Quando l'app passa allo sfondo e un'altra app la oscura completamente
D / MainActivity: chiamata onPause da MainActivity
D / MainActivity: chiamata onStop da MainActivity
e quando l'app è tornata in primo piano pronta ad accettare input utente,
D / MainActivity: chiamando onRestart da MainActivity
D / MainActivity: chiamando onStart da MainActivity
D / MainActivity: chiamata su Riduci da MainActivity
sono chiamati

Caso 3: quando viene chiamata un'attività per soddisfare l'intenzione implicita e l'utente ha effettuato una selezione. Ad esempio, quando viene premuto il pulsante di condivisione e l'utente deve selezionare un'app dall'elenco di applicazioni visualizzate
D / MainActivity: chiamata onPause da MainActivity

L'attività è visibile ma non attiva ora. Quando la selezione è completata e l'app è attiva
D / MainActivity: chiamata su Riduci da MainActivity
è chiamato

case4:
Quando l'app viene uccisa in background (per liberare risorse per un'altra app in primo piano), *onPause* (per dispositivo pre-honeycomb) o *onStop* (per poiché dispositivo a nido d'ape) sarà l'ultimo ad essere chiamato prima che l'app venga terminata.

onCreate e onDestroy verranno chiamati al massimo una volta ogni volta che viene eseguita l'applicazione. Ma onPause, onStop, onRestart, onStart, onResume possono essere chiamati più volte durante il ciclo di vita.

Activity launchMode

La modalità di avvio definisce il comportamento dell'attività nuova o esistente nell'attività.
Ci sono possibili modalità di lancio:

- standard
- singleTop

- singleTask
- singola istanza

Dovrebbe essere definito in Android manifest in `<activity/>` element come `android:launchMode` attributo `android:launchMode` .

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

Standard:

Valore predefinito. Se questa modalità è impostata, verrà sempre creata una nuova attività per ogni nuovo intento. Quindi è possibile ottenere molte attività dello stesso tipo. La nuova attività verrà posizionata nella parte superiore dell'attività. C'è qualche differenza per le diverse versioni di Android: se l'attività parte da un'altra applicazione, su androidi <= 4.4 verrà messa sullo stesso compito dell'applicazione starter, ma su> = 5.0 verrà creata una nuova attività.

SingleTop:

Questa modalità è quasi la stessa dello `standard` . Potrebbero essere create molte istanze di attività singleTop. La differenza è che se un'istanza di attività esiste già nella parte superiore dello stack corrente, verrà chiamato `onNewIntent()` anziché creare una nuova istanza.

SingleTask:

L'attività con questa modalità di avvio può avere solo un'istanza **nel sistema** . Verrà creata una nuova attività per attività, se non esiste. In caso contrario, l'attività con l'attività verrà spostata in primo piano e verrà chiamato `onNewIntent` .

Singola istanza:

Questa modalità è simile a `singleTask` . La differenza è un'attività che contiene un'attività con `singleInstance` potrebbe avere solo questa attività e nient'altro. Quando `singleInstance` attività `singleInstance` crea un'altra attività, verrà creata una nuova attività per posizionarla.

Presentazione dell'interfaccia utente con setContentView

La classe di attività si occupa di creare una finestra per te in cui puoi posizionare l'interfaccia utente con `setContentView` .
Esistono tre metodi `setContentView` :

- `setContentView(int layoutResID)` - Imposta il contenuto dell'attività da una risorsa di layout.
- `setContentView(View view)` - Imposta il contenuto dell'attività su una vista esplicita.
- `setContentView(View view, ViewGroup.LayoutParams params)` - Imposta il contenuto dell'attività su una vista esplicita con i parametri forniti.

Quando viene chiamato `setContentView` , questa vista viene posizionata direttamente nella gerarchia della vista dell'attività. Può essere essa stessa una complessa gerarchia delle visualizzazioni.

Esempi

Imposta il contenuto dal file di risorse:

Aggiungi file di risorse (main.xml in questo esempio) con la gerarchia di visualizzazione:

```
<?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>
```

Impostalo come contenuto nell'attività:

```
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);
    }
}
```

Imposta il contenuto su una vista esplicita:

```
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);
    }
}
```

Cancella il tuo attuale stack di attività e avvia una nuova attività

Se vuoi cancellare il tuo attuale stack di attività e avviare una nuova attività (ad esempio, disconnettendo l'app e avviando un log in Activity), sembrano esserci due approcci.

1. Target (API> = 16)

Chiamando `finishAffinity()` da un'attività

2. Target (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();
```

Termina l'applicazione con l'esclusione da Recenti

Definire innanzitutto un'AbilitàAttività in AndroidManifest.xml

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

Successivamente la classe 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);
    }
}
```

Navigazione verso l'alto per le attività

La navigazione verso l'alto viene eseguita in Android aggiungendo `android:parentActivityName=""` in Manifest.xml al tag attività. Fondamentalmente con questo tag si indica al sistema l'attività principale di un'attività.

Com'è fatto?

```
<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>
```

Ora quando cliccherò sulla freccia all'interno della barra degli strumenti di HomeActivity tornerò all'attività principale.

Codice Java

Qui scriverò il codice java appropriato richiesto per questa funzionalità.

```
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.
}
}
```

Se esegui questo codice, vedrai quando premi il pulsante indietro per tornare a MainActivity. Per una migliore comprensione di Up Navigation consiglio di leggere i documenti

Puoi personalizzare maggiormente questo comportamento in base alle tue esigenze ignorando

```
@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 semplice

Questa è una semplice modifica che viene principalmente utilizzata per navigare nell'attività dei genitori se il genitore si trova nel backstack. Chiamando `onBackPressed()` se id è uguale 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);
    }
```

Leggi Attività online: https://riptutorial.com/it/android/topic/1481/attivita

Capitolo 27: AudioManager

Examples

Richiesta di messa a fuoco audio transitoria

```
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);
        }
    }
}
```

Richiesta di messa a fuoco audio

```
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.
        }
    }
}
```

Leggi AudioManager online: https://riptutorial.com/it/android/topic/6798/audiomanager

Capitolo 28: Autenticatore Android

Examples

Servizio di autenticazione account di base

Il sistema Autenticatore account Android può essere utilizzato per rendere il client autenticato con un server remoto. Sono richieste tre informazioni:

- Un servizio, attivato da `android.accounts.AccountAuthenticator` . Il suo metodo `onBind` dovrebbe restituire una sottoclasse di `AbstractAccountAuthenticator` .
- Un'attività per richiedere all'utente le credenziali (attività di accesso)
- Un file di risorse xml per descrivere l'account

1. Il servizio:

Inserisci le seguenti autorizzazioni nel tuo 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" />
```

Dichiarare il servizio nel file manifest:

```
<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>
```

Tieni presente che `android.accounts.AccountAuthenticator` è incluso nel tag `intent-filter` . La risorsa xml (denominata `authenticator` qui) è specificata nel tag `meta-data` .

La classe di servizio:

```
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. La risorsa 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" />
```

Non assegnare direttamente una stringa ad `android:label` o assegna i drawable mancanti. Crollerà senza preavviso.

3. Estendi la classe 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;
    }
}
```

Il metodo `addAccount()` nella classe `AbstractAccountAuthenticator` è importante poiché questo metodo viene chiamato quando si aggiunge un account dalla schermata "Aggiungi account" in Impostazioni. `AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE` è importante, poiché includerà l'oggetto AccountAuthenticatorResponse necessario per restituire le chiavi dell'account in seguito alla verifica dell'utente.

Leggi Autenticatore Android online: https://riptutorial.com/it/android/topic/6759/autenticatore-android

Capitolo 29: AutoCompleteTextView

Osservazioni

Se desideri offrire suggerimenti all'utente quando digita in un campo di testo modificabile, puoi utilizzare `AutoCompleteTextView` . Fornisce suggerimenti automaticamente quando l'utente sta digitando. L'elenco di suggerimenti viene visualizzato in un menu a discesa dal quale l'utente può selezionarne uno per sostituire il contenuto della casella di modifica.

Examples

AutoCompleteTextView semplice e codificato

Design (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" />
```

Trova la vista nel codice dopo `setContentView()` (o il suo frammento o equivalente vista personalizzata):

```
final AutoCompleteTextView myAutoCompleteTextView =
    (AutoCompleteTextView) findViewById(R.id.autoCompleteTextView1);
```

Fornire dati hard-coded tramite un adattatore:

```
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);
```

Suggerimento: sebbene il modo migliore sarebbe quello di fornire dati tramite un `Loader` di qualche tipo invece di un elenco codificato come questo.

Completamento automatico con CustomAdapter, ClickListener e Filter

Layout principale: 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>
```

Riga layout 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;
    }
}
```

Classe del modello: 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;
    }
```

```
}
```

Classe adattatore: 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();
                }
            }
        }
    };
}
```

Leggi AutoCompleteTextView online:
https://riptutorial.com/it/android/topic/5300/autocompletetextview

Capitolo 30: Autorizzazioni di runtime in API-23 +

introduzione

Android Marshmallow ha introdotto il modello di autorizzazione di runtime . Le autorizzazioni sono suddivise in due categorie, ovvero Autorizzazioni normali e pericolose . dove le autorizzazioni pericolose sono ora concesse dall'utente in fase di esecuzione.

Osservazioni

Da sdk 23 Android richiede autorizzazioni di runtime per le autorizzazioni sui dispositivi con Android 6.0 e versioni successive, all'interno di quelli classificati come gruppi di permessi pericolosi. I gruppi di permessi pericolosi sono considerati compromessi dalla privacy e / o dalla sicurezza dell'utente.

Di seguito è riportato un elenco di gruppi di autorizzazioni pericolose:

Gruppi di permessi pericolosi

Gruppo di Permessi
CALENDARIO
TELECAMERA
CONTATTI
POSIZIONE
MICROFONO
TELEFONO
SENSORI
sms
CONSERVAZIONE

Qualsiasi autorizzazione da questi gruppi richiede la gestione delle autorizzazioni di runtime per i dispositivi su Android 6.0 e versioni successive con un sdk di destinazione di 23 o versione successiva.

Permessi normali

Di seguito è riportato un elenco di autorizzazioni normali. Questi non sono considerati pericolosi per la privacy o la sicurezza dell'utente e quindi non richiedono permessi di runtime per sdk 23 e successivi.

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
IMPOSTA SVEGLIA
SET_TIME_ZONE
IMPOSTA SFONDO
SET_WALLPAPER_HINTS
TRANSMIT_IR
UNINSTALL_SHORTCUT
USE_FINGERPRINT
VIBRARE
WAKE_LOCK
WRITE_SYNC_SETTINGS

Examples

Autorizzazioni multiple di Android 6.0

Questo esempio mostra come verificare le autorizzazioni in fase di esecuzione in Android 6 e versioni successive.

```
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;
        }
    }
}
```

Applicazione delle autorizzazioni nelle trasmissioni, URI

È possibile eseguire un controllo delle autorizzazioni quando si invia un'intenzione a un destinatario di una trasmissione registrata. Le autorizzazioni inviate vengono confrontate con quelle registrate sotto il tag. Limitano chi può inviare trasmissioni al ricevitore associato.

Per inviare una richiesta di trasmissione con autorizzazioni, specifica l'autorizzazione come stringa nella chiamata `Context.sendBroadcast(Intent intent, String permission)` , ma tieni presente che l'app del ricevente **DEVE** avere tale autorizzazione per ricevere la tua trasmissione. Il ricevitore dovrebbe essere installato prima del mittente.

La firma del metodo è:

```
 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");
```

ed è possibile specificare nel manifest che il mittente broadcast è necessario per includere l'autorizzazione richiesta inviata attraverso sendBroadcast:

```
<!--  Your special permission -->
<permission android:name="org.quadcore.mypermission"
   android:label="my_permission"
   android:protectionLevel="dangerous"></permission>
```

Dichiara inoltre il permesso nel manifest dell'applicazione che si suppone di ricevere questa trasmissione:

```
<!--  I use the permission ! -->
<uses-permission android:name="org.quadcore.mypermission"/>
<!-- along with the receiver -->
<receiver android:name="Bcastreceiver" android:exported="true" />
```

Nota: sia un ricevitore che un broadcaster possono richiedere un'autorizzazione e, quando ciò accade, entrambi i controlli delle autorizzazioni devono passare affinché l'intent venga consegnato alla destinazione associata. L'app che definisce l'autorizzazione deve essere installata per prima.

Trova la documentazione completa qui su Autorizzazioni.

Più autorizzazioni di runtime dagli stessi gruppi di autorizzazioni

Nel manifest abbiamo quattro pericolose autorizzazioni di runtime da due gruppi.

```
<!-- 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"/>
```

Nell'attività in cui sono richieste le autorizzazioni. Nota è importante verificare le autorizzazioni in qualsiasi attività che richiede autorizzazioni, poiché le autorizzazioni possono essere revocate mentre l'app è in background e l'app si arresta in modo anomalo.

```
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();
    }
```

Abbiamo solo bisogno di chiedere il permesso per uno di questi da ciascun gruppo e tutte le altre autorizzazioni da questo gruppo sono concesse a meno che l'autorizzazione non venga revocata dall'utente.

```
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;
}
```

Questo riguarda il risultato dell'utente che consente o non autorizza le autorizzazioni. In questo esempio, se le autorizzazioni non sono consentite, l'app viene uccisa.

```
@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);
    }
}
```

Ulteriori informazioni https://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en

Utilizzando PermissionUtil

PermissionUtil è un modo semplice e conveniente di chiedere permessi nel contesto. È possibile fornire facilmente ciò che dovrebbe accadere in caso di tutte le autorizzazioni richieste concesse (`onAllGranted()`), qualsiasi richiesta è stata negata (`onAnyDenied()`) o nel caso in cui sia necessario un razionale (`onRational()`).

Ovunque nel tuo AppCompatActivity o Framment che vuoi chiedere la permisssion dell'utente

```
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);
```

E aggiungilo a `onRequestPermissionsResult`

```
if(mRequestObject!=null){
    mRequestObject.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

Aggiungi anche l'autorizzazione richiesta al tuo AndroidManifest.xml

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

Includere tutti i codici relativi alle autorizzazioni a una classe di base astratta ed estendere l'attività di questa classe base per ottenere un codice più pulito / riutilizzabile

```
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);
    }
}
```

Esempio di utilizzo nell'attività

L'attività dovrebbe estendere la classe base astratta definita sopra come segue:

```
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.
        }
    });
}
```

Leggi Autorizzazioni di runtime in API-23 + online:
https://riptutorial.com/it/android/topic/1525/autorizzazioni-di-runtime-in-api-23-plus

Capitolo 31: Autosizing TextViews

introduzione

Un TextView che ridimensiona automaticamente il testo per adattarsi perfettamente ai suoi limiti.

Android O ti consente di istruire un TextView affinché le dimensioni del testo si espandano o si contraggano automaticamente per riempire il suo layout in base alle caratteristiche e ai limiti del TextView.

È possibile impostare l'autosizing TextView in entrambi i codici o XML.

Esistono due modi per impostare l'autosizing di TextView: **Granularity** e **dimensioni predefinite**

Examples

granularità

In Java:

Chiama il metodo `setAutoSizeTextTypeUniformWithConfiguration()` :

```
setAutoSizeTextTypeUniformWithConfiguration(int autoSizeMinTextSize, int autoSizeMaxTextSize,
int autoSizeStepGranularity, int unit)
```

In XML:

Utilizzare gli `autoSizeMinTextSize` , `autoSizeMaxTextSize` e `autoSizeStepGranularity` per impostare le dimensioni di ridimensionamento automatico nel file XML di layout:

```
<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" />
```

Guarda la demo di AutosizingTextViews su GitHub per maggiori dettagli.

Dimensioni predefinite

In Java:

Chiama il metodo `setAutoSizeTextTypeUniformWithPresetSizes()` :

```
setAutoSizeTextTypeUniformWithPresetSizes(int[] presetSizes, int unit)
```

In XML:

Utilizzare l'attributo `autoSizePresetSizes` nel file XML di layout:

```
<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" />
```

Per accedere alla matrice come risorsa, definire la matrice nel file *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>
```

Guarda la demo di AutosizingTextViews su GitHub per maggiori dettagli.

Leggi Autosizing TextViews online: https://riptutorial.com/it/android/topic/9652/autosizing-textviews

Capitolo 32: Avvertenze sui pelucchi

Osservazioni

Lo **strumento Lint** controlla i file di origine del progetto Android per potenziali bug e miglioramenti di ottimizzazione per correttezza, sicurezza, prestazioni, usabilità, accessibilità e internazionalizzazione. È possibile eseguire Lint dalla riga di comando o da Android Studio.

Documentazione ufficiale:

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

Examples

Utilizzo degli strumenti: ignora nei file xml

Gli `tools:ignore` attributo `tools:ignore` possono essere utilizzati nei file xml per eliminare gli avvertimenti sui pelucchi.

MA ignorare gli avvertimenti sui pelucchi con questa tecnica è la maggior parte delle volte il modo sbagliato di procedere.

Un avvertimento sui pelucchi deve essere compreso e corretto ... può essere ignorato se e solo se si ha una piena comprensione del suo significato e un motivo valido per ignorarlo.

Ecco un caso d'uso in cui è legittimo ignorare un avvertimento relativo ai pelucchi:

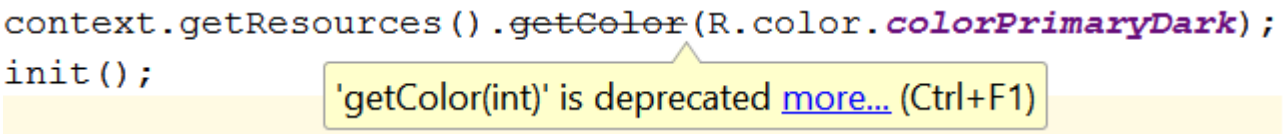
- Stai sviluppando un'applicazione di sistema (firmata con la chiave del produttore del dispositivo)
- La tua app deve modificare la data del dispositivo (o qualsiasi altra azione protetta)

Quindi puoi farlo nel tuo manifest: (ad esempio richiedendo l'autorizzazione protetta e ignorando l'avviso del filtro perché sai che nel tuo caso l'autorizzazione sarà concessa)

```
<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"/>
```

Importazione di risorse senza errore "Deprecated"

Utilizzando l'API di Android 23 o superiore, molto spesso tale situazione può essere vista:



Questa situazione è causata dal cambiamento strutturale dell'API di Android per quanto riguarda il recupero delle risorse. Ora la funzione:

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

dovrebbe essere usato. Ma la libreria android.support.v4 ha un'altra soluzione.

Aggiungi la seguente dipendenza al file build.gradle:

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

Quindi sono disponibili tutti i metodi dalla libreria di supporto:

```
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));
```

Inoltre è possibile utilizzare più metodi dalla libreria di supporto:

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

Configura LintOptions con gradle

Puoi configurare lint aggiungendo una sezione `lintOptions` nel file `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
    }
}
```

È possibile eseguire lanci per una variante specifica (vedere sotto), ad es. / `./gradlew lintRelease` , o per tutte le varianti (`./gradlew lint`), nel qual caso produce un report che descrive a quali varianti specifiche si applica un determinato problema.

Controlla qui il riferimento DSL per tutte le opzioni disponibili .

Come configurare il file lint.xml

È possibile specificare le preferenze di controllo Lint nel file `lint.xml` . Se stai creando questo file manualmente, inseriscilo nella directory principale del tuo progetto Android. Se stai configurando le preferenze di Lint in Android Studio, il file lint.xml viene automaticamente creato e aggiunto al tuo progetto Android per te.

Esempio:

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

Impostando il valore dell'attributo di severità nel tag, è possibile disabilitare il controllo dei rifiuti per un problema o modificare il livello di gravità per un problema.

L'esempio seguente mostra il contenuto di un file `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>
```

Configurazione del controllo del lint nei file di origine Java e XML

È possibile disabilitare il controllo dei pelucchi dai file di origine Java e XML.

Configurazione del controllo dei pelucchi in Java

Per disabilitare il controllo di Lint in modo specifico per una **classe** o un metodo **Java** nel tuo progetto Android, aggiungi l' **annotazione** `@SuppressLint` a quel codice Java.

Esempio:

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

Per disabilitare il controllo di tutti i problemi Lint:

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

Configurazione del controllo del lint in XML

È possibile utilizzare gli `tools:ignore` attributo per disabilitare il controllo dei rifiuti per sezioni specifiche dei **file XML** .

Per esempio:

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

Per sopprimere il controllo di tutti i problemi di Lint nell'elemento XML, utilizzare

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

Mark Sopprimere gli avvertimenti

È buona norma contrassegnare alcuni avvisi nel codice. Ad esempio, alcuni metodi deprecati sono necessari per il test o la vecchia versione di supporto. Ma il controllo dei lanci contrassegnerà quel codice con degli avvertimenti. Per evitare questo problema, è necessario utilizzare l'annotazione @SuppressWarnings.

Ad esempio, aggiungi ignorando gli avvisi ai metodi deprecati. È necessario inserire la descrizione degli avvertimenti nell'annotazione anche:

```
@SuppressWarnings("deprecated");
public void setAnotherColor (int newColor) {
```

```
    getApplicationContext().getResources().getColor(newColor)
}
```

Usando questa annotazione puoi ignorare tutti gli avvisi, inclusi Lint, Android e altri. L'uso di Ometti avvisi aiuta a capire il codice correttamente!

Leggi Avvertenze sui pelucchi online: https://riptutorial.com/it/android/topic/129/avvertenze-sui-pelucchi

Capitolo 33: Barra di avanzamento

Osservazioni

Documentazione ufficiale: ProgressBar

Examples

ProgressBar indeterminato

Un ProgressBar indeterminato mostra un'animazione ciclica senza indicazione del progresso.

ProgressBar indeterminato di base (rotella)

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

ProgressBar indeterminato orizzontale (barra piatta)

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Altri stili ProgressBar incorporati

```
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"
```

Utilizzare la ProgressBar indeterminata in un'attività

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setVisibility(View.VISIBLE);
progressBar.setVisibility(View.GONE);
```

Determina ProgressBar

Un determinato ProgressBar mostra il progresso corrente verso uno specifico valore massimo.

ProgressBar determinato orizzontale

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="10dp"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

ProgressBar determinato verticale

```
<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>
```

Anello determinato 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>
```

Utilizzare la ProgressBar determinata in un'attività.

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setSecondaryProgress(100);
progressBar.setProgress(10);
progressBar.setMax(100);
```

Barra di avanzamento personalizzata

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();

    }
}
```

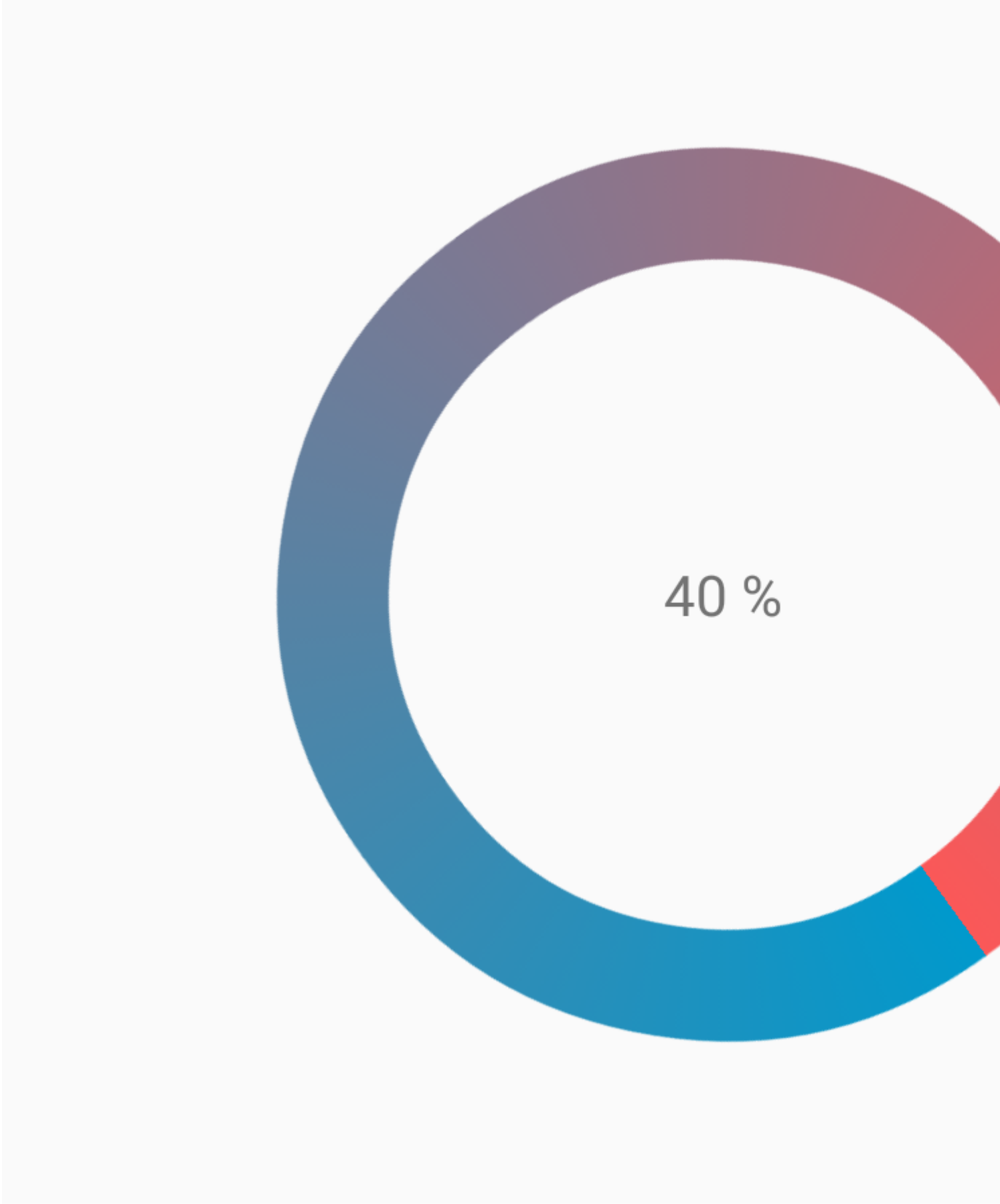
activity_custom_progressbar.xml :

```
<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>
```

Schermata di riferimento:



Tinting ProgressBar

Usando un tema AppCompat, il colore di `ProgressBar` sarà il `colorAccent` che hai definito.

5.0

Per cambiare il colore di `ProgressBar` senza modificare il colore dell'accento, puoi utilizzare l'attributo `android:theme` sostituisce il colore dell'accento:

```
<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>
```

Per colorare la `ProgressBar` puoi usare nel file xml gli attributi `android:indeterminateTintMode` e `android:indeterminateTint`

```
<ProgressBar
    android:indeterminateTintMode="src_in"
    android:indeterminateTint="@color/my_color"
/>
```

Materiale lineare ProgressBar

Secondo la documentazione del materiale :

> Un indicatore di progresso lineare dovrebbe sempre riempire da 0% a 100% e non diminuire mai di valore.
> Dovrebbe essere rappresentato da barre sul bordo di un'intestazione o di un foglio che appaiono e scompaiono.

Per utilizzare un materiale ProgressBar lineare, basta usare nel tuo xml:

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Indeterminato

Per creare una ProgressBar **indeterminata,** imposta l'attributo `android:indeterminate` a `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"/>
```

Determinato

Per creare **determinati** ProgressBar, imposta l'attributo `android:indeterminate` a `false` e usa gli attributi `android:max` e `android:progress` :

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"/>
```

Basta usare questo codice per aggiornare il valore:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
```

Buffer

Per creare un effetto **buffer** con ProgressBar, impostare l'attributo `android:indeterminate` su `false` e utilizzare `android:max` , `android:progress` e `android:secondaryProgress` attributi di `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"/>
```

Il valore del buffer è definito da `android:secondaryProgress` attributo `android:secondaryProgress` .
Basta usare questo codice per aggiornare i valori:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
progressBar.setSecondaryProgress(50);
```

Indeterminato e Determinare

Per ottenere questo tipo di ProgressBar basta usare una ProgressBar indeterminata usando l'attributo `android:indeterminate` a true.

```
<ProgressBar
    android:id="@+id/progressBar"
```

```
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="true"/>
```

Quindi, quando è necessario passare da un progresso indeterminato a uno determinato, utilizzare il metodo `setIndeterminate()` .

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setIndeterminate(false);
```

Creazione della finestra di dialogo Avanzamento personalizzato

Creando la classe di dialogo Progresso personalizzato, la finestra di dialogo può essere utilizzata per mostrare nell'istanza dell'interfaccia utente, senza ricreare la finestra di dialogo.

Prima crea una classe di dialogo di avanzamento personalizzata.

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;
       }
   }
}
```

Ora crea il layout di avanzamento personalizzato

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>
```

Questo è. Ora per chiamare la finestra di dialogo nel codice

```
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();
```

Leggi Barra di avanzamento online: https://riptutorial.com/it/android/topic/3353/barra-di-avanzamento

Capitolo 34: Bitmap Cache

introduzione

Memoria di caching bitmap efficiente: questo è particolarmente importante se l'applicazione utilizza le animazioni in quanto verranno interrotte durante la pulizia di GC e renderà l'applicazione pigra all'utente. Una cache consente di riutilizzare oggetti che sono costosi da creare. Se si carica l'oggetto in memoria, si può pensare a questo come una cache per l'oggetto. Il lavoro con bitmap in Android è complicato. È più importante memorizzare nella cache il bimap se lo si utilizzerà ripetutamente.

Sintassi

- `LruCache<String, Bitmap> mMemoryCache;//declaration of LruCache object.`
- void addBitmapToMemoryCache (String key, Bitmap bitmap) {} // dichiarazione del metodo generico che aggiunge bitmap nella memoria cache
- Bitmap getBitmapFromMemCache (Chiave stringa) {} // dichiarazione del metodo generico per ottenere bimap dalla cache.

Parametri

Parametro	Dettagli
chiave	chiave per memorizzare bitmap nella memoria cache
bitmap	valore bitmap che verrà memorizzato nella cache

Examples

Cache bitmap con cache LRU

LRU Cache

Il seguente codice di esempio mostra una possibile implementazione della classe LruCache per la memorizzazione nella cache delle immagini.

```
private LruCache<String, Bitmap> mMemoryCache;
```

Qui il valore stringa è la chiave per il valore bitmap.

```
// 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;
    }
};
```

Per aggiungere bitmap alla memoria cache

```
public void addBitmapToMemoryCache(String key, Bitmap bitmap) {
if (getBitmapFromMemCache(key) == null) {
        mMemoryCache.put(key, bitmap);
    }
}
```

Per ottenere bitmap dalla memoria cache

```
public Bitmap getBitmapFromMemCache(String key) {
    return mMemoryCache.get(key);
}
```

Per caricare bitmap in imageview basta usare **getBitmapFromMemCache ("** Passkey **").**

Leggi Bitmap Cache online: https://riptutorial.com/it/android/topic/9901/bitmap-cache

Capitolo 35: Bluetooth e Bluetooth LE API

Osservazioni

Bluetooth Classic è disponibile da Android 2.0 (livello API 5) e versioni successive. Bluetooth LE è disponibile da Android 4.3 (livello API 18) e versioni successive.

Examples

permessi

Aggiungi questa autorizzazione al file manifest per utilizzare le funzionalità Bluetooth nella tua applicazione:

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

Se devi avviare il rilevamento dei dispositivi o modificare le impostazioni Bluetooth, devi anche aggiungere questa autorizzazione:

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

Per raggiungere il livello API Android 23 e versioni successive, è necessario l'accesso alla posizione:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<!-- OR -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

* Vedere anche l'argomento Autorizzazioni per ulteriori dettagli su come utilizzare le autorizzazioni in modo appropriato.

Controlla se il bluetooth è abilitato

```
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
        }
    }
}
```

Rendi il dispositivo rilevabile

```
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
        }
    }
}
```

Trova dispositivi bluetooth nelle vicinanze

Prima dichiarare un `BluetoothAdapter` .

```
BluetoothAdapter mBluetoothAdapter;
```

Ora crea un `BroadcastReceiver` per `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());
    }
  }
};
```

Registra il `BroadcastReceiver`

```
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);
```

Quindi inizia a scoprire i dispositivi bluetooth vicini chiamando `startDiscovery`

```
mBluetoothAdapter.startDiscovery();
```

Non dimenticare di `onDestroy` la registrazione di `BroadcastReceiver` all'interno di `onDestroy`

```
unregisterReceiver(mReceiver);
```

Connetti al dispositivo Bluetooth

Dopo aver ottenuto il dispositivo Bluetooth, puoi comunicare con esso. Questo tipo di comunicazione viene eseguita utilizzando i flussi input / output socket:

Questi sono i passaggi fondamentali per lo stabilimento di comunicazione Bluetooth:

1) Inizializza socket:

```
 private BluetoothSocket _socket;
 //...
 public InitializeSocket(BluetoothDevice device){
    try {
        _socket = device.createRfcommSocketToServiceRecord(<Your app UDID>);
    } catch (IOException e) {
        //Error
    }
  }
```

2) Connetti alla presa:

```
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) Ottenere socket Input / Output stream

```
private InputStream _inStream;
private OutputStream _outStream;
//....
try {
    _inStream = _socket.getInputStream();
    _outStream =  _socket.getOutputStream();
} catch (IOException e) {
   //Error
}
```

Flusso di input : utilizzato come canale di dati in entrata (riceve i dati dal dispositivo collegato)

Flusso di output : utilizzato come canale di dati in uscita (invia dati al dispositivo collegato)

Dopo aver terminato il terzo passaggio, possiamo ricevere e inviare dati tra entrambi i dispositivi utilizzando flussi precedentemente inizializzati:

1) Ricezione di dati (lettura dal flusso di input 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) Invio di dati (Scrittura sul flusso di output)

```
public void write(byte[] bytes) {
    try {
        _outStream.write(bytes);
    } catch (IOException e) {
        //Error
    }
}
```

- Naturalmente, le funzionalità di connessione, lettura e scrittura dovrebbero essere eseguite in un thread dedicato.
- Gli zoccoli e gli oggetti Stream devono essere

Trova dispositivi Bluetooth Low Energy nelle vicinanze

L'API BluetoothLE è stata introdotta nell'API 18. Tuttavia, la modalità di scansione dei dispositivi è cambiata nell'API 21. La ricerca dei dispositivi deve iniziare con la definizione dell'UUID del servizio che deve essere scansionato (sia UICID a 16 bit sia quelli proprietari). . Questo esempio illustra come rendere un modo indipendente da API di cercare i dispositivi BLE:

1. Crea un modello di 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. Definisci l'interfaccia di scansione Bluetooth:

```
public interface ScanningAdapter {

    void startScanning(String name, String[] uuids);
    void stopScanning();
    List<BTDevice> getFoundDeviceList();
}
```

3. Crea una classe di fabbrica di scansione:

```
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. Crea un'implementazione di fabbrica per ciascuna 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. Ottieni l'elenco dei dispositivi trovati chiamando:

```
scanningFactory.startScanning({uuidlist});

wait few seconds...

List<BTDevice> bluetoothDeviceList = scanningFactory.getFoundDeviceList();
```

Leggi Bluetooth e Bluetooth LE API online: https://riptutorial.com/it/android/topic/2462/bluetooth-e-bluetooth-le-api

Capitolo 36: Bluetooth Low Energy

introduzione

Questa documentazione è intesa come un miglioramento rispetto alla documentazione originale e si concentrerà sulle più recenti API Bluetooth LE introdotte in Android 5.0 (API 21). Verranno trattati sia i ruoli centrali che quelli periferici nonché le modalità di avvio delle operazioni di scansione e pubblicità.

Examples

Ricerca di dispositivi BLE

Per utilizzare le API Bluetooth sono necessarie le seguenti autorizzazioni:

```
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
```

Se stai prendendo di mira dispositivi con Android 6.0 (**livello API 23**) o superiore e desideri eseguire operazioni di scansione / pubblicità, devi disporre di un permesso di posizione:

```
android.permission.ACCESS_FINE_LOCATION
```

o

```
android.permission.ACCESS_COARSE_LOCATION
```

Nota. I dispositivi con Android 6.0 (livello API 23 o superiore) devono anche avere i servizi di localizzazione abilitati.

Per avviare le operazioni di scansione / pubblicità è necessario un oggetto BluetoothAdapter:

```
BluetoothManager bluetoothManager = (BluetoothManager)
context.getSystemService(Context.BLUETOOTH_SERVICE);
bluetoothAdapter = bluetoothManager.getAdapter();
```

Il `startScan (ScanCallback callback)` della classe BluetoothLeScanner è il modo più semplice per avviare un'operazione di scansione. Per ricevere i risultati è necessario un oggetto `ScanCallback` :

```
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());
       }
    });
```

Connessione a un server GATT

Una volta scoperto l'oggetto BluetoothDevice desiderato, è possibile connettersi ad esso utilizzando il suo metodo `connectGatt()` che accetta come parametri un oggetto Context, un booleano che indica se connettersi automaticamente al dispositivo BLE e un riferimento BluetoothGattCallback dove eventi di connessione e operazioni client i risultati saranno consegnati:

```
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        device.connectGatt(context, false, bluetoothGattCallback,
BluetoothDevice.TRANSPORT_AUTO);
    } else {
        device.connectGatt(context, false, bluetoothGattCallback);
    }
```

Sostituire `onConnectionStateChange` in BluetoothGattCallback per ricevere la connessione di eventi di disconnessione:

```
    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.");
        }
    }
   };
```

Scrivere e leggere dalle caratteristiche

Una volta connesso a un server Gatt, interagirai con esso scrivendo e leggendo dalle caratteristiche del server. Per fare questo, prima devi scoprire quali servizi sono disponibili su questo server e quali caratteristiche sono disponibili in ogni servizio:

```
 @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
                    }
                }
            }
        }
```

Un'operazione di scrittura di base va così:

```
characteristic.setValue(newValue);
characteristic.setWriteType(BluetoothGattCharacteristic.WRITE_TYPE_DEFAULT);
gatt.writeCharacteristic(characteristic);
```

Al termine del processo di scrittura, verrà chiamato il metodo `onCharacteristicWrite` del tuo `BluetoothGattCallback` :

```
    @Override
    public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
        super.onCharacteristicWrite(gatt, characteristic, status);
        Log.d(TAG, "Characteristic " + characteristic.getUuid() + " written);
    }
```

Un'operazione di scrittura di base va così:

```
gatt.readCharacteristic(characteristic);
```

Al termine del processo di scrittura, verrà chiamato il metodo `onCharacteristicRead` del tuo `BluetoothGattCallback` :

```
@Override
public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
  super.onCharacteristicRead(gatt, characteristic, status);
  byte[] value = characteristic.getValue();
  }
```

Sottoscrizione alle notifiche dal server Gatt

È possibile richiedere di essere avvisati dal server Gatt quando il valore di una caratteristica è stato modificato:

```
gatt.setCharacteristicNotification(characteristic, true);
BluetoothGattDescriptor descriptor = characteristic.getDescriptor(
    UUID.fromString("00002902-0000-1000-8000-00805f9b34fb");
descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
mBluetoothGatt.writeDescriptor(descriptor);
```

Tutte le notifiche dal server verranno ricevute nel metodo `onCharacteristicChanged` di

BluetoothGattCallback:

```
    @Override
    public void onCharacteristicChanged(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic) {
        super.onCharacteristicChanged(gatt, characteristic);
        byte[] newValue = characteristic.getValue();
}
```

Pubblicità di un dispositivo BLE

È possibile utilizzare Bluetooth LE Advertising per trasmettere pacchetti di dati a tutti i dispositivi vicini senza prima stabilire una connessione. Ricorda che esiste un limite rigoroso di 31 byte di dati pubblicitari. Pubblicizzare il tuo dispositivo è anche il primo passo per consentire ad altri utenti di connettersi a te.

Poiché non tutti i dispositivi supportano Bluetooth LE Advertising, il primo passo consiste nel verificare che il dispositivo disponga di tutti i requisiti necessari per supportarlo. Successivamente, puoi inizializzare un oggetto `BluetoothLeAdvertiser` e con esso puoi iniziare le operazioni pubblicitarie:

```
        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);
        }
```

Utilizzando un server Gatt

Affinché il dispositivo possa agire come periferica, per prima cosa è necessario aprire un `BluetoothGattServer` e popolarlo con almeno un `BluetoothGattService` e un `BluetoothGattCharacteristic` :

```
BluetoothGattServer server=bluetoothManager.openGattServer(context,
bluetoothGattServerCallback);

BluetoothGattService service = new BluetoothGattService(SERVICE_UUID,
BluetoothGattService.SERVICE_TYPE_PRIMARY);
```

Questo è un esempio di BluetoothGattCharacteristic con autorizzazioni complete di scrittura, lettura e notifica. In base alle tue esigenze, potresti voler perfezionare le autorizzazioni che concedi questa caratteristica:

```
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);
```

Il `BluetoothGattServerCallback` è responsabile della ricezione di tutti gli eventi relativi a `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);
                }
```

Ogni volta che si riceve una richiesta di scrittura / lettura su una caratteristica o un descrittore, è necessario inviare una risposta al fine di completare la richiesta con successo:

```
@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);
}
```

Leggi Bluetooth Low Energy online: https://riptutorial.com/it/android/topic/10020/bluetooth-low-energy

Capitolo 37: BottomNavigationView

introduzione

La Bottom Navigation View è stata per qualche tempo nelle linee guida sulla progettazione dei materiali, ma non è stato facile per noi implementarla nelle nostre app.

Alcune applicazioni hanno creato le proprie soluzioni, mentre altre si sono affidate a librerie open source di terze parti per portare a termine il lavoro.

Ora la libreria di supporto del design sta vedendo l'aggiunta di questa barra di navigazione in basso, facciamo un tuffo nel modo in cui possiamo usarlo!

Osservazioni

Rappresenta una barra di navigazione in basso standard per l'applicazione. Si tratta di un'implementazione della navigazione bottom design del materiale.

link:

- Javadoc ufficiale

Examples

Implementazione di base

Per aggiungere il `BottomNavigationView` attenersi alla seguente `BottomNavigationView` :

1. Aggiungi la `build.gradle` alla **dipendenza** :

```
compile 'com.android.support:design:25.1.0'
```

2. Aggiungi il `BottomNavigationView` nel **tuo layout** :

```
<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 il **menu** per popolare 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. Allegare **un listener** per gli eventi click:

```
//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
            }
        });
```

Acquista il codice demo su BottomNavigation-Demo

Personalizzazione di BottomNavigationView

Nota: presumo che tu sappia come utilizzare ***`BottomNavigationView`.***

In questo esempio spiegherò come aggiungere selector per **`BottomNavigationView`** . Quindi puoi indicare sull'interfaccia utente per icone e testi.

Crea disegnabile `bottom_navigation_view_selector.xml` come

```
<?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>
```

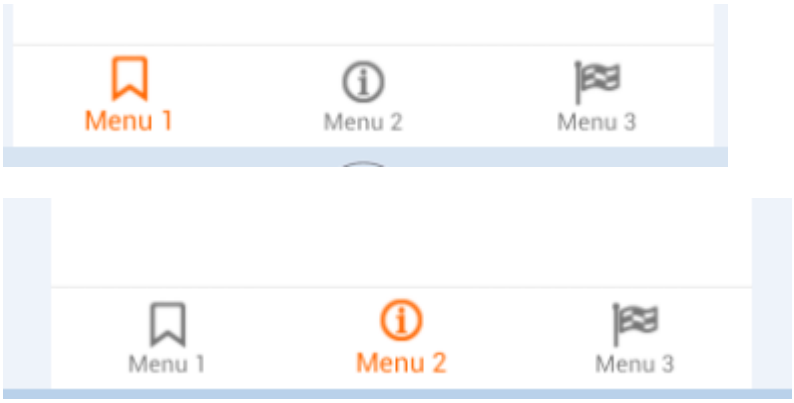
E usa gli attributi sotto in **`BottomNavigationView`** nel file di layout

```
app:itemIconTint="@drawable/bottom_navigation_view_selector"
```

```
app:itemTextColor="@drawable/bottom_navigation_view_selector"
```

> Nell'esempio precedente, ho utilizzato lo stesso selettore `bottom_navigation_view_selector` per `app:itemIconTint` e `app:itemTextColor` per mantenere i colori di testo e icone uguali. Ma se il tuo disegno ha colori diversi per il testo e l'icona, puoi definire 2 selettori diversi e usarli.

L'output sarà simile al seguente



Gestione degli stati abilitati / disabilitati

Crea un selettore per abilitare / disabilitare la voce di menu.

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" />
```

Consentendo più di 3 menu

Questo esempio è strettamente una soluzione poiché, attualmente non è possibile disattivare un comportamento noto come ShiftMode.

Crea una funzione in quanto tale.

```
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);
    }
}
```

Questo disabilita il comportamento Shifting del menu quando il numero di oggetti supera i 3 n.

USO

```
BottomNavigationView navView = (BottomNavigationView)
findViewById(R.id.bottom_navigation_bar);
disableMenuShiftMode(navView);
```

Problema di Proguard : aggiungi anche il seguente file di configurazione proguard, altrimenti non funzionerebbe.

```
-keepclassmembers class android.support.design.internal.BottomNavigationMenuView {
boolean mShiftingMode;
}
```

In alternativa, puoi creare una classe e accedere a questo metodo da lì. Vedere la risposta originale qui

NOTA : questo è un **HOTFIX** basato su **Reflection** , si prega di aggiornare questo una volta che la libreria di supporto di Google viene aggiornata con una chiamata di funzione diretta.

Leggi BottomNavigationView online:
https://riptutorial.com/it/android/topic/7565/bottomnavigationview

Capitolo 38: BroadcastReceiver

introduzione

BroadcastReceiver (ricevitore) è un componente Android che consente di registrarsi per eventi di sistema o applicazione. Tutti i ricevitori registrati per un evento vengono notificati dal runtime Android quando si verifica questo evento.

ad esempio, una trasmissione che annuncia che lo schermo è spento, la batteria è scarica o è stata scattata una foto.

Le applicazioni possono anche avviare le trasmissioni, ad esempio per consentire ad altre applicazioni di sapere che alcuni dati sono stati scaricati sul dispositivo ed è disponibile per essere utilizzati.

Examples

Introduzione al ricevitore Broadcast

Un ricevitore Broadcast è un componente Android che consente di registrarsi per eventi di sistema o applicazione.

Un ricevitore può essere registrato tramite il file `AndroidManifest.xml` o dinamicamente tramite il metodo `Context.registerReceiver()` .

```
public class MyReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      //Your implementation goes here.
   }
}
```

Qui ho preso un esempio di `ACTION_BOOT_COMPLETED` che viene `ACTION_BOOT_COMPLETED` dal sistema una volta che Android ha completato il processo di avvio.

Puoi registrare un ricevitore nel file manifest come questo:

```
<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>
```

Ora il dispositivo viene avviato, viene chiamato il metodo `onReceive()` e quindi è possibile eseguire il proprio lavoro (ad esempio avviare un servizio, avviare un allarme).

Informazioni di base su BroadcastReceiver

BroadcastReceivers viene utilizzato per ricevere gli eventi di trasmissione inviati dal sistema operativo Android, da altre app o dalla stessa app.

Ogni intent viene creato con un *filtro Intent* , che richiede *un'azione* String. Ulteriori informazioni possono essere configurate nell'intento.

Allo stesso modo, BroadcastReceivers si registra per ricevere Intenti con un particolare Filtro Intento. Possono essere registrati a livello di codice:

```
mContext.registerReceiver(new BroadcastReceiver() {
    @Override
   public void onReceive(Context context, Intent intent) {
      //Your implementation goes here.
   }
}, new IntentFilter("Some Action"));
```

o nel file `AndroidManifest.xml` :

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="Some Action"/>
    </intent-filter>
</receiver>
```

Per ricevere l'intenzione, imposta l'azione su qualcosa documentato dal sistema operativo Android, da un'altra app o API o all'interno della tua applicazione, utilizzando `sendBroadcast` :

```
mContext.sendBroadcast(new Intent("Some Action"));
```

Inoltre, l'intento può contenere informazioni, come stringhe, primitive e *Parcelables* , che possono essere visualizzate in `onReceive` .

Utilizzando LocalBroadcastManager

LocalBroadcastManager viene utilizzato per inviare Broadcast Intents all'interno di un'applicazione, senza esporli agli ascoltatori indesiderati.

L'uso di *LocalBroadcastManager* è più efficiente e più sicuro dell'uso di `context.sendBroadcast()` , poiché non è necessario preoccuparsi di eventuali trasmissioni di programmi falsificati da altre applicazioni, il che può costituire un rischio per la sicurezza.

Ecco un semplice esempio di invio e ricezione di trasmissioni locali:

```
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);
```

Ricevitore di trasmissione Bluetooth

aggiungi permesso nel file manifest

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

Nel tuo frammento (o attività)

- Aggiungi il metodo ricevitore

```
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;
    }
};
```

Registra trasmissione

- Chiama questo metodo su onResume ()

```
private void registerBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    manager.registerReceiver(mBluetoothStatusChangedReceiver, new
```

```
IntentFilter(Constants.BROADCAST_BLUETOOTH_STATE));
}
```

Annulla registrazione

- Chiama questo metodo su onPause ()

```
private void unregisterBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    // Beacon□□□
    manager.unregisterReceiver(mBluetoothStatusChangedReceiver);
}
```

Abilitazione e disabilitazione di un ricevitore di trasmissione a livello di programmazione

Per abilitare o disabilitare un `BroadcastReceiver` , abbiamo bisogno di ottenere un riferimento al `PackageManager` e abbiamo bisogno di un oggetto `ComponentName` contenente la classe del ricevitore che vogliamo abilitare / disabilitare:

```
ComponentName componentName = new ComponentName(context, MyBroadcastReceiver.class);
PackageManager packageManager = context.getPackageManager();
```

Ora possiamo chiamare il seguente metodo per abilitare `BroadcastReceiver` :

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP);
```

Oppure possiamo usare `COMPONENT_ENABLED_STATE_DISABLED` per disabilitare il ricevitore:

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
    PackageManager.DONT_KILL_APP);
```

BroadcastReceiver per gestire eventi BOOT_COMPLETED

L'esempio seguente mostra come creare un `BroadcastReceiver` grado di ricevere eventi `BOOT_COMPLETED` . In questo modo, si è in grado di avviare un `Service` o avviare `Activity` nel più breve dispositivo è stato acceso.

Inoltre, puoi utilizzare `BOOT_COMPLETED` eventi `BOOT_COMPLETED` per ripristinare gli allarmi poiché vengono distrutti quando il dispositivo viene spento.

NOTA: l'utente deve aver avviato l'applicazione almeno una volta prima di poter ricevere l'azione `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
            }
        }
    }
}
```

Esempio di un LocalBroadcastManager

Un BroadcastReceiver è fondamentalmente un meccanismo per inoltrare Intenti attraverso il sistema operativo per eseguire azioni specifiche. Un essere di definizione classica

> "Un ricevitore Broadcast è un componente Android che ti consente di registrarti per eventi di sistema o applicazioni."

LocalBroadcastManager è un modo per inviare o ricevere trasmissioni all'interno di un processo di applicazione. Questo meccanismo ha molti vantaggi

1. poiché i dati rimangono all'interno del processo di applicazione, i dati non possono essere trapelati.
2. I LocalBroadcast sono risolti più velocemente, poiché la risoluzione di una normale trasmissione avviene in fase di esecuzione su tutto il sistema operativo.

Un semplice esempio di LocalBroastManager è:

SenderActivity

```
  Intent intent = new Intent("anEvent");
  intent.putExtra("key", "This is an event");
  LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
```

ReceiverActivity

1. Registra un ricevitore

```
LocalBroadcastManager.getInstance(this).registerReceiver(aLBReceiver,
              new IntentFilter("anEvent"));
```

2. Un oggetto concreto per eseguire azioni quando viene chiamato il ricevitore

```
private BroadcastReceiver aLBReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        // perform action here.
    }
};
```

3. annullare la registrazione quando la vista non è più visibile.

```
@Override
protected void onPause() {
    // Unregister since the activity is about to be closed.
    LocalBroadcastManager.getInstance(this).unregisterReceiver(aLBReceiver);
    super.onDestroy();
}
```

Comunicare due attività tramite ricevitore Broadcast personalizzato

È possibile comunicare due attività in modo che l'attività A possa essere notificata di un evento che si verifica nell'attività B.

Attività 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
    }
};
```

Attività B

```
final String eventName = "your.package.goes.here.EVENT";

private void launchEvent() {
    Intent eventIntent = new Intent(eventName);
    this.sendBroadcast(eventIntent);
}
```

Naturalmente è possibile aggiungere ulteriori informazioni alla trasmissione aggiungendo degli extra all'Intento che viene passato tra le attività. Non aggiunto per mantenere l'esempio il più semplice possibile.

Trasmissione appiccicosa

Se stiamo usando il metodo sendStickyBroadcast (intento), l'intento corrispondente è appiccicoso, il che significa che l'intento che stai inviando rimane in attesa dopo che la trasmissione è completa. Un StickyBroadcast come suggerisce il nome è un meccanismo per leggere i dati da una trasmissione, una volta completata la trasmissione. Questo può essere usato in uno scenario in cui si potrebbe voler dire di dire in un `Activity's onCreate()` il valore di una chiave nell'intento prima che quell'attività fosse lanciata.

```
Intent intent = new Intent("com.org.action");
intent.putExtra("anIntegerKey", 0);
sendStickyBroadcast(intent);
```

Utilizzando le trasmissioni ordinate

Le trasmissioni ordinate vengono utilizzate quando è necessario specificare una priorità per i listener di trasmissione.

In questo esempio `firstReceiver` riceverà la trasmissione sempre prima di un secondo `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);
```

Inoltre, il destinatario della trasmissione può interrompere la trasmissione ordinata:

```
@Override
public void onReceive(final Context context, final Intent intent) {
    abortBroadcast();
}
```

in questo caso tutti i ricevitori con priorità inferiore non riceveranno un messaggio broadcast.

Android ha smesso di funzionare

A partire da Android 3.1 tutte le applicazioni, al momento dell'installazione, vengono messe in stato di arresto. Quando è in stato di arresto, l'applicazione non verrà eseguita per nessun motivo, ad eccezione di un avvio manuale di un'attività o di un intento **esplicito** che indirizzi un'attività, un servizio o una trasmissione.

Quando scrivi un'app di sistema che installa gli APK direttamente, tieni presente che l'APP appena installata non riceverà alcuna trasmissione fino a quando non verrà spostata in uno stato non arrestato.

Un modo semplice per attivare un'app è inviare una trasmissione esplicita a questa app. poiché la maggior parte delle app implementa `INSTALL_REFERRER` , possiamo usarlo come punto di aggancio

Esegui la scansione del manifest dell'applicazione installata e invia una trasmissione esplicita a ciascun destinatario:

```
Intent intent = new Intent();
intent.addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES);
intent.setComponent(new ComponentName(packageName, fullClassName));
sendBroadcast(intent);
```

Leggi BroadcastReceiver online: https://riptutorial.com/it/android/topic/1460/broadcastreceiver

Capitolo 39: Camera 2 API

Parametri

Parametro	Dettagli
`CameraCaptureSession`	Una sessione di acquisizione configurata per un dispositivo `CameraDevice` , utilizzata per catturare immagini dalla fotocamera o rielaborare le immagini acquisite dalla fotocamera nella stessa sessione in precedenza
`CameraDevice`	Una rappresentazione di una singola videocamera collegata a un dispositivo Android
`CameraCharacteristics`	Le proprietà che descrivono un dispositivo CameraDevice. Queste proprietà sono corrette per un determinato CameraDevice e possono essere interrogate tramite l'interfaccia di CameraManager con `getCameraCharacteristics(String)`
`CameraManager`	Un gestore di servizi di sistema per il rilevamento, la caratterizzazione e la connessione a `CameraDevices` . È possibile ottenere un'istanza di questa classe chiamando `Context.getSystemService()`
`CaptureRequest`	Un pacchetto immutabile di impostazioni e uscite necessarie per acquisire una singola immagine dal dispositivo della fotocamera. Contiene la configurazione per l'hardware di acquisizione (sensore, obiettivo, flash), la pipeline di elaborazione, gli algoritmi di controllo e i buffer di uscita. Contiene anche l'elenco delle Superfici di destinazione per inviare dati di immagine a per questa cattura. Può essere creato utilizzando un'istanza `CaptureRequest.Builder` , ottenuta chiamando `createCaptureRequest(int)`
`CaptureResult`	Il sottoinsieme dei risultati di una singola acquisizione di immagini dal sensore di immagine. Contiene un sottoinsieme della configurazione finale per l'hardware di acquisizione (sensore, obiettivo, flash), la pipeline di elaborazione, gli algoritmi di controllo e i buffer di uscita. Viene prodotto da un dispositivo `CameraDevice` dopo l'elaborazione di `CaptureRequest`

Osservazioni

- Le API Camera2 sono disponibili in API 21+ (Lollipop e oltre)
- Anche se un dispositivo Android ha ufficialmente una ROM 21+, non è garantito che implementa le API di Camera2, è totalmente compito del produttore implementarlo o meno (Esempio: LG G2 ha il supporto ufficiale di Lollipop, ma nessuna API Camera2)

- Con Camera2, Camera ("Camera1") è deprecato
- Con una grande potenza arriva una grande responsabilità: è più facile rovinarlo quando usi questa API.
- Ricorda, se vuoi solo scattare una foto nella tua app, e semplicemente ottenerla, **non è** necessario implementare Camera2, puoi aprire l'app della fotocamera del dispositivo tramite un Intento e riceverla indietro

Examples

Visualizza l'anteprima della fotocamera principale in un TextureView

In questo caso, costruire contro l'API 23, quindi anche le autorizzazioni sono gestite.

Devi aggiungere nel manifesto la seguente autorizzazione (ovunque il livello API stai usando):

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

Stiamo per creare un'attività (Camera2Activity.java) che riempie un `TextureView` con l'anteprima della fotocamera del dispositivo.

L'attività che useremo è una tipica appCompatActivity:

```
public class Camera2Activity extends AppCompatActivity {
```

Attributi (potrebbe essere necessario leggere l'intero esempio per comprenderne alcuni)

`MAX_PREVIEW_SIZE` garantito da Camera2 API è 1920x1080

```
private static final int MAX_PREVIEW_WIDTH = 1920;
private static final int MAX_PREVIEW_HEIGHT = 1080;
```

`TextureView.SurfaceTextureListener` gestisce diversi eventi del ciclo di vita su un `TextureView` . In questo caso, stiamo ascoltando quegli eventi. Quando SurfaceTexture è pronto, inizializziamo la videocamera. Quando cambia la dimensione, impostiamo l'anteprima proveniente dalla fotocamera di conseguenza

```
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 dispositivo `CameraDevice` rappresenta la videocamera di un dispositivo fisico. In questo attributo, salviamo l'ID del dispositivo `CameraDevice` corrente

```
private String mCameraId;
```

Questa è la vista (`TextureView`) che useremo per "disegnare" l'anteprima della Camera

```
private TextureView mTextureView;
```

`CameraCaptureSession` per l'anteprima della fotocamera

```
private CameraCaptureSession mCaptureSession;
```

Un riferimento al `CameraDevice` aperto

```
private CameraDevice mCameraDevice;
```

La `Size` dell'anteprima della fotocamera.

```
private Size mPreviewSize;
```

`CameraDevice.StateCallback` viene chiamato quando `CameraDevice` modifica il suo stato

```
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 thread aggiuntivo per l'esecuzione di attività che non dovrebbero bloccare l'interfaccia utente

```
private HandlerThread mBackgroundThread;
```

Un `Handler` per eseguire attività in background

```
private Handler mBackgroundHandler;
```

Un `ImageReader` che gestisce l'acquisizione di immagini statiche

```
private ImageReader mImageReader;
```

`CaptureRequest.Builder` per l'anteprima della fotocamera

```
private CaptureRequest.Builder mPreviewRequestBuilder;
```

`CaptureRequest` generato da `mPreviewRequestBuilder`

```
private CaptureRequest mPreviewRequest;
```

Un `Semaphore` per impedire all'app di uscire prima di chiudere la fotocamera.

```
private Semaphore mCameraOpenCloseLock = new Semaphore(1);
```

ID costante della richiesta di autorizzazione

```
private static final int REQUEST_CAMERA_PERMISSION = 1;
```

Metodi Android Lifecycle

```
@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();
}
```

Metodi correlati a Camera2

Quelli sono metodi che utilizzano le API 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);
    }
}
```

Chiude la videocamera corrente

```
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();
    }
}
```

Imposta le variabili membro relative alla telecamera

```
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 nuova `CameraCaptureSession` per l'anteprima della videocamera

```
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();
    }
}
```

Metodi correlati alle autorizzazioni Per l'API Android 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);
    }
}
```

Metodi di thread di background / handler

```
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();
    }
}
```

Metodi di utilità

Date le scelte di `Size` s supportate da una telecamera, scegliete quella più piccola che è almeno ampia come le rispettive dimensioni della vista della trama, e che è grande quanto la rispettiva dimensione massima e le cui proporzioni corrispondono al valore specificato. Se non esiste, scegli il più grande che sia grande al massimo come la rispettiva dimensione massima e le cui proporzioni corrispondono al valore specificato

```
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];
    }
}
```

Questo metodo congfigura la trasformazione `mTextureView Matrix` in `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);
}
```

Questo metodo confronta due `Size` base alle loro aree.

```
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());
    }
}
```

Non c'è molto da vedere qui

```
/**
 * 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();
        }
    });
}
```

Leggi Camera 2 API online: https://riptutorial.com/it/android/topic/619/camera-2-api

Capitolo 40: Canale di notifica Android O

introduzione

I canali di notifica ci consentono agli sviluppatori di app di raggruppare le nostre notifiche in gruppi-canali con l'utente che ha la possibilità di modificare le impostazioni di notifica per l'intero canale contemporaneamente. In Android O questa funzione è stata introdotta. Ora è disponibile l'anteprima degli sviluppatori.

Sintassi

1. class NotificationUtils {} // Per creare un canale di notifica
2. createChannel () // Metodo generico per la creazione di notifiche

Parametri

Metodo	Descrizione
IMPORTANCE_MAX	non usato
IMPORTANCE_HIGH	mostra ovunque, fa rumore e sbircia
IMPORTANCE_DEFAULT	mostra ovunque, fa rumore, ma non si intromette visivamente
IMPORTANCE_LOW	mostra ovunque, ma non è invadente
IMPORTANCE_MIN	mostra solo all'ombra, sotto la piega
IMPORTANCE_NONE	una notifica senza importanza; non mostra nell'ombra

Examples

Canale di notifica

Quali sono i canali di notifica?

I canali di notifica ci consentono agli sviluppatori di app di raggruppare le nostre notifiche in canali di gruppo, con l'utente che ha la possibilità di modificare le impostazioni di notifica per l'intero canale contemporaneamente. Ad esempio, per ciascun canale, gli utenti possono bloccare completamente tutte le notifiche, ignorare il livello di importanza o consentire la visualizzazione di un badge di notifica. Questa nuova funzione aiuta a migliorare notevolmente l'esperienza utente di un'app.

Crea canali di notifica

```
 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;
}}
```

Nel codice sopra, abbiamo creato due istanze di NotificationChannel, passando uniqueid un nome di canale, e anche un livello di importanza nel suo costruttore. Per ogni canale di notifica, abbiamo applicato le seguenti caratteristiche.

1. Suono
2. Luci
3. Vibrazione
4. Notifica da mostrare sulla schermata di blocco.

Infine, abbiamo ottenuto NotificationManager dal sistema e registrato il canale chiamando il metodo createNotificationChannel (), passando il canale che abbiamo creato.

Possiamo creare più canali di notifica contemporaneamente con createNotificationChannels (), passando un elenco Java di istanze di NotificationChannel. Puoi ottenere tutti i canali di notifica per un'app con getNotificationChannels () e ottenere un canale specifico con getNotificationChannel (), passando solo l'id del canale come argomento.

Livello di importanza nei canali di notifica

Metodo	Descrizione
IMPORTANCE_MAX	non usato
IMPORTANCE_HIGH	mostra ovunque, fa rumore e sbircia
IMPORTANCE_DEFAULT	mostra ovunque, fa rumore, ma non si intromette visivamente
IMPORTANCE_LOW	mostra ovunque, ma non è invadente, il valore è 0
IMPORTANCE_MIN	mostra solo all'ombra, sotto la piega
IMPORTANCE_NONE	una notifica senza importanza; non mostra nell'ombra

Crea notifiche e invia al canale

Abbiamo creato due notifiche una usando NotificationUtils un'altra 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);
}
```

Possiamo anche impostare NotificationChannel usando Notification.Builder (). Per questo possiamo usare ***setChannel (String channelId)*** .

Aggiorna le impostazioni del canale di notifica

Una volta creato un canale di notifica, l'utente è responsabile delle sue impostazioni e del suo comportamento. È possibile chiamare nuovamente createNotificationChannel () per rinominare un canale di notifica esistente o aggiornarne la descrizione. Il seguente codice di esempio descrive come è possibile reindirizzare un utente alle impostazioni per un canale di notifica creando un intent per avviare un'attività. In questo caso, l'intento richiede dati estesi incluso l'ID del canale di notifica e il nome del pacchetto della tua app.

```
@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);
        }
    });
}
```

File XML:

```
<!--...-->
<Button
    android:id="@+id/btn_android_notif_settings"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Notification Settings"/>
<!--...-->
```

Eliminazione del canale di notifica

È possibile eliminare i canali di notifica chiamando 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);
```

Ora crea MainActivity e 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

modificheremo la nostra MainActivity in modo da poter ottenere il titolo e l'autore dai componenti EditText e quindi inviarli al canale Android. Otteniamo Notification.Builder per il canale Android che abbiamo creato nel nostro NotificationUtils e quindi notifichiamo NotificationManager.

importare android.app.Notification; importare android.os.Bundle; import android.support.v7.app.AppCompatActivity; importare android.text.TextUtils; importa android.view.View; importa android.widget.Button; importare 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());
                }
            }
        });
    }
}
```

Leggi Canale di notifica Android O online: https://riptutorial.com/it/android/topic/10018/canale-di-

notifica-android-o

Capitolo 41: Caratteri personalizzati

Examples

Inserire un carattere personalizzato nella tua app

1. Vai alla (cartella del progetto)
2. Quindi app -> src -> main.
3. Crea cartella 'assets -> fonts' nella cartella principale.
4. Metti il tuo 'fontfile.ttf' nella cartella dei caratteri.

Inizializzazione di un font

```
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");
}
```

Utilizzando un carattere personalizzato in un TextView

```
public void setFont(TextView textView) {
    textView.setTypeface(myFont);
}
```

Applicare font su TextView da xml (codice Java non richiesto)

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: (Dove posizionare **res / valori**)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="TextViewPlus">
        <attr name="customFont" format="string"/>
    </declare-styleable>
</resources>
```

Come usare:

```
<?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>
```

Carattere personalizzato nel testo della tela

Disegna il testo su tela con il tuo font dalle risorse.

```
Typeface typeface = Typeface.createFromAsset(getAssets(), "fonts/SomeFont.ttf");
Paint textPaint = new Paint();
textPaint.setTypeface(typeface);
canvas.drawText("Your text here", x, y, textPaint);
```

Caricamento del carattere tipografico efficiente

Il caricamento di caratteri personalizzati può comportare un cattivo rendimento. Consiglio vivamente di utilizzare questo piccolo helper che salva / carica i font già utilizzati in 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);
```

Carattere personalizzato per l'intera attività

```
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();
    }
}
```

Quindi nella tua attività, nel metodo `onCreate()` :

```
// Put your font to assets folder...

ReplaceFont.changeDefaultFont(getApplication(), "DEFAULT", "LinLibertine.ttf");
```

Lavorare con i caratteri in Android O

Android O cambia il modo di lavorare con i caratteri.

Android O introduce una nuova funzionalità, chiamata *Fonts in XML* , che consente di utilizzare i caratteri come risorse. Ciò significa che non è necessario raggruppare i font come risorse. I caratteri sono ora compilati in un file *R* e sono automaticamente disponibili nel sistema come una risorsa.

Per aggiungere un nuovo **font** , devi fare quanto segue:

- Crea una nuova directory di risorse: `res/font` .
- Aggiungi i tuoi file di font in questa cartella di font. Ad esempio, aggiungendo `myfont.ttf` , sarà possibile utilizzare questo carattere tramite `R.font.myfont` .

Puoi anche creare la tua **famiglia di caratteri** aggiungendo il seguente file XML nella directory `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>
```

È possibile utilizzare sia il file di **font** e il file **famiglia di font** nello stesso modo:

- **In un file XML,** utilizzando l'attributo `android:fontFamily` , ad esempio in questo modo:

  ```
  <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:fontFamily="@font/myfont"/>
  ```

  O in questo modo:

  ```
  <style name="customfontstyle" parent="@android:style/TextAppearance.Small">
      <item name="android:fontFamily">@font/myfont</item>
  </style>
  ```

- **Nel codice** , utilizzando le seguenti righe di codice:

```
Typeface typeface = getResources().getFont(R.font.myfont);
textView.setTypeface(typeface);
```

Leggi Caratteri personalizzati online: https://riptutorial.com/it/android/topic/3358/caratteri-personalizzati

Capitolo 42: CardView

introduzione

Un FrameLayout con uno sfondo e un'ombra arrotondati.

CardView utilizza la proprietà elevation su Lollipop per le ombre e ricade su un'implementazione shadow emulata personalizzata su piattaforme meno recenti.

A causa della natura costosa del ritaglio dell'angolo arrotondato, su piattaforme prima di Lollipop, CardView non ritaglia i suoi figli che si intersecano con gli angoli arrotondati. Invece, aggiunge padding per evitare tale intersezione (vedi setPreventCornerOverlap (booleano) per cambiare questo comportamento).

Parametri

Parametro	Dettagli
cardBackgroundColor	Colore di sfondo per CardView.
cardCornerRadius	Raggio d'angolo per CardView.
cardElevation	Elevazione per CardView.
cardMaxElevation	Elevazione massima per CardView.
cardPreventCornerOverlap	Aggiungi padding a CardView su v20 e prima per evitare intersezioni tra il contenuto della Carta e gli angoli arrotondati.
cardUseCompatPadding	Aggiungi padding in API v21 + per avere le stesse misurazioni con le versioni precedenti. Può essere un valore booleano, come "true" o "false".
contentPadding	Imbottitura interna tra i bordi della Card e i bambini di CardView.
contentPaddingBottom	Imbottitura interna tra il bordo inferiore della scheda e i bambini di CardView.
contentPaddingLeft	Imbottitura interna tra il bordo sinistro della scheda e i bambini del CardView.
contentPaddingRight	Elevazione per CardView.
cardElevation	Imbottitura interna tra il bordo destro della Card e i bambini del CardView.
contentPaddingTop	Imbottitura interna tra il bordo superiore della Card e i bambini

Parametro	Dettagli
	del CardView.

Osservazioni

`CardView` utilizza l'elevazione reale e le ombre dinamiche su Lollipop (API 21) e sopra. Tuttavia, prima che Lollipop `CardView` ricada su un'implementazione shadow programmatica.

Se cerchi di realizzare un `ImageView` all'interno degli angoli arrotondati di un `CardView` , potresti notare che non sembra corretto pre-Lollipop (API 21). Per risolvere questo problema, devi chiamare `setPreventCornerOverlap(false)` sul tuo `CardView` o aggiungere `app:cardPreventCornerOverlap="false"` al tuo layout.

Prima di usare `CardView` devi aggiungere la dipendenza della libreria di supporto nel file `build.gradle` :

```
dependencies{
    compile 'com.android.support:cardview-v7:25.2.0'
}
```

Un numero della versione più recente può essere trovato qui

Documentazione ufficiale:

https://developer.android.com/reference/android/support/v7/widget/CardView.html
https://developer.android.com/training/material/lists-cards.html

Examples

Iniziare con CardView

`CardView` è un membro della libreria di supporto Android e fornisce un layout per le carte.

Per aggiungere `CardView` al tuo progetto, aggiungi la seguente riga alle dipendenze `build.gradle` .

```
compile 'com.android.support:cardview-v7:25.1.1'
```

Un numero della versione più recente può essere trovato qui

Nel tuo layout puoi quindi aggiungere quanto segue per ottenere una carta.

```
<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>
```

È quindi possibile aggiungere altri layout all'interno di questo e saranno racchiusi in una scheda.

Inoltre, CardView può essere popolato con qualsiasi elemento dell'interfaccia utente e manipolato dal codice .

```
<?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>
```

Personalizzazione di CardView

CardView fornisce un elevazione e un raggio di punta predefiniti in modo che le carte abbiano un

aspetto coerente attraverso le piattaforme.

È possibile personalizzare questi valori predefiniti utilizzando questi attributi nel file xml:

1. `card_view:cardElevation` attributo `card_view:cardElevation` aggiunge l'elevazione in CardView.
2. `card_view:cardBackgroundColor` attributo `card_view:cardBackgroundColor` viene utilizzato per personalizzare il colore di sfondo dello sfondo di CardView (puoi dare qualsiasi colore).
3. `card_view:cardCornerRadius` attributo `card_view:cardCornerRadius` viene utilizzato per curvare i 4 bordi di CardView
4. `card_view:contentPadding` attributo `card_view:contentPadding` aggiunge il padding tra la carta e i figli della carta

Nota: card_view è uno spazio dei nomi definito nella vista layout principale in alto. xmlns: card_view = " http://schemas.android.com/apk/res-auto "

Ecco un esempio:

```
<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>
```

Puoi anche farlo a livello di programmazione usando:

```
card.setCardBackgroundColor(....);
card.setCardElevation(...);
card.setRadius(....);
card.setContentPadding();
```

Controlla la javadoc ufficiale per ulteriori proprietà.

Aggiungere animazione Ripple

Per abilitare l'animazione ripple in un CardView, aggiungi i seguenti attributi:

```
<android.support.v7.widget.CardView
  ...
  android:clickable="true"
  android:foreground="?android:attr/selectableItemBackground">
  ...
</android.support.v7.widget.CardView>
```

Uso delle immagini come sfondo in CardView (problemi relativi al dispositivo

Pre-Lollipop)

Mentre si utilizza Immagine / Colore come sfondo in un CardView, si potrebbero finire con leggeri rilievi bianchi (se il colore predefinito della Carta è bianco) sui bordi. Ciò si verifica a causa degli angoli arrotondati predefiniti nella Vista scheda. Ecco come evitare questi margini nei dispositivi Pre-lollipop.

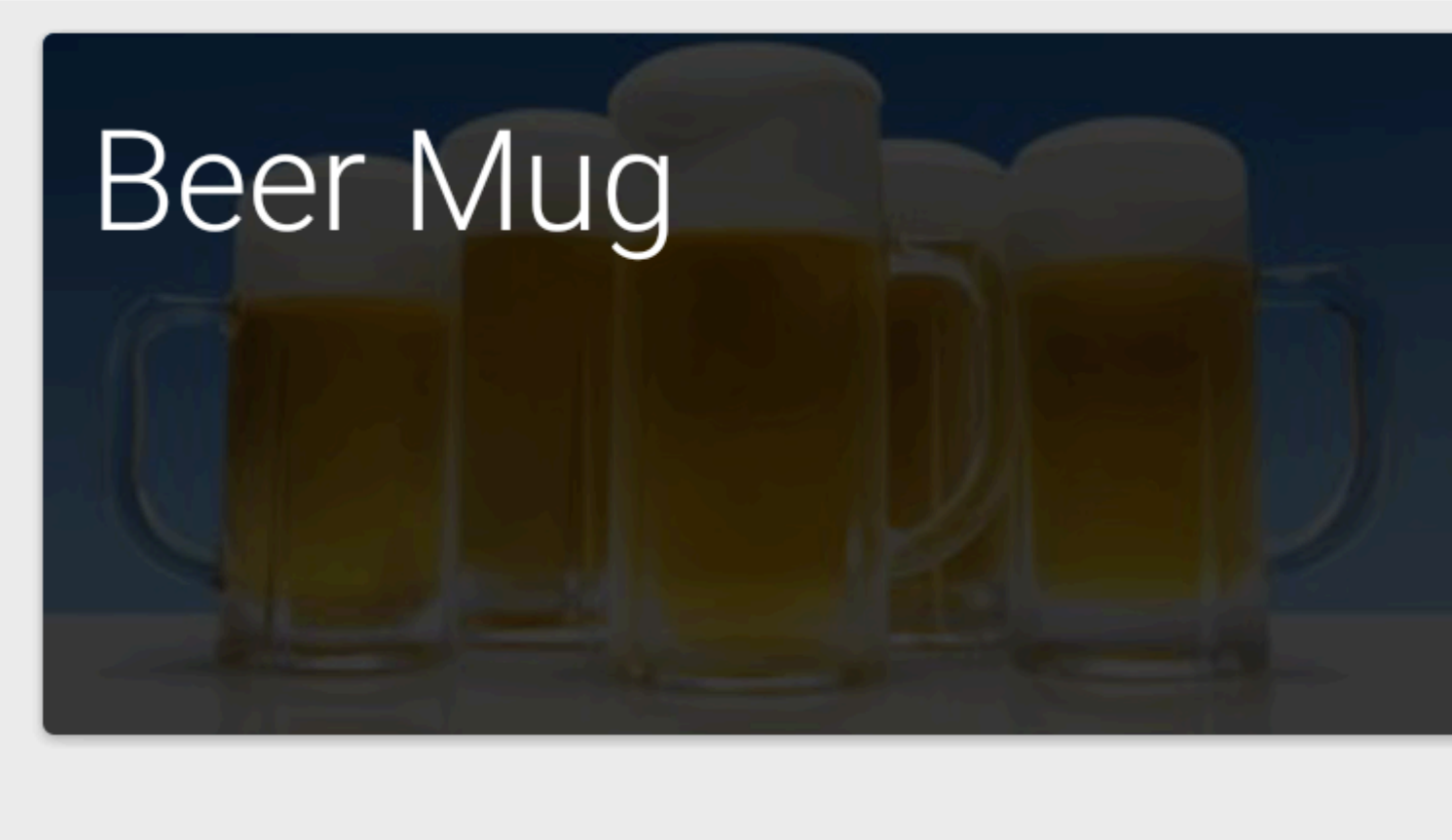
Dobbiamo usare un attributo `card_view:cardPreventCornerOverlap="false"` nel CardView. 1). In XML usa il seguente frammento.

```
<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>
```

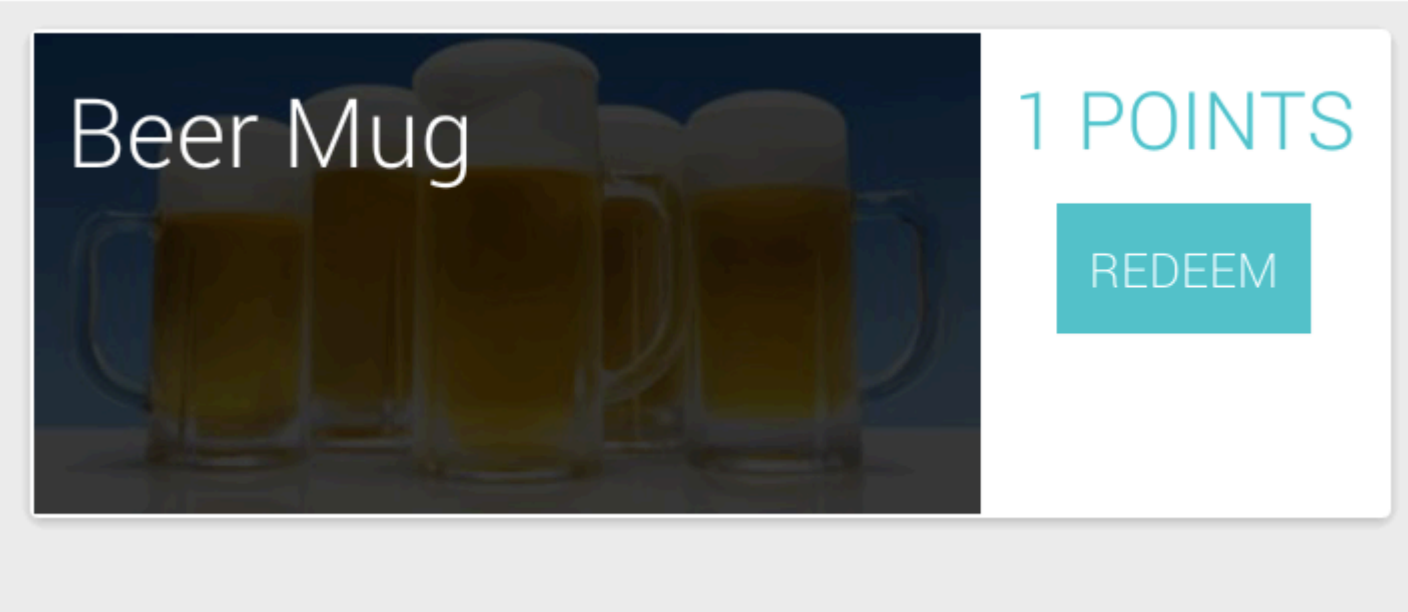
2. In Java come questo `cardView.setPreventCornerOverlap(false)` .

In questo modo si rimuove una imbottitura indesiderata sui bordi della scheda. Ecco alcuni esempi visivi relativi a questa implementazione.

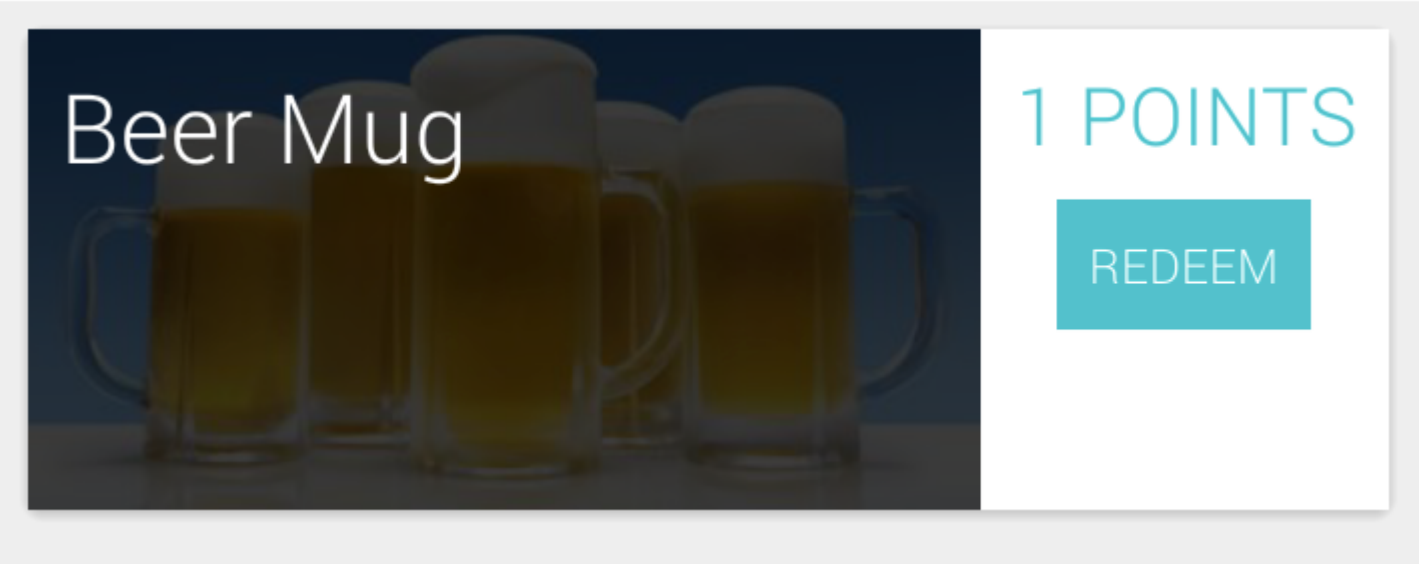
1 scheda con sfondo immagine in API 21 (perfettamente funzionante)



2 Scheda con sfondo immagine in API 19 senza attributi (notare i riquadri intorno all'immagine)



3 Scheda FISSA con sfondo immagine in API 19 con attributo
`cardView.setPreventCornerOverlap(false)` (Problema ora risolto)



Leggi anche questo su Documentazione qui
Post SOF originale qui

Anima il colore di sfondo CardView 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);
}
```

Leggi CardView online: https://riptutorial.com/it/android/topic/726/cardview

Capitolo 43: Caricamento efficace di bitmap

introduzione

Questo argomento si concentra principalmente sul caricamento efficace dei bitmap nei dispositivi Android.

Quando si tratta di caricare una bitmap, la domanda arriva da dove viene caricata. Qui discuteremo su come caricare Bitmap dalla risorsa con il dispositivo Android. ad es. dalla galleria.

Passeremo attraverso questo esempio che viene discusso di seguito.

Sintassi

- `<uses-permission>` **->** Tag utilizzato per l'autorizzazione.
- `android:name` **->** Un attributo usato per dare un nome al permesso che stiamo per richiedere.
- `android.permission.READ_EXTERNAL_STORAGE` **->** Sono permessi di sistema
- esempio "android.permission.CAMERA" o "android.permission.READ_CONTACTS"

Examples

Carica l'immagine dalla risorsa dal dispositivo Android. Utilizzo di Intenti.

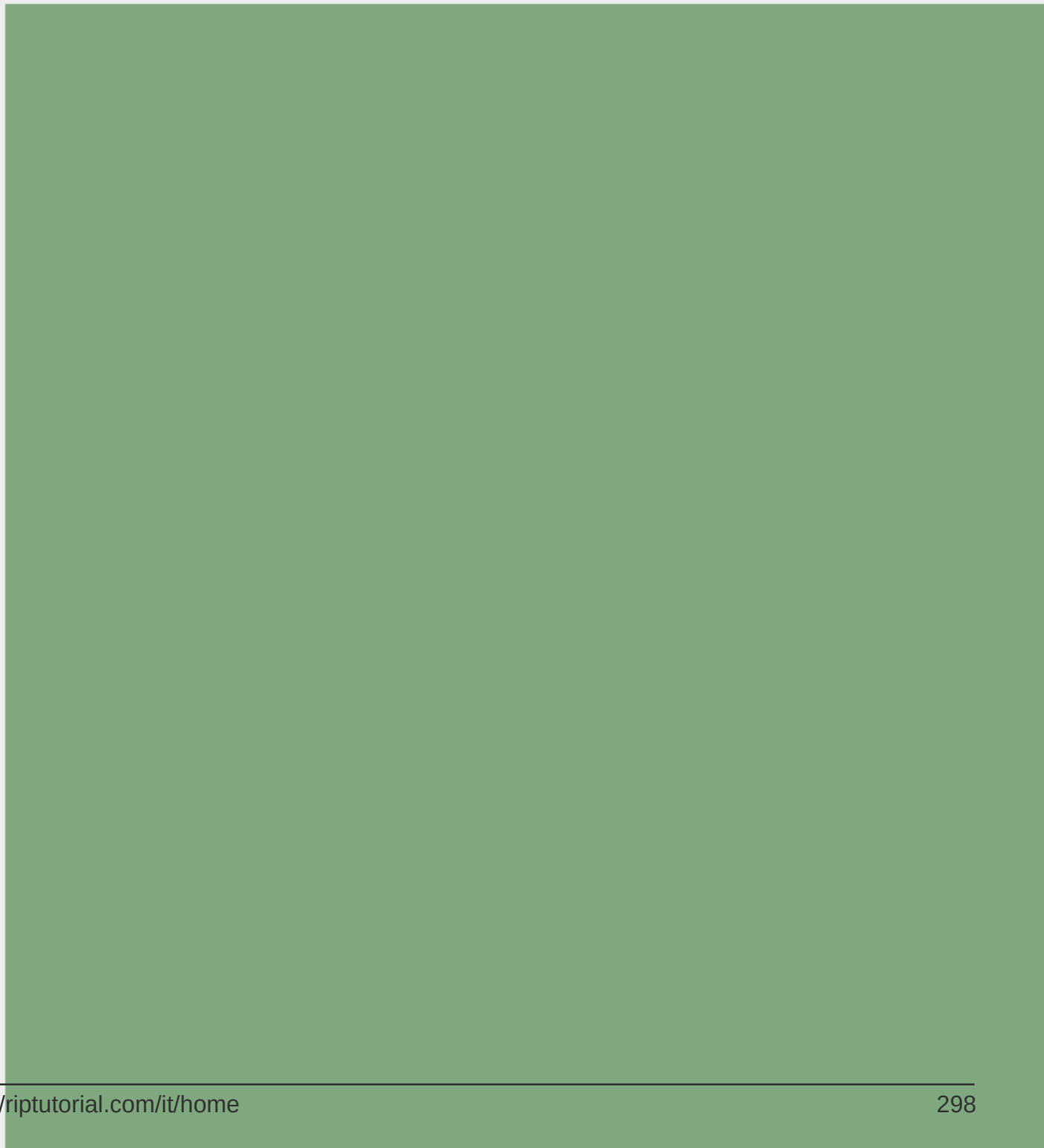
Utilizzo di Intenti per caricare l'immagine dalla galleria.

1. Inizialmente è necessario il permesso

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

2. Utilizzare il seguente codice per avere il layout come progettato di seguito.

LoadImageFrmGallery



https://riptutorial.com/it/android/topic/10902/caricamento-efficace-di-bitmap

Capitolo 44: caricatore

introduzione

Loader è una buona scelta per prevenire perdite di memoria se si desidera caricare i dati in background quando viene chiamato il metodo oncreate. Ad esempio, quando eseguiamo Asynctask nel metodo oncreate e ruotiamo lo schermo in modo da ricreare l'attività che eseguirà di nuovo un altro AsyncTask, quindi probabilmente due Asyntask in esecuzione in parallelo piuttosto che come loader che continuerà il processo in background eseguito in precedenza.

Parametri

Classe	Descrizione
LoaderManager	Una classe astratta associata ad un attività o frammento per la gestione di una o più istanze Loader.
LoaderManager.LoaderCallbacks	Un'interfaccia di callback per un client per interagire con LoaderManager.
caricatore	Una classe astratta che esegue il caricamento asincrono dei dati.
AsyncTaskLoader	Caricatore astratto che fornisce un AsyncTask per eseguire il lavoro.
CursorLoader	Una sottoclasse di AsyncTaskLoader che interroga ContentResolver e restituisce un Cursore.

Osservazioni

Introdotto in Android 3.0, i caricatori semplificano il caricamento asincrono dei dati in un'attività o in un frammento. I caricatori hanno queste caratteristiche:

- Sono disponibili per ogni attività e frammento .
- Forniscono il caricamento asincrono dei dati.
- Monitorano la fonte dei loro dati e forniscono nuovi risultati quando il contenuto cambia.
- Si riconnettono automaticamente al cursore dell'ultimo caricatore quando vengono ricreati dopo una modifica della configurazione. Quindi, non hanno bisogno di ri-interrogare i loro dati.

Quando non usare i caricatori

Non è necessario utilizzare Caricatori se è necessario completare le attività in background.

Android distrugge i caricatori insieme alle attività / ai frammenti a cui appartengono. Se si desidera eseguire alcune attività, che devono essere eseguite fino al completamento, non utilizzare Caricatori. Dovresti invece usare i servizi per questo genere di cose.

Examples

AsyncTaskLoader di base

`AsyncTaskLoader` è un programma di `Loader` astratto che fornisce un `AsyncTask` per eseguire il lavoro.

Ecco alcune implementazioni di base:

```
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();
    }
}
```

In genere `Loader` viene inizializzato all'interno del metodo `onCreate()` dell'attività o all'interno di `onActivityCreated()` del frammento. Inoltre, solitamente l'attività o il frammento implementa `LoaderManager.LoaderCallbacks` interfaccia `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) {
    }
}
```

In questo esempio, quando il caricatore è completato, verrà mostrato un brindisi con il risultato.

AsyncTaskLoader con cache

È buona norma memorizzare nella cache i risultati caricati per evitare il caricamento multiplo degli stessi dati.
Per invalidare la cache `onContentChanged()` dovrebbe essere chiamato. Se il loader è già stato avviato, verrà chiamato `forceLoad()` , altrimenti (se il caricatore è in stato di arresto) il caricatore sarà in grado di comprendere la modifica del contenuto con il controllo `takeContentChanged()` .

Nota: `onContentChanged()` deve essere chiamato dal thread principale del processo.

Javadocs dice di takeContentChanged ():

> Prendi il flag corrente per indicare se il contenuto del caricatore è stato modificato mentre era fermo. In tal caso, viene restituito true e la bandiera viene cancellata.

```
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
    }
}
```

Ricaricamento

Per invalidare i vecchi dati e riavviare il caricatore esistente è possibile utilizzare il metodo `restartLoader()` :

```
private void reload() {
    getLoaderManager().reastartLoader(LOADER_ID, Bundle.EMPTY, this);
}
```

Passa i parametri usando un pacchetto

Puoi passare i parametri per Bundle:

```
Bundle myBundle = new Bundle();
myBundle.putString(MY_KEY, myValue);
```

Ottieni il valore in onCreateLoader:

```
@Override
public Loader<String> onCreateLoader(int id, final Bundle args) {
    final String myParam = args.getString(MY_KEY);
    ...
}
```

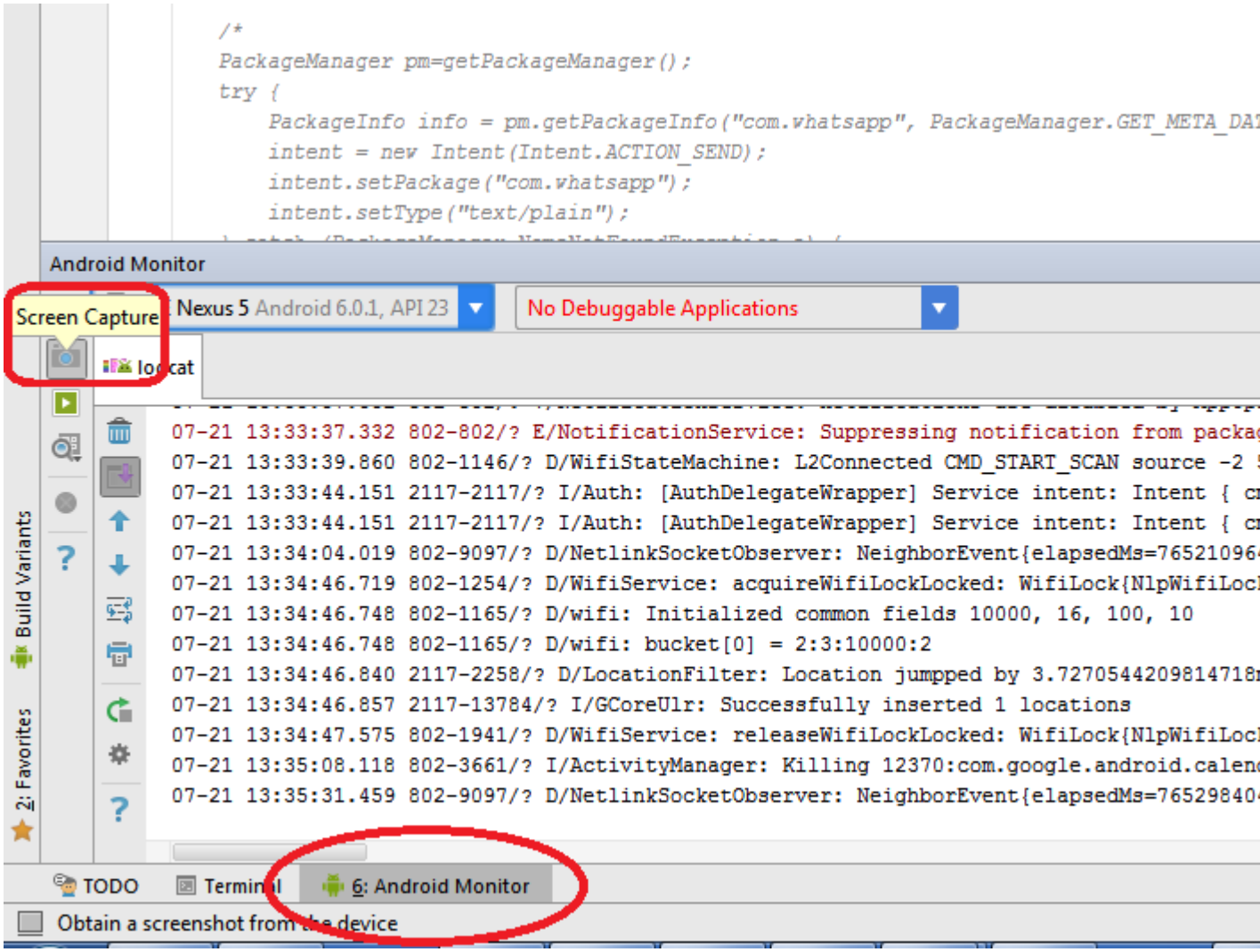
Leggi caricatore online: https://riptutorial.com/it/android/topic/4390/caricatore

Capitolo 45: Cattura di screenshot

Examples

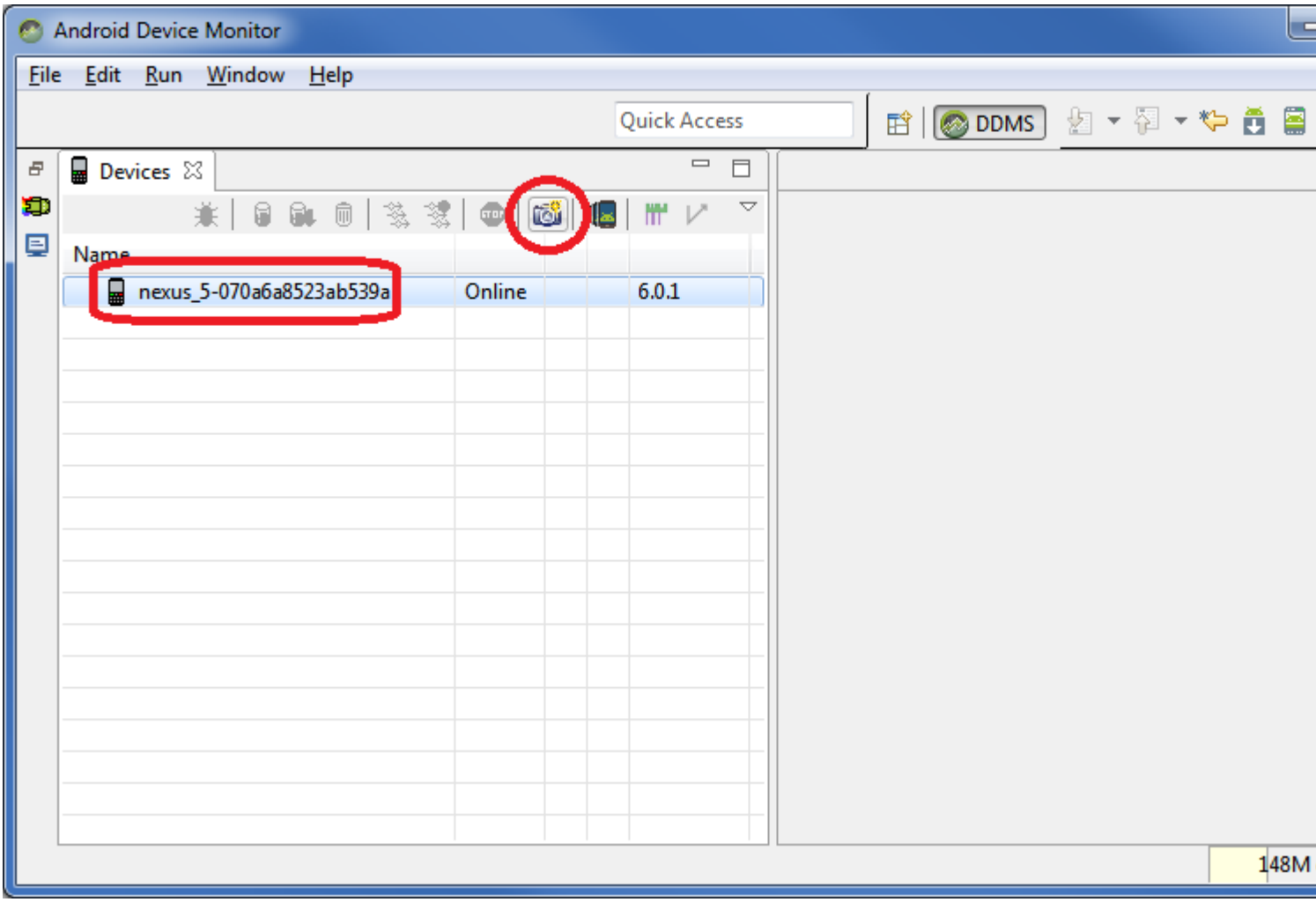
Cattura Screenshot tramite Android Studio

1. Apri la scheda Monitor Android
2. Fare clic sul pulsante Cattura schermo



Cattura di screenshot tramite Monitor dispositivo Android

1. Apri Monitor dispositivo Android (ad *es. C: <ANDROID_SDK_LOCATION> \ tools \ monitor.bat*)
2. Seleziona il tuo dispositivo
3. Fare clic sul pulsante Cattura schermo



Cattura Screenshot tramite ADB

L'esempio seguente salva uno screenshot sulla memoria interna dei dispositivi.

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

Cattura Screenshot tramite ADB e salva direttamente nel tuo PC

Se usi Linux (o Windows con Cygwin), puoi eseguire:

```
adb shell screencap -p | sed 's/\r$//' > screenshot.png
```

Scattare uno screenshot di una vista particolare

Se vuoi fare uno screenshot di una particolare View `v` , allora puoi usare il seguente codice:

```
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();
}
```

Leggi Cattura di screenshot online: https://riptutorial.com/it/android/topic/4506/cattura-di-screenshot

Capitolo 46: CleverTap

introduzione

Quick hacks per l'analisi e l'engagement SDK fornito da CleverTap - Android

Osservazioni

Ottieni le tue credenziali CleverTap da https://clevertap.com .

Examples

Ottieni un'istanza dell'SDK per registrare eventi

```
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
}
```

Impostazione del livello di debug

Nella classe dell'applicazione personalizzata, sovrascrivi il metodo `onCreate()` , aggiungi la riga seguente:

```
CleverTapAPI.setDebugLevel(1);
```

Leggi CleverTap online: https://riptutorial.com/it/android/topic/9337/clevertap

Capitolo 47: Colori

Examples

Manipolazione del colore

Per manipolare i colori modificheremo i valori argb (Alfa, Rosso, Verde e Blu) di un colore.

Per prima cosa estrai i valori RGB dal tuo colore.

```
int yourColor = Color.parse("#ae1f67");

int red = Color.red(yourColor);
int green = Color.green(yourColor);
int blue = Color.blue(yourColor);
```

Ora puoi ridurre o aumentare i valori di rosso, verde e blu e combinarli nuovamente come colore:

```
 int newColor = Color.rgb(red, green, blue);
```

O se vuoi aggiungere dell'alpha, puoi aggiungerlo mentre crei il colore:

```
 int newColor = Color.argb(alpha, red, green, blue);
```

I valori alfa e RGB dovrebbero essere compresi nell'intervallo [0-225].

Leggi Colori online: https://riptutorial.com/it/android/topic/4986/colori

Capitolo 48: Coltello da burro

introduzione

Butterknife è uno strumento di associazione delle viste che utilizza le annotazioni per generare il codice boilerplate per noi. Questo strumento è stato sviluppato da Jake Wharton in Square ed è essenzialmente utilizzato per risparmiare la digitazione di righe ripetitive di codice come `findViewById(R.id.view)` quando si tratta di visualizzazioni, rendendo il nostro codice molto più pulito.

Per essere chiari, Butterknife **non** è **una libreria di dipendenze** . Butterknife inietta il codice al momento della compilazione. È molto simile al lavoro svolto da Android Annotations.

Osservazioni

Coltello da burro

Associazione dei campi e dei metodi per le viste Android che utilizza l'elaborazione delle annotazioni per generare automaticamente il codice di zona.

- Elimina le chiamate findViewById utilizzando @BindView nei campi.
- Raggruppa più viste in un elenco o array. Operare su tutti contemporaneamente con azioni, setter o proprietà.
- Elimina le inner class anonime per gli ascoltatori annotando i metodi con @OnClick e altri.
- Elimina le ricerche di risorse utilizzando annotazioni di risorse sui campi.

Maggiori informazioni: http://jakewharton.github.io/butterknife/

Licenza

Examples

Configurare ButterKnife nel tuo progetto

Configura il tuo `build.gradle` a livello di `build.gradle` per includere il plugin per `android-apt` :

```
buildscript {
   repositories {
      mavenCentral()
   }

   dependencies {
      classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
   }
}
```

Quindi, applica il plugin per `android-apt` nel `build.gradle` livello di `build.gradle` e aggiungi le dipendenze di ButterKnife:

```
apply plugin: 'android-apt'

android {
    ...
}

dependencies {
      compile 'com.jakewharton:butterknife:8.5.1'
      annotationProcessor 'com.jakewharton:butterknife-compiler:8.5.1'
}
```

Nota: se stai usando il nuovo compilatore Jack con la versione 2.2.0 o più recente non hai bisogno del plugin per `android-apt` e puoi invece sostituire apt con `annotationProcessor` quando dichiari la dipendenza del compilatore.

Per utilizzare le annotazioni di ButterKnife, non dimenticare di `onCreate()` in `onCreate()` delle tue attività o `onCreateView()` dei tuoi frammenti:

```
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;
  }

}
```

Le istantanee della versione di sviluppo sono disponibili nel repository di istantanee di Sonatype .

Di seguito sono riportati i passaggi aggiuntivi che dovresti fare per utilizzare ButterKnife in un progetto di libreria

Per usare ButterKnife in un progetto di libreria, aggiungi il plugin al tuo `build.gradle` livello di `build.gradle` :

```
buildscript {
    dependencies {
        classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
    }
}
```

... e poi applica al tuo modulo aggiungendo queste righe nella parte superiore del tuo livello di libreria `build.gradle` :

```
apply plugin: 'com.android.library'
// ...
apply plugin: 'com.jakewharton.butterknife'
```

Ora assicurati di usare `R2` invece di `R` all'interno di tutte le annotazioni 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
  }
```

```
}
```

Visualizzazioni vincolanti con ButterKnife

possiamo annotare campi con `@BindView` e un ID vista per Butter Knife per trovare e lanciare automaticamente la vista corrispondente nel nostro layout.

Visualizzazioni vincolanti

Visualizzazioni vincolanti in attività

```
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...
  }
}
```

Viste vincolanti in frammenti

```
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();
    }
}
```

Visualizzazioni vincolanti nelle finestre di dialogo

Possiamo usare `ButterKnife.findById` per trovare viste su una vista, attività o finestra di dialogo. Usa i generici per inferire il tipo di ritorno e esegue automaticamente il cast.

```
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);
```

Visualizzazioni vincolanti in 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);
    }
  }
```

Risorse vincolanti

Oltre ad essere utile per le viste vincolanti, è possibile utilizzare ButterKnife per associare risorse come quelle definite all'interno di `strings.xml` , `drawables.xml` , `colors.xml` , `dimens.xml` , ecc.

```
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);
    }

}
```

Elenchi di vista vincolanti

È possibile raggruppare più viste in un elenco o array. Questo è molto utile quando è necessario eseguire un'azione su più viste contemporaneamente.

```
@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);
  }
};
```

Attacchi opzionali

Per impostazione predefinita, sono richiesti entrambi i binding di `@Bind` e listener. Viene generata un'eccezione se non è possibile trovare la vista di destinazione. Ma se non siamo sicuri che una vista sarà presente o meno, possiamo aggiungere un'annotazione `@Nullable` ai campi o l'annotazione `@Optional` ai metodi per sopprimere questo comportamento e creare un'associazione facoltativa.

```
@Nullable
@BindView(R.id.might_not_be_there) TextView mightNotBeThere;

@Optional
@OnClick(R.id.maybe_missing)
void onMaybeMissingClicked() {
  // TODO ...
}
```

Ascoltatori vincolanti con ButterKnife

Listener di OnClick:

```
@OnClick(R.id.login)
public void login(View view) {
  // Additional logic
}
```

Tutti gli argomenti del metodo listener sono facoltativi:

```
@OnClick(R.id.login)
public void login() {
   // Additional logic
}
```

Il tipo specifico verrà automaticamente castato:

```
@OnClick(R.id.submit)
public void sayHi(Button button) {
  button.setText("Hello!");
}
```

Più ID in un singolo binding per la gestione di eventi comuni:

```
@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();
  }
}
```

Le viste personalizzate possono associarsi ai propri ascoltatori non specificando un ID:

```
public class CustomButton extends Button {
  @OnClick
  public void onClick() {
    // TODO
  }
}
```

Visualizzazioni non vincenti in ButterKnife

I frammenti hanno un ciclo di vita di visualizzazione diverso rispetto alle attività. Quando si associa un frammento in onCreateView, impostare le viste su null in onDestroyView. Butter Knife restituisce un'istanza di Unbinder quando chiami bind per farlo per te. Chiamare il metodo unbind nel callback del ciclo di vita appropriato.

Un esempio:

```
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();
  }
}
```

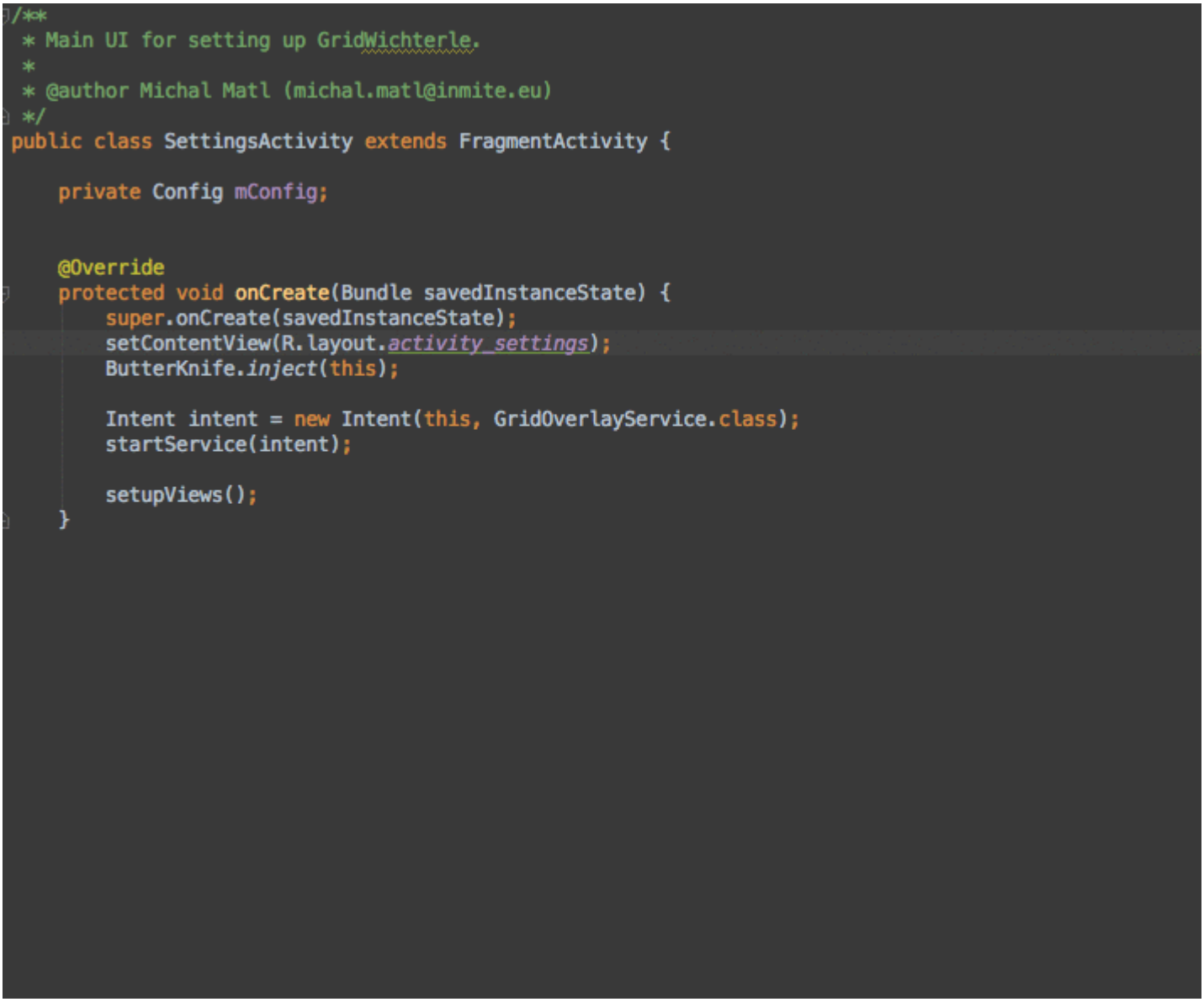
Nota: la chiamata di unbind () in onDestroyView () non è richiesta, ma è consigliata in quanto consente di risparmiare un bel po 'di memoria se l'app ha un backstack di grandi dimensioni.

Android Studio ButterKnife Plugin

Android ButterKnife Zelezny

Plugin per generare iniezioni ButterKnife da XML di layout selezionati in attività / frammenti / adattatori.

Nota: assicurati di fare clic con il pulsante destro del mouse su ***your_xml_layou*** `(R.layout.your_xml_layou`) altrimenti il *menu Genera* non conterrà l'opzione Iniettore Butterknife.



Link: Plugin Jetbrains Android ButterKnife Zelezny

Leggi Coltello da burro online: https://riptutorial.com/it/android/topic/1072/coltello-da-burro

Capitolo 49: Come conservare le password in modo sicuro

Examples

Utilizzo di AES per la crittografia della password salata

Questo esempio utilizza l'algoritmo AES per crittografare le password. La lunghezza del sale può essere fino a 128 bit.

Stiamo usando la classe `SecureRandom` per generare un salt, che è combinato con la password per generare una chiave segreta. Le classi utilizzate sono già presenti nei pacchetti Android `javax.crypto` e `java.security` .

Una volta generata una chiave, dobbiamo conservare questa chiave in una variabile o memorizzarla. Lo stiamo archiviando tra le preferenze condivise nel valore `S_KEY` . Quindi, una password viene crittografata utilizzando il metodo `doFinal` della classe `Cipher` una volta inizializzata in `ENCRYPT_MODE` . Successivamente, la password crittografata viene convertita da una matrice di byte in una stringa e archiviata tra le preferenze condivise. La chiave utilizzata per generare una password crittografata può essere utilizzata per decodificare la password in modo simile:

```
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;
    }
```

```
}
```

Leggi Come conservare le password in modo sicuro online:
https://riptutorial.com/it/android/topic/9093/come-conservare-le-password-in-modo-sicuro

Capitolo 50: Come usare SparseArray

introduzione

Un `SparseArray` è un'alternativa per una `Map` . Una `Map` richiede che le sue chiavi siano oggetti. Il fenomeno dell'autoboxing si verifica quando vogliamo usare un valore `int` primitivo come chiave. Il compilatore converte automaticamente i valori primitivi nei loro tipi di box (ad es. `int` in `Integer`). La differenza nel footprint della memoria è notevole: `int` usa 4 byte, `Integer` usa 16 byte. Un `SparseArray` utilizza `int` come valore chiave.

Osservazioni

Vantaggio:

- Meno utilizzo della memoria (a causa delle chiavi primitive).
- Nessun auto-box.

Svantaggio:

- SparseArray utilizza la ricerca binaria per trovare il valore (O (log n)), quindi potrebbe non essere la soluzione migliore se si deve lavorare con un numero elevato di elementi (usare HashMap).

Esistono diverse varianti della famiglia: -SparseArray <Integer, Object> -SparseBooleanArray <Integer, Boolean> -SparseIntArray <Integer, Integer> -SparseLongArray <Integer, Long> -LongSparseArray <Long, Object> -LongSparseLongArray <Long, Long >

Operazioni SparseArray

- aggiungendo element - put (int, x): aggiunge una mappatura dalla chiave specificata al valore specificato, sostituendo la mappatura precedente dalla chiave specificata se ce n'era una. - append (int, x): inserisce una coppia chiave / valore nell'array, ottimizzando il caso in cui la chiave è maggiore di tutte le chiavi esistenti nell'array. Dovresti usare append () in caso di chiavi sequenziali per ottimizzare le prestazioni. Altrimenti put () va bene.

- remove element - delete (int): rimuove il mapping dalla chiave specificata, se ce n'era. - removeAt (int): rimuove il mapping sull'indice specificato. - removeAtRange (int, int): rimuove un intervallo di mapping come batch.

- accessing element - get (int): Ottiene l'int mappato dalla chiave specificata, o 0 se non è stata effettuata alcuna mappatura. - get (int, E): ottiene l'int mappato dalla chiave specificata, o il valore specificato se non è stata effettuata alcuna mappatura. - valueAt (int): Dato un indice compreso nell'intervallo 0 ... size () - 1, restituisce il valore dal mapping di valori-chiave indexth che questo SparseIntArray memorizza. Gli indici sono ordinati in ordine crescente.

- ricerca indice / chiave - keyAt (int): Dato un indice compreso nell'intervallo 0 ... size () - 1, restituisce la chiave dal mapping del valore-chiave indexth memorizzato da SparseIntArray. Gli indici sono ordinati in ordine crescente. - valueAt (int): Dato un indice compreso nell'intervallo 0 ... size () - 1, restituisce il valore dal mapping di valori-chiave indexth che questo SparseIntArray memorizza. Gli indici sono ordinati in ordine crescente. - indexOfKey (int): restituisce l'indice per il quale keyAt (int) restituisce la chiave specificata o un numero negativo se la chiave specificata non è mappata. - indexOfValue (E): restituisce un indice per il quale valueAt (int) restituisce la chiave specificata o un numero negativo se nessuna chiave esegue la mappatura sul valore specificato. Fai attenzione che questa è una ricerca lineare, a differenza delle ricerche per chiave, e che più chiavi possono mappare allo stesso valore e questo ne troverà solo uno. La differenza nel loro impronta di memoria è evidente: l'int utilizza 4 byte, il numero intero utilizza 16 byte.SparseArray utilizza int come valore chiave.

Examples

Esempio di base con 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);
```

Esercitazione su YouTube

Leggi Come usare SparseArray online: https://riptutorial.com/it/android/topic/8824/come-usare-sparsearray

Capitolo 51: Componenti di architettura Android

introduzione

Android Architecture Components è una nuova raccolta di librerie che ti aiuta a progettare app robuste, testabili e manutenibili. Parti principali sono: Cicli di vita, ViewModel, LiveData, Room.

Examples

Aggiungi componenti di architettura

Progetto 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'
}
```

Gradle di applicazione

```
// 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"
```

Utilizzo del ciclo di vita in AppCompatActivity

Estendi la tua attività da questa attività

```
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 trasformazioni 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;
    }
}
```

Da qualche parte nell'interfaccia utente:

```
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);
        }
}
```

Peristence della stanza

Le room richiedono quattro parti: classe Database, classi DAO, classi entità e classi di migrazione (ora puoi utilizzare **solo i metodi DDL**):

Classi di entità

```
// 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;
}
```

Lezioni 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);
}
```

Classe di database

```
// register your entities and DAOs
@Database(entities = {VideoItem.class, VideoTag.class}, version = 2)
public abstract class ContentDatabase extends RoomDatabase {
    public abstract VideoDao videoDao();
}
```

migrazioni

```
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() {
    }
}
```

Utilizzare in classe Application o fornire tramite Dagger

```
ContentDatabase provideContentDatabase() {
    return Room.databaseBuilder(context, ContentDatabase.class, "data.db")
            .addMigrations(Migrations.ALL).build();
}
```

Scrivi il tuo repository:

```
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);
    }
}
```

Usa in ViewModel:

```
ContentRepository contentRepository = ...;
contentRepository.getVideoByTag(tag, limit);
```

LiveData personalizzato

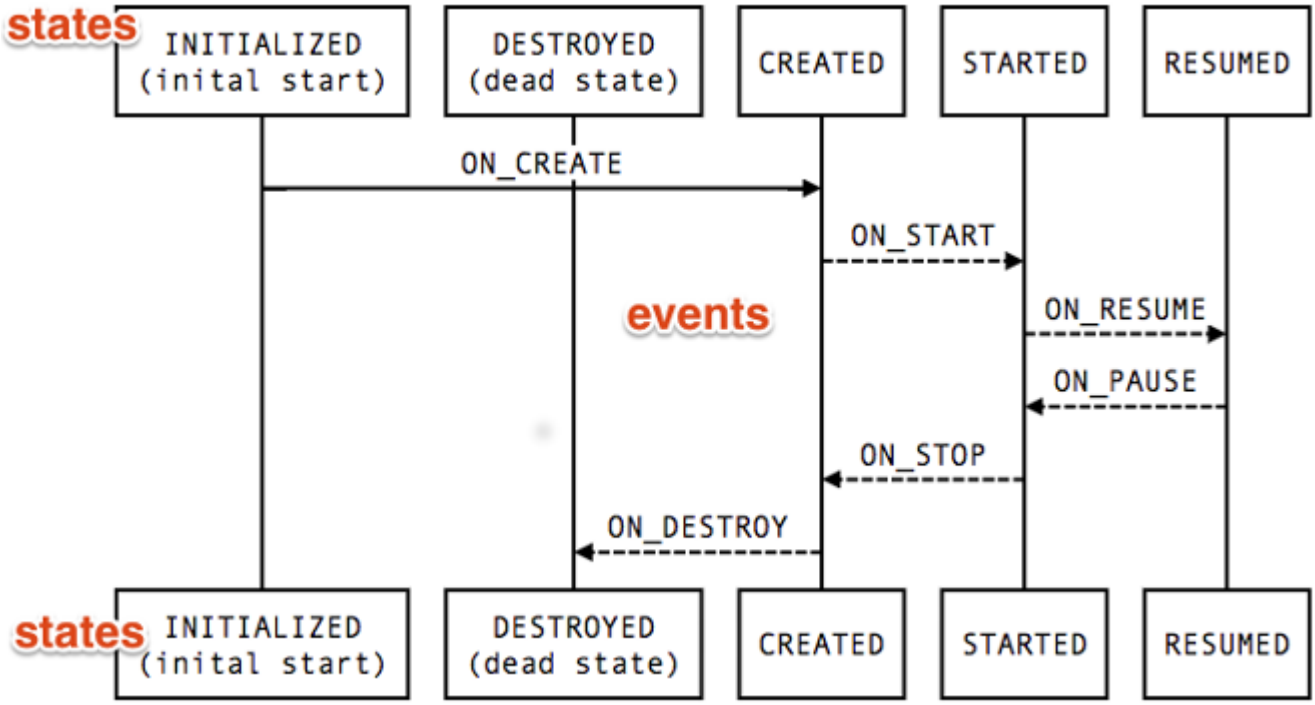
Puoi scrivere LiveData personalizzato, se hai bisogno di una logica personalizzata.
Non scrivere classi personalizzate, se hai solo bisogno di trasformare i dati (usa la classe Transformations)

```
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 che riconosce il ciclo di vita personalizzato

Ogni ciclo di vita dei componenti dell'interfaccia utente è cambiato come mostrato nell'immagine.



È possibile creare un componente, che verrà informato sulla modifica dello stato del ciclo di vita:

```
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
    }
}
```

Leggi Componenti di architettura Android online:
https://riptutorial.com/it/android/topic/10872/componenti-di-architettura-android

Capitolo 52: Compressione dell'immagine

Examples

Come comprimere l'immagine senza cambiare la dimensione

Ottieni **bitmap compressi** dalla classe 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;
    }
}
```

Le dimensioni sono le stesse dopo aver compresso Bitmap .

Come ho controllato?

```
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());
```

Produzione:

```
Before Compress : Dimension: 1080-1452
After Compress : Dimension: 1080-1452
```

Leggi Compressione dell'immagine online:
https://riptutorial.com/it/android/topic/5588/compressione-dell-immagine

Capitolo 53: Configurazione di Jenkins CI per progetti Android

Examples

Approccio graduale per configurare Jenkins per Android

Questa è una guida passo passo per configurare il processo di compilazione automatizzata utilizzando l'IC Jenkins per i tuoi progetti Android. I passi seguenti presumono che tu abbia un nuovo hardware con tutti i gusti di Linux installati. Si è anche tenuto conto del fatto che si potrebbe avere una macchina remota.

PARTE I: configurazione iniziale sulla macchina

1. Accedi tramite *ssh* alla tua macchina Ubuntu:

   ssh nomeutente@xxx.xxx.xxx

2. Scarica una versione di Android SDK sulla tua macchina:

   wget https://dl.google.com/android/android-sdk_r24.4.1-linux.tgz

3. Decomprimere il file *tar* scaricato:

   sudo apt-get install tar
   tar -xvf android-sdk_r24.4.1-linux.tgz

4. Ora è necessario installare Java 8 sulla tua macchina Ubuntu, che è un requisito per i build di Android su Nougat. Jenkins richiederebbe l'installazione di JDK e JRE 7 seguendo i passaggi seguenti:

   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. Ora installa Jenkins sulla tua macchina 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. Scarica l'ultima versione di Gradle supportata per la configurazione di Android:

   wget https://services.gradle.org/distributions/gradle-2.14.1-all.zip
   decomprimere gradle-2.14.1-all.zip

7. Configura Android sulla tua macchina Ubuntu. Passa innanzitutto alla cartella degli *strumenti* nella cartella SDK di Android scaricata nel passaggio 2:

   cd android-sdk-linux / tools ***// elenca l'SDK disponibile***
   android update sdk --no-ui ***// Aggiorna la versione dell'SDK***
   lista android sdk -a | grep "SDK Build-tools" ***// elenca gli strumenti di compilazione disponibili***
   android update sdk -a -u -t 4 ***// aggiorna la versione degli strumenti di build a uno elencato come 4 di prev. cmd.***
   aggiorna java

8. Installa *Git* o qualsiasi altro VCS sulla tua macchina:

   sudo apt-get install git

9. Ora accedi a Jenkins usando il tuo browser internet. Digita `ipAddress:8080` nella barra degli indirizzi.

10. Per ricevere la password per il primo accesso, si prega di controllare il file corrispondente come segue (sono necessarie le autorizzazioni su per accedere a questo file):

    cat / var / lib / jenkins / secrets / initialAdminPassword

PARTE II: Configura Jenkins per creare lavori Android

1. Una volta effettuato l'accesso, andare al seguente percorso:

   Jenkins> Gestisci Jenkins> Global Tool Configuration

2. In questa posizione, aggiungi `JAVA_HOME` con le seguenti voci:

   Nome = JAVA_HOME
   JAVA_HOME = / usr / lib / jvm / java-8-openjdk-amd64

3. Aggiungi anche i seguenti valori a *Git* e salva le variabili di ambiente:

   Nome = predefinito
   / Usr / bin / git

4. Ora vai al seguente percorso:

   Jenkins> Gestisci Jenkins> Configurazione

5. In questa posizione, aggiungi `ANDROID_HOME` alle "proprietà globali":

   Nome = ANDROID_HOME
   Valore = / home / nome utente / android-sdk-linux

Parte III: crea un lavoro Jenkins per il tuo progetto Android

1. Fai clic su *Nuovo elemento* nella schermata principale di Jenkins.
2. Aggiungi un *nome* e una *descrizione del progetto* .
3. Nella scheda *Generale* , selezionare *Avanzate* . Quindi selezionare *Usa area di lavoro personalizzata* :

   Directory / home / utente / codice / ProjectFolder
4. Nella gestione del codice sorgente seleziona *Git* . Sto usando *Bitbucket* ai fini di questo esempio:

   URL del repository = https: // nomeutente: password@bitbucket.org/project/projectname.git
5. Seleziona comportamenti aggiuntivi per il tuo repository:

   Pulire prima del checkout
   Checkout a una sottodirectory. Sottodirectory locale per repo / home / utente / codice / ProjectFolder
6. Seleziona un ramo che vuoi costruire:

   */maestro
7. Nella scheda *Costruisci* , seleziona *Execute Shell* in *Aggiungi fase di creazione* .
8. Nella *shell Execute* , aggiungere il seguente comando:

   cd / home / user / Code / ProjectFolder && gradle clean assembla --no-daemon
9. Se si desidera eseguire Lint sul progetto, quindi aggiungere un altro passo di build nella *shell Execute* :

   /home/user/gradle/gradle-2.14.1/bin/gradle lint

Ora il tuo sistema è finalmente pronto per costruire progetti Android usando Jenkins. Questa configurazione rende la vita molto più semplice per il rilascio di build ai team QA e UAT.

PS: Dato che Jenkins è un utente diverso sulla tua macchina Ubuntu, dovresti dargli i diritti per

creare cartelle nel tuo spazio di lavoro eseguendo il seguente comando:

```
chown -R jenkins .git
```

Leggi Configurazione di Jenkins CI per progetti Android online:
https://riptutorial.com/it/android/topic/7830/configurazione-di-jenkins-ci-per-progetti-android

Capitolo 54: Connessioni Wi-Fi

Examples

Connettiti con la crittografia WEP

Questo esempio si connette a un punto di accesso Wi-Fi con crittografia WEP, dato un SSID e la password.

```
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;
    }
}
```

Connettiti con la crittografia WPA2

Questo esempio si connette a un punto di accesso Wi-Fi con crittografia 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;
  }
}
```

Cerca i punti di accesso

Questo esempio ricerca gli access point disponibili e le reti ad hoc. `btnScan` attiva una scansione avviata dal metodo `WifiManager.startScan()` . Dopo la scansione, `WifiManager` chiama l'intento `SCAN_RESULTS_AVAILABLE_ACTION` e la classe `WifiScanReceiver` elabora il risultato della scansione. I risultati sono visualizzati in una `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");
            }
        }
    }
}
```

permessi

Le seguenti autorizzazioni devono essere definite in *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` è necessario per chiamare `WifiManager.getScanResults()` . Senza `android.permission.CHANGE_WIFI_STATE` non è possibile avviare una scansione con `WifiManager.startScan()` .

Durante la compilazione del progetto per api livello 23 o successivo (Android 6.0 e versioni successive), è necessario inserire `android.permission.ACCESS_FINE_LOCATION` o `android.permission.ACCESS_COARSE_LOCATION` . Inoltre, è necessario richiedere il permesso, ad

esempio nel metodo `onCreate` della propria attività principale:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    String[] PERMS_INITIAL={
            Manifest.permission.ACCESS_FINE_LOCATION,
    };
    ActivityCompat.requestPermissions(this, PERMS_INITIAL, 127);
}
```

Leggi Connessioni Wi-Fi online: https://riptutorial.com/it/android/topic/3288/connessioni-wi-fi

Capitolo 55: ConstraintLayout

introduzione

`ConstraintLayout` è un `ViewGroup` che consente di posizionare e dimensionare i widget in modo flessibile. È compatibile con Android 2.3 (livello API 9) e versioni successive.

Permette di creare layout grandi e complessi con una gerarchia di viste piatte. È simile a `RelativeLayout` in quanto tutte le visualizzazioni sono disposte in base alle relazioni tra le viste di pari livello e il layout principale, ma è più flessibile di `RelativeLayout` e più facile da utilizzare con l'Editor di layout di Android Studio.

Sintassi

- **ConstraintLayout**
  - public void addView (Visualizza child, int index, ViewGroup.LayoutParams params)
  - ConstraintLayout pubblico.LayoutParams generaLayoutParams (AttributeSet attrs)
  - public void onViewAdded (Visualizza vista)
  - public void onViewRemoved (Visualizza visualizzazione)
  - public void removeView (Visualizza visualizzazione)
  - public void requestLayout ()
  - protetto booleano checkLayoutParams (Parametri ViewGroup.LayoutParams)
  - protetto ConstraintLayout.LayoutParams generateDefaultLayoutParams ()
  - ViewGroup.LayoutParams protetti generateLayoutParams (Parametri ViewGroup.LayoutParams)
  - protected void onLayout (booleano modificato, int left, int top, int right, int bottom)
  - protected void onMeasure (int widthMeasureSpec, int heightMeasureSpec)
- **ConstraintLayout.LayoutParams**
  - public void resolveLayoutDirection (int layoutDirection)
  - public void validate ()
  - protected void setBaseAttributes (TypedArray a, int widthAttr, int heightAttr)

Parametri

Parametro	Dettagli
bambino	La `View` da aggiungere al layout
indice	L'indice della `View` nella gerarchia del layout
params	Il `LayoutParams` della `View`
attrs	L' `AttributeSet` che definisce il `LayoutParams`
vista	La `View` che è stata aggiunta o rimossa
cambiato	Indica se questa `View` ha cambiato dimensione o posizione
sinistra	La posizione sinistra, relativa alla `View` genitore
superiore	La posizione superiore, relativa alla `View` genitore
destra	La posizione corretta, relativa alla `View` genitore
parte inferiore	La posizione in basso, relativa alla `View` genitore
widthMeasureSpec	I requisiti di spazio orizzontale imposti dalla `View` principale
heightMeasureSpec	I requisiti di spazio verticale imposti dalla `View` padre
layoutDirection	-
un	-
widthAttr	-
heightAttr	-

Osservazioni

A Google IO 2016 Google ha annunciato un nuovo layout Android denominato ConstraintLayout. Fai attenzione perché al momento questo layout è una **versione beta** .

Più informazioni sul layout dei vincoli:

https://codelabs.developers.google.com/codelabs/constraint-layout/index.html

Examples

Aggiunta di ConstraintLayout al progetto

Per lavorare con ConstraintLayout, è necessario Android Studio versione 2.2 o successiva e avere

almeno la versione 32 (o successiva) del repository di supporto Android.

1. Aggiungi la libreria Constraint Layout come dipendenza nel tuo file `build.gradle` :

```
dependencies {
   compile 'com.android.support.constraint:constraint-layout:1.0.2'
}
```

2. Sincronizza progetto

Per aggiungere un nuovo layout di vincoli al tuo progetto:

1. **Fare clic con** il **tasto destro** sulla directory di layout del modulo, quindi fare clic su `New > XML > Layout XML.`
2. Inserisci un **nome** per il layout e inserisci `"android.support.constraint.ConstraintLayout"` per il tag di root.
3. Fai clic su **Fine** .

Altrimenti aggiungi semplicemente un file di layout:

```
<?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>
```

Catene

Poiché `ConstraintLayout` alpha 9, le **catene** sono disponibili. Una **Catena** è un insieme di viste all'interno di un `ConstraintLayout` che sono collegate in modo bidirezionale tra loro, cioè **A** collegato a **B** con un vincolo e **B** collegato ad **A** con un altro vincolo.

Esempio:

```
<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>
```

In questo esempio, le due viste sono posizionate una sotto l'altra ed entrambe sono centrate verticalmente. È possibile modificare la posizione verticale di queste viste regolando il **bias** della catena. Aggiungi il seguente codice al primo elemento di una catena:

```
app:layout_constraintVertical_bias="0.2"
```

In una catena verticale, il primo elemento è la vista più in alto, e in una catena orizzontale è la vista più a sinistra. Il primo elemento definisce il comportamento dell'intera catena.

Le catene sono una nuova funzionalità e vengono aggiornate frequentemente. Ecco una documentazione ufficiale Android su Chains.

Leggi ConstraintLayout online: https://riptutorial.com/it/android/topic/5076/constraintlayout

Capitolo 56: ConstraintSet

introduzione

Questa classe consente di definire a livello di programmazione una serie di vincoli da utilizzare con `ConstraintLayout` . Ti consente di creare e salvare i vincoli e applicarli a un `ConstraintLayout` esistente.

Examples

ConstraintSet with ContraintLayout Programmatically

```
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
        }
    }
}
```

Leggi ConstraintSet online: https://riptutorial.com/it/android/topic/9334/constraintset

Capitolo 57: Contesto

introduzione

Secondo la documentazione di Google: "Interfaccia con le informazioni globali su un ambiente applicativo, che consente l'accesso a risorse e classi specifiche dell'applicazione, nonché up-call per operazioni a livello di applicazione come attività di lancio, trasmissione e ricezione di intenti, ecc."

Più semplicemente, Context è lo stato attuale della tua applicazione. Ti consente di fornire informazioni agli oggetti in modo che possano essere a conoscenza di ciò che accade in altre parti della tua applicazione.

Sintassi

- `getApplicationContext()`
- `getBaseContext()`
- `getContext()`
- `this`

Osservazioni

Questa pagina di StackOverflow ha diverse spiegazioni complete e ben scritte del concetto di contesto:

Cos'è il contesto?

Examples

Esempi di base

Utilizzo standard nell'attività:

```
Context context = getApplicationContext();
```

Utilizzo standard in Frammento:

```
Context context = getActivity().getApplicationContext();
```

`this` (quando in una classe che si estende da Context, come le classi Application, Activity, Service e IntentService)

```
TextView textView = new TextView(this);
```

altro `this` esempio:

```
Intent intent = new Intent(this, MainActivity.class);
startActivity(intent);
```

Leggi Contesto online: https://riptutorial.com/it/android/topic/9774/contesto

Capitolo 58: Conto alla rovescia

Parametri

Parametro	Dettagli
`long millisInFuture`	La durata totale per cui verrà eseguito il timer, ovvero fino a che punto in futuro si desidera che il timer termini. In millisecondi.
`long countDownInterval`	L'intervallo al quale si desidera ricevere gli aggiornamenti del timer. In millisecondi.
`long millisUntilFinished`	Un parametro fornito in `onTick()` che indica per quanto tempo il CountDownTimer è rimasto. In millisecondi

Osservazioni

CountDownTimer è una classe piuttosto magra - fa una cosa molto bene. Poiché è possibile avviare / annullare solo un CountDownTimer, è necessario implementare la funzionalità di pausa / ripresa come mostrato nel secondo esempio. Per funzionalità più complesse o per specificare un timer che deve essere eseguito indefinitamente, utilizzare l'oggetto Timer .

Examples

Creazione di un semplice timer per il conto alla rovescia

CountDownTimer è utile per eseguire ripetutamente un'azione in un intervallo costante per una durata impostata. In questo esempio, aggiorneremo una visualizzazione testuale ogni secondo per 30 secondi indicando quanto tempo rimane. Quindi, quando il timer termina, imposteremo TextView per dire "Fatto".

```
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 esempio più complesso

In questo esempio, metteremo in pausa / riprendere il CountDownTimer basato sul ciclo di vita dell'Attività.

```
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;
}
```

Leggi Conto alla rovescia online: https://riptutorial.com/it/android/topic/6063/conto-alla-rovescia

Capitolo 59: Controlla la connessione dati

Examples

Controlla la connessione dati

Questo metodo consente di verificare la connessione dati eseguendo il ping di determinati IP o nome di 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;
}
```

Controllare la connessione usando ConnectivityManager

```
    public static boolean isConnectedNetwork (Context context) {

        ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
        return cm.getActiveNetworkInfo () != null && cm.getActiveNetworkInfo
().isConnectedOrConnecting ();

    }
```

Utilizza gli intenti di rete per eseguire attività mentre i dati sono consentiti

Quando il dispositivo si connette a una rete, viene inviato un tentativo. Molte app non controllano questi intenti, ma per far funzionare correttamente la tua applicazione, puoi ascoltare gli intenti di modifica della rete che ti diranno quando è possibile la comunicazione. Per verificare la connettività di rete è possibile, ad esempio, utilizzare la seguente clausola:

```
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
}
```

Leggi Controlla la connessione dati online: https://riptutorial.com/it/android/topic/8670/controlla-la-connessione-dati

Capitolo 60: Conversazione di testo in testo

Examples

Discorso su testo con finestra di dialogo predefinita di Google Prompt

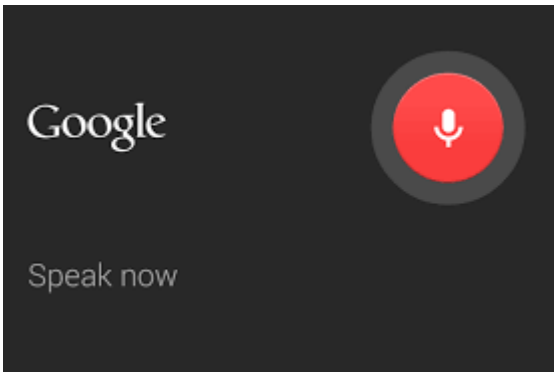
Attiva il discorso alla traduzione del testo

```
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();
    }
}
```

Ottieni risultati tradotti inAttivitàRisultato

```
@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;
    }

    }
}
```

Produzione



Discorso al testo senza finestra di dialogo

Il seguente codice può essere utilizzato per attivare la conversione da testo a testo senza visualizzare una finestra di dialogo:

```
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 classe listener personalizzata `CustomRecognitionListener` utilizzata nel codice sopra riportato è implementata come segue:

```
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);
    }
}
```

Leggi Conversazione di testo in testo online:
https://riptutorial.com/it/android/topic/6252/conversazione-di-testo-in-testo

Capitolo 61: Converti la stringa vietnamita in una stringa inglese Android

Examples

esempio:

```
String myStr = convert("Lê Minh Thoại là người Việt Nam");
```

convertito:

```
"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;
}
```

Leggi Converti la stringa vietnamita in una stringa inglese Android online: https://riptutorial.com/it/android/topic/10946/converti-la-stringa-vietnamita-in-una-stringa-inglese-android

Capitolo 62: CoordinatorLayout and Behaviors

introduzione

Il CoordinatorLayout è un FrameLayout superpotente e l'obiettivo di questo ViewGroup è di coordinare le viste al suo interno.

L'attrattiva principale di CoordinatorLayout è la sua capacità di coordinare le animazioni e le transizioni delle viste all'interno del file XML stesso.

CoordinatorLayout è destinato a due casi di utilizzo principale:

: Come decorazione per applicazioni di alto livello o layout cromati

: Come contenitore per una specifica interazione con una o più viste secondarie

Osservazioni

Il `CoordinatorLayout` è un contenitore che estende il `FrameLayout` .
Associando un `CoordinatorLayout.Behavior` a un figlio diretto di `CoordinatorLayout` , sarete in grado di intercettare gli eventi tattili, le finestre, le misure, il layout e lo scorrimento annidato.

Specificando `Behaviors` per le visualizzazioni secondarie di un `CoordinatorLayout` è possibile fornire molte interazioni diverse all'interno di un singolo genitore e tali visualizzazioni possono anche interagire tra loro. Le classi di vista possono specificare un comportamento predefinito quando vengono utilizzate come figlio di un `CoordinatorLayout` utilizzando l'annotazione `DefaultBehavior` .

Examples

Creare un semplice comportamento

Per creare un `Behavior` basta estendere la classe `CoordinatorLayout.Behavior` .

Estendi il CoordinatorLayout.Behavior

Esempio:

```
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);
    }
}
```

Questo comportamento deve essere collegato a un figlio Visualizzazione di un `CoordinatorLayout` da chiamare.

Allegare un comportamento a livello di codice

```
MyBehavior myBehavior = new MyBehavior();
CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams)
view.getLayoutParams();
params.setBehavior(myBehavior);
```

Allegare un comportamento in XML

Puoi utilizzare l'attributo `layout_behavior` per collegare il comportamento in XML:

```
<View
  android:layout_height="...."
  android:layout_width="...."
  app:layout_behavior=".MyBehavior" />
```

Allegare un comportamento automaticamente

Se si sta lavorando con una vista personalizzata, è possibile allegare il comportamento utilizzando l'annotazione `@CoordinatorLayout.DefaultBehavior` :

```
@CoordinatorLayout.DefaultBehavior(MyBehavior.class)
public class MyView extends ..... {

}
```

Utilizzando il SwipeDismissBehavior

`SwipeDismissBehavior` funziona su qualsiasi vista e implementa la funzionalità di scorrimento per

ignorare nei nostri layout con un `CoordinatorLayout` .

Basta usare:

```
        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);
```

Crea dipendenze tra le viste

Puoi utilizzare `CoordinatorLayout.Behavior` per creare dipendenze tra le viste. Puoi ancorare una `View` a un'altra `View` di:

- usando l'attributo `layout_anchor` .
- creando un `Behavior` personalizzato e implementando il metodo `layoutDependsOn` che restituisce `true` .

Ad esempio, per creare un `Behavior` per lo spostamento di un `ImageView` quando ne viene spostato un altro (barra degli strumenti di esempio), effettuare le seguenti operazioni:

- Crea il comportamento personalizzato :

  ```
  public class MyBehavior extends CoordinatorLayout.Behavior<ImageView> {...}
  ```

- Sovrascrivi il metodo `layoutDependsOn` che restituisce `true` . Questo metodo viene chiamato ogni volta che si verifica una modifica al layout:

  ```
  @Override
  public boolean layoutDependsOn(CoordinatorLayout parent,
          ImageView child, View dependency) {
      // Returns true to add a dependency.
      return dependency instanceof Toolbar;
  }
  ```

- Ogni volta che il metodo `layoutDependsOn` restituisce `true` , viene chiamato il metodo `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);
}
```

Leggi CoordinatorLayout and Behaviors online:
https://riptutorial.com/it/android/topic/5714/coordinatorlayout-and-behaviors

Capitolo 63: corsia di sorpasso

Osservazioni

fastlane è uno strumento per gli sviluppatori iOS, Mac e Android per automatizzare attività noiose come la generazione di schermate, la gestione dei profili di provisioning e il rilascio della tua applicazione.

Documenti: https://docs.fastlane.tools/

Codice sorgente: https://github.com/fastlane/fastlane

Examples

Fastfile per creare e caricare più versioni su Beta da Crashlytics

Questa è una configurazione di **Fastfile di** esempio per **un'app** multi-flavor. Ti dà la possibilità di costruire e distribuire tutti i sapori o un singolo sapore. Dopo la distribuzione, riporta a **Slack** lo stato della distribuzione e invia una notifica ai tester in Beta dal gruppo di tester Crashlytics.

Per costruire e distribuire tutti i sapori utilizzare:

```
fastlane android beta
```

Per creare un singolo APK e distribuire l'uso:

```
fastlane android beta app:flavorName
```

Utilizzando un singolo file Fastlane, è possibile gestire app iOS, Android e Mac. Se si utilizza questo file solo per una `platform` app non è richiesto.

Come funziona

1. argomento `android` dice fastlane che useremo `:android` piattaforma `:android` .
2. All'interno `:android` piattaforma `:android` è possibile avere più corsie. Attualmente, ho solo `:beta` lane. Il secondo argomento del comando precedente specifica la corsia che vogliamo usare.
3. `options[:app]`
4. Ci sono due compiti **Gradle** . In primo luogo, funziona `gradle clean` . Se hai fornito un sapore con la chiave `app` , fastfile esegue `gradle assembleReleaseFlavor` . Altrimenti, esegue `gradle assembleRelease` per costruire tutti i sapori di build.
5. Se stiamo costruendo per tutti i gusti, una serie di nomi di file APK generati viene archiviata in `SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS` . Lo usiamo per fare il loop dei file generati e distribuirli su **Beta da Crashlytics** . `notifications` e i campi dei `groups` sono facoltativi. Sono utilizzati per informare i tester registrati per l'app su **Beta da Crashlytics** .

6. Se hai familiarità con Crashlytics, potresti sapere che per attivare un'app nel portale, devi eseguirla su un dispositivo e usarla prima. Altrimenti, Crashlytics assumerà l'app inattiva e genererà un errore. In questo scenario, l'ho catturato e segnalato a **Slack** come un errore, quindi saprai quale app non è attiva.
7. Se la distribuzione ha successo, **fastlane** invierà un messaggio di successo a **Slack** .
8. `#{/([^\/]*)$/.match(apk)}` questo regex è usato per ottenere il nome di sapore dal percorso APK. Puoi rimuoverlo se non funziona per te.
9. `get_version_name` e `get_version_code` sono due plugin di **Fastlane** per recuperare nome e codice della versione dell'app. Devi installare queste gemme se vuoi usarle o puoi rimuoverle. Leggi di più sui plugin qui.
10. La dichiarazione `else` verrà eseguita se si sta creando e distribuendo un singolo APK. Non dobbiamo fornire `apk_path` a Crashlytics poiché abbiamo una sola app.
11. `error do` blocco di `error do` alla fine viene usato per ricevere una notifica se qualcosa va storto durante l'esecuzione.

Nota

Non dimenticare di sostituire `SLACK_URL` , `API_TOKEN` , `GROUP_NAME` e `BUILD_SECRET` con le tue credenziali.

```
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 per costruire e installare tutti gli aromi per un determinato tipo di

build su un dispositivo

Aggiungi questa corsia al tuo **Fastfile** ed esegui `fastlane installAll type:{BUILD_TYPE}` nella riga di comando. Sostituisci `BUILD_TYPE` con il tipo di build che vuoi costruire.

Ad esempio: `fastlane installAll type:Debug`

Questo comando creerà tutte le caratteristiche del tipo specificato e lo installerà sul tuo dispositivo. Attualmente, non funziona se hai più di un dispositivo collegato. Assicurati di averne solo uno. In futuro ho intenzione di aggiungere un'opzione per selezionare il dispositivo di destinazione.

```
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
```

Leggi corsia di sorpasso online: https://riptutorial.com/it/android/topic/8215/corsia-di-sorpasso

Capitolo 64: Cos'è ProGuard? Cosa si usa in Android?

introduzione

Proguard è un programma per la riduzione di file Java, ottimizzatore, offuscatore e preverificatore gratuito. Rileva e rimuove classi, campi, metodi e attributi non utilizzati. Ottimizza bytecode e rimuove le istruzioni inutilizzate. Rinomina le restanti classi, campi e metodi usando nomi brevi e privi di significato.

Examples

Riduci il codice e le risorse con proguard

Per rendere il file APK il più piccolo possibile, è necessario abilitare la riduzione per rimuovere il codice e le risorse inutilizzati nella build di rilascio. Questa pagina descrive come farlo e come specificare quale codice e quali risorse tenere o scartare durante la compilazione.

Il restringimento del codice è disponibile con ProGuard, che rileva e rimuove classi, campi, metodi e attributi non utilizzati dall'app pacchettizzata, inclusi quelli delle librerie di codici incluse (rendendolo un valido strumento per aggirare il limite di riferimento a 64k). ProGuard ottimizza anche il bytecode, rimuove le istruzioni inutilizzate del codice e nasconde le restanti classi, campi e metodi con nomi brevi. Il codice offuscato rende il tuo APK difficile da decodificare, il che è particolarmente utile quando l'app utilizza funzionalità sensibili alla sicurezza, come la verifica della licenza.

Il restringimento delle risorse è disponibile con il plug-in Android per Gradle, che rimuove le risorse inutilizzate dall'app pacchettizzata, incluse le risorse inutilizzate nelle librerie di codice. Funziona in combinazione con la riduzione del codice in modo tale che una volta rimosso il codice non utilizzato, tutte le risorse non più referenziate possono essere rimosse in sicurezza.

Riduci il tuo codice

Per abilitare la riduzione del codice con `ProGuard` , aggiungi `minifyEnabled` true al tipo di build appropriato nel tuo file `build.gradle` .

Tieni presente che la riduzione del codice rallenta i tempi di costruzione, quindi dovresti evitare di usarla nella tua build di debug se possibile. Tuttavia, è importante abilitare la riduzione del codice sull'APK finale utilizzato per il test, poiché potrebbe introdurre dei bug se non si personalizza sufficientemente il codice da conservare.

Ad esempio, il seguente frammento di un file `build.gradle` consente di ridurre il codice per la build di rilascio:

```
android {
```

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

Oltre alla proprietà `minifyEnabled` , la proprietà `proguardFiles` definisce le `ProGuard rules` :

Il metodo getDefaultProguardFile ('proguard-android.txt') ottiene le impostazioni predefinite di ProGuard dagli `tools/proguard/ folder` Android SDK `tools/proguard/ folder` . Suggerimento: per ridurre ancora di più il codice, prova il file `proguard-android-optimize.txt` nella stessa posizione. Include le stesse regole di ProGuard, ma con altre ottimizzazioni che eseguono l'analisi a livello di bytecode, all'interno e tra diversi metodi, per ridurre ulteriormente le dimensioni dell'APK e accelerare l'esecuzione. Il file `proguard-rules.pro` è dove puoi aggiungere regole ProGuard personalizzate. Per impostazione predefinita, questo file si trova nella radice del modulo (accanto al file build.gradle). Per aggiungere più regole ProGuard che sono specifici per ciascuna variante costruttiva, aggiungere un'altra proprietà proguardFiles nel corrispondente `productFlavor` blocco. Ad esempio, il seguente file Gradle aggiunge flavor2-rules.pro al sapore flavor2 del prodotto. Ora flavor2 usa tutte e tre le regole ProGuard perché vengono applicate anche quelle del blocco di rilascio.

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

Leggi Cos'è ProGuard? Cosa si usa in Android? online:
https://riptutorial.com/it/android/topic/9205/cos-e-proguard--cosa-si-usa-in-android-

Capitolo 65: Cose Android

Examples

Controllo di un servomotore

Questo esempio presuppone che tu abbia un servo con le seguenti caratteristiche, che sono tipiche:

- movimento tra 0 e 180 gradi
- periodo di impulso di 20 ms
- durata minima dell'impulso di 0,5 ms
- durata massima dell'impulso di 2,5 ms

È necessario verificare se tali valori corrispondono al proprio hardware, poiché forzarlo ad andare al di fuori dell'intervallo operativo specificato può danneggiare il servo. Un servo danneggiato a sua volta ha il potenziale per danneggiare il dispositivo Android Things. La classe `ServoController` esempio è composta da due metodi, `setup()` e `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();
```

```
        }
    }
}
```

Puoi scoprire nomi di piedini che supportano PWM sul tuo dispositivo come segue:

```
PeripheralManagerService service = new PeripheralManagerService();

for (String pinName : service.getPwmList() ) {
    Log.i("ServoControlled","Pwm pin found: " + pinName);
}
```

Per rendere il tuo servo oscillante per sempre tra 80 gradi e 100 gradi, puoi semplicemente utilizzare il seguente codice:

```
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();
```

È possibile compilare e distribuire tutto il codice sopra senza effettivamente agganciare alcun servomotore al dispositivo di calcolo. Per il cablaggio, fare riferimento alla tabella di pinout del dispositivo di calcolo (ad esempio un grafico di pinout Raspberry Pi 3 è disponibile qui).

Quindi devi agganciare il tuo servo a Vcc, Gnd e segnale.

Leggi Cose Android online: https://riptutorial.com/it/android/topic/8938/cose-android

Capitolo 66: Costruire le app compatibili con le versioni precedenti

Examples

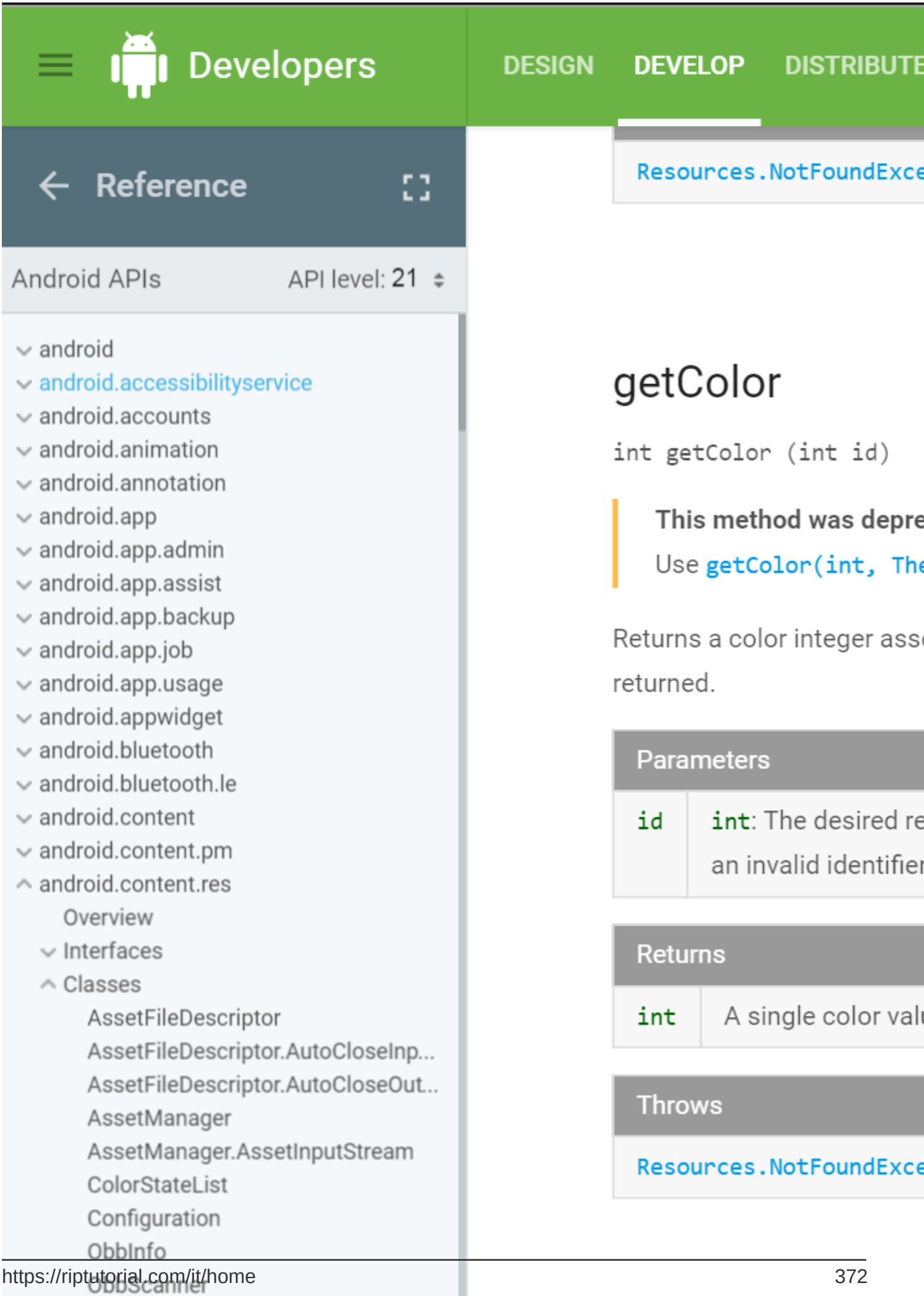
Come gestire l'API deprecata

È improbabile che uno sviluppatore non incontri un'API obsoleta durante un processo di sviluppo. Un elemento del programma deprecato è uno che i programmatori sono scoraggiati dall'usare, in genere perché è pericoloso o perché esiste un'alternativa migliore. I compilatori e gli analizzatori (come LINT) avvisano quando un elemento del programma deprecato viene utilizzato o sovrascritto nel codice non deprecato.

Un'API obsoleta viene solitamente identificata in Android Studio mediante un strikeout. Nell'esempio seguente, il metodo `.getColor(int id)` è deprecato:

```
getResources().~~getColor~~(R.color.colorAccent));
```

Se possibile, gli sviluppatori sono incoraggiati a utilizzare API ed elementi alternativi. È possibile verificare la compatibilità all'indietro di una libreria visitando la documentazione di Android per la libreria e controllando la sezione "Aggiunta in livello 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
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/it/android/topic/4291/costruire-le-app-compatibili-con-le-versioni-precedenti

Capitolo 67: Crash Reporting Tools

Osservazioni

Il wiki completo migliore è disponibile qui in github .

Examples

Tessuto - Crashlytics

Fabric è una piattaforma mobile modulare che fornisce kit utili che puoi mixare per creare la tua applicazione. **Crashlytics** è uno strumento di segnalazione di arresti **anomali** fornito da Fabric che consente di monitorare e monitorare le applicazioni in dettaglio.

Come configurare Fabric-Crashlytics

Passaggio 1: modifica il tuo `build.gradle` :

Aggiungi il repository plugin e il 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.+'
  }
}
```

Applica il plugin:

```
apply plugin: 'com.android.application'
//Put Fabric plugin after Android plugin
apply plugin: 'io.fabric'
```

Aggiungi il repository Fabric:

```
repositories {
  maven { url 'https://maven.fabric.io/public' }
}
```

Aggiungi il kit Crashlyrics:

```
dependencies {

  compile('com.crashlytics.sdk.android:crashlytics:2.6.6@aar') {
    transitive = true;
  }
}
```

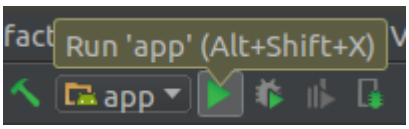
Passaggio 2: aggiungere la **chiave API** e l'autorizzazione **INTERNET** in `AndroidManifest.xml`

```
<?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>
```

Passaggio 3: avviare il kit in fase di esecuzione nel codice utente, ad esempio:

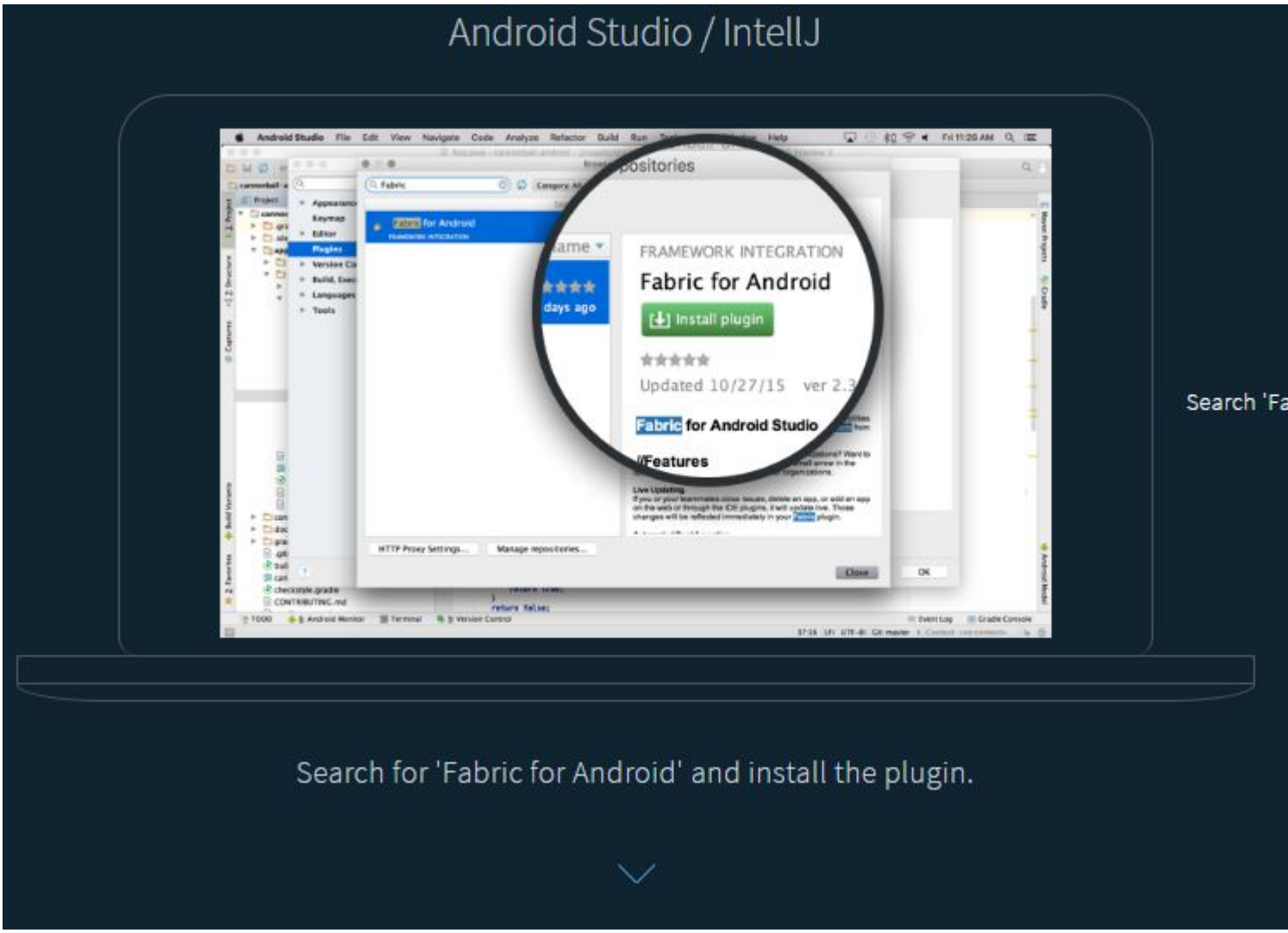
```
public class MainActivity extends ActionBarActivity {

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

      //Init the KIT
      Fabric.with(this, new Crashlytics());

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

Passaggio 4: crea un progetto. Per costruire ed eseguire:



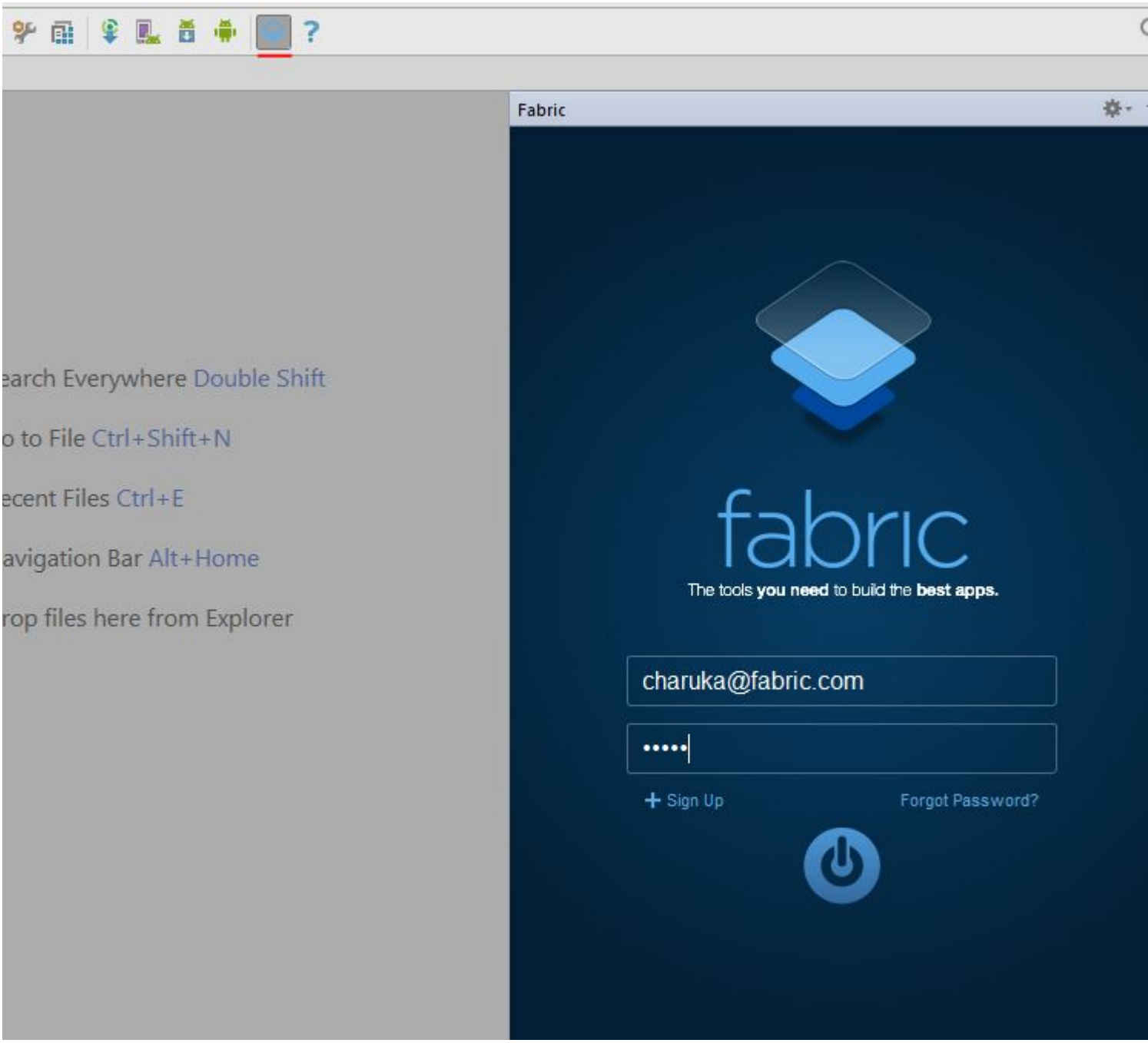
Utilizzando il plug-in IDE Fabric

I kit possono essere installati utilizzando il plug-in Fabric IDE per Android Studio o IntelliJ seguendo questo link.



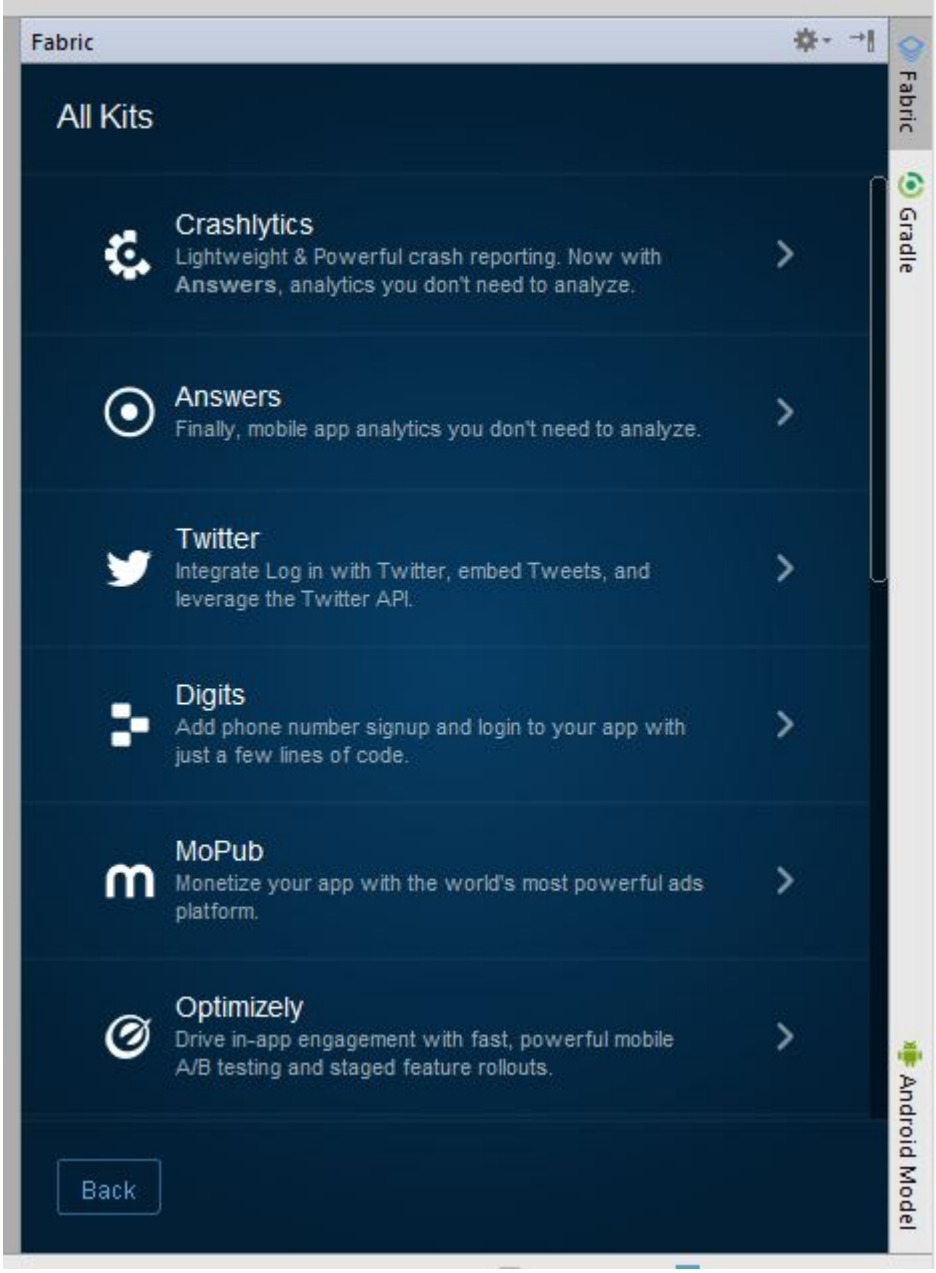
Dopo aver installato il plug-in, **riavvia** Android Studio e **accedi** con il tuo account utilizzando **Android Studio** .

(tasto `CTRL + L` > `CTRL + L`)

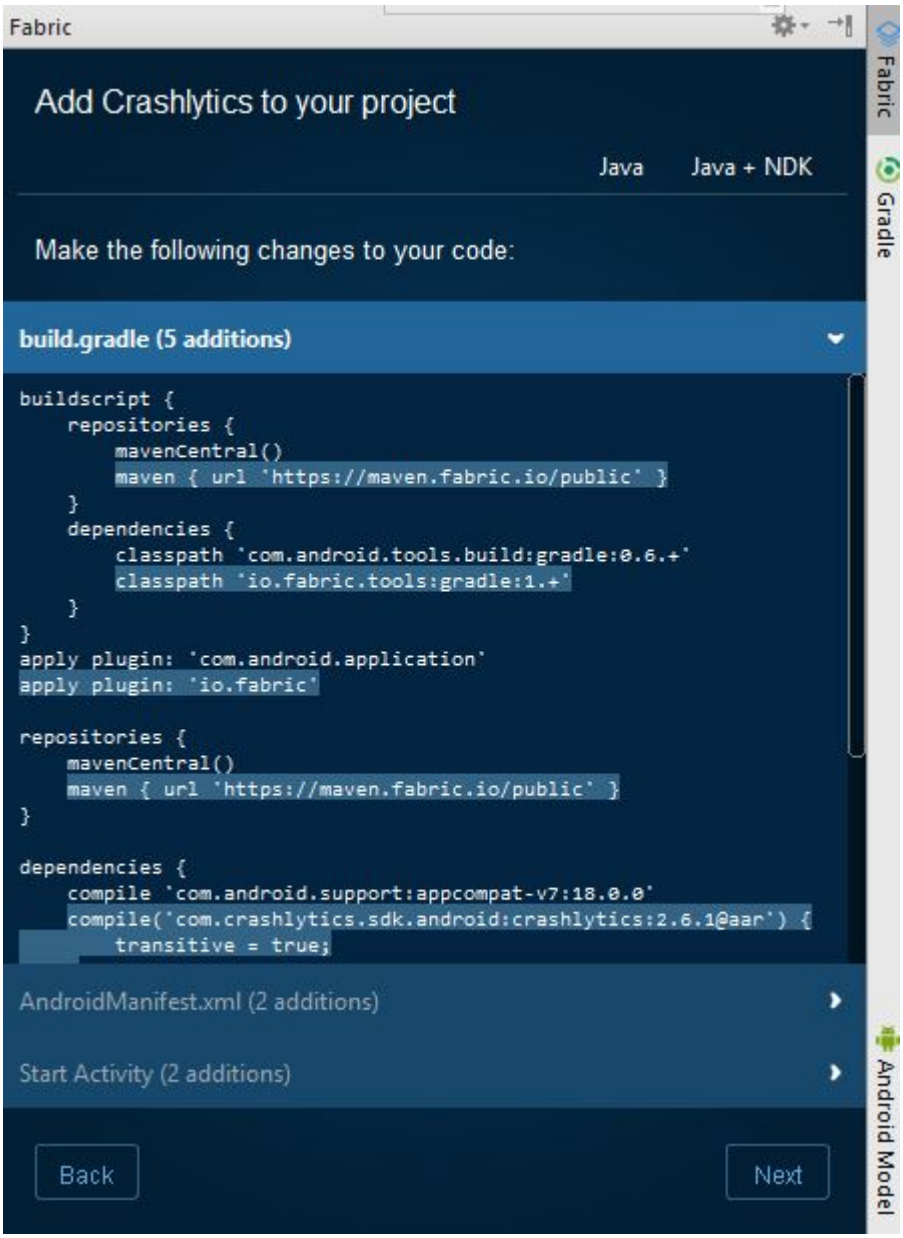


Quindi mostrerà i progetti che hai / il progetto che hai aperto, seleziona quello di cui hai bisogno e fai clic su Avanti ... successivo.

Seleziona il kit che vorresti aggiungere, per il suo esempio è **Crashlytics** :



Quindi premi `Install` . Non è necessario aggiungerlo manualmente questa volta come sopra il **plugin gradle** , invece verrà creato per te.



Fatto!

Segnalazione di crash con ACRA

Passaggio 1: aggiungere la dipendenza dell'ultimo ACRA AAR al gradle dell'applicazione (build.gradle).

Passaggio 2: nella classe dell'applicazione (la classe che estende l'applicazione, se non la crea) aggiungere un'annotazione `@ReportsCrashes` e sovrascrivere il metodo `attachBaseContext()` .

Passaggio 3: inizializzare la classe ACRA nella classe dell'applicazione

```
@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);
            }
    }
```

Dove AUTH_USERNAME e AUTH_PASSWORD sono le credenziali dei tuoi backend desiderati.

Passaggio 4: definire la classe dell'applicazione in AndroidManifest.xml

```
 <application
        android:name=".MyApplication">
        <service></service>
        <activity></activity>
        <receiver></receiver>
 </application>
```

Passaggio 5: assicurarsi di disporre dell'autorizzazione `internet` per ricevere il rapporto dall'applicazione arrestata

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

Nel caso in cui si desideri inviare il rapporto silenzioso al back-end, utilizzare il seguente metodo per ottenerlo.

```
ACRA.getErrorReporter().handleSilentException(e);
```

Forza un crash di prova con tessuto

Aggiungi un pulsante che puoi toccare per attivare un arresto anomalo. Incolla questo codice nel layout in cui desideri che venga visualizzato il pulsante.

```
<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" />
```

Lancia una RuntimeException

```
public void forceCrash(View view) {
    throw new RuntimeException("This is a crash");
}
```

Esegui la tua app e tocca il nuovo pulsante per causare un arresto anomalo. In un minuto o due dovresti essere in grado di vedere l'incidente sul cruscotto Crashlytics così come riceverai una mail.

Cattura gli arresti anomali con Sherlock

Sherlock cattura tutti i tuoi arresti anomali e li segnala come notifica. Quando si tocca la notifica, si apre un'attività con tutti i dettagli del crash insieme alle informazioni su dispositivo e applicazione

Come integrare Sherlock con la tua applicazione?

Devi solo aggiungere Sherlock come dipendenza gradle nel tuo progetto.

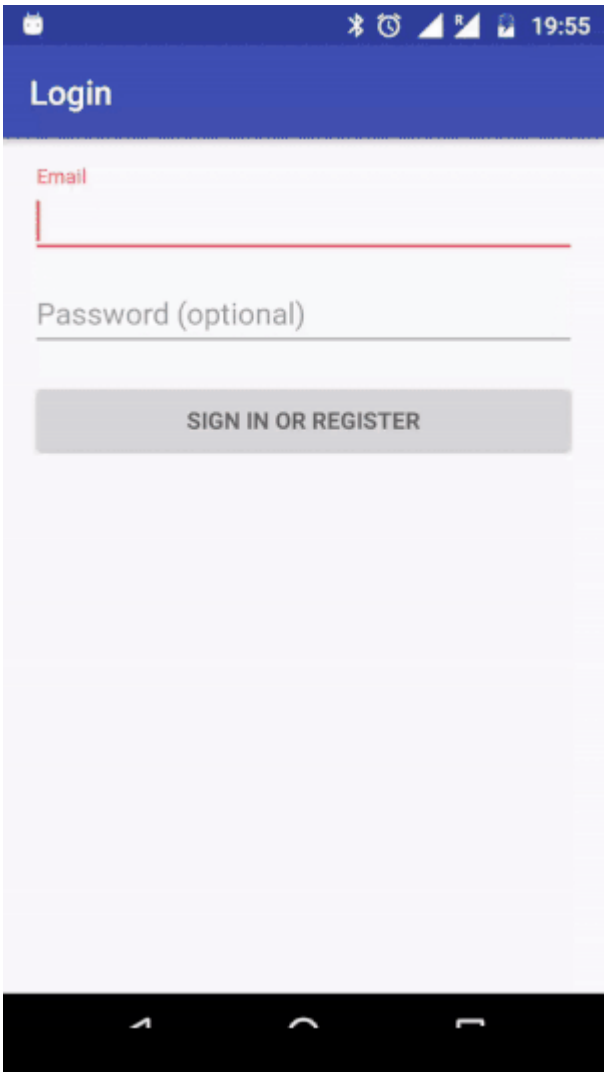
```
dependencies {
    compile('com.github.ajitsing:sherlock:1.0.1@aar') {
        transitive = true
    }
}
```

Dopo aver sincronizzato il tuo studio Android, inizializza Sherlock nella tua classe Application.

```
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);
  }
}
```

Questo è tutto ciò che devi fare. Anche Sherlock fa molto di più che segnalare un incidente. Per controllare tutte le sue caratteristiche date un'occhiata a questo articolo .

dimostrazione



Leggi Crash Reporting Tools online: https://riptutorial.com/it/android/topic/3871/crash-reporting-tools

Capitolo 68: Crea ROM personalizzate per Android

Examples

Preparare la macchina per l'edilizia!

Prima di poter costruire qualsiasi cosa, è necessario rendere la macchina pronta per la costruzione. Per questo è necessario installare un sacco di librerie e moduli. La distribuzione Linux più raccomandata è Ubuntu, quindi questo esempio si concentrerà sull'installazione di tutto ciò che è necessario su Ubuntu.

Installazione di Java

Innanzitutto, aggiungi il seguente archivio di pacchetti personali (PPA): `sudo apt-add-repository ppa:openjdk-r/ppa` .

Quindi, aggiorna i sorgenti eseguendo: `sudo apt-get update` .

Installazione di dipendenze aggiuntive

Tutte le dipendenze aggiuntive richieste possono essere installate con il seguente 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
```

Preparare il sistema per lo sviluppo

Ora che tutte le dipendenze sono installate, prepariamo il sistema per lo sviluppo eseguendo:

```
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
```

Infine, impostiamo la cache e il repository tramite i seguenti comandi:

```
sudo install utils/repo /usr/bin/
sudo install utils/ccache /usr/bin/
```

Nota: possiamo anche ottenere questa configurazione eseguendo gli script automatici creati da Akhil Narang (*akhilnarang*), uno dei manutentori del Resurrection Remix OS . Questi script possono essere trovati su GitHub .

Leggi Crea ROM personalizzate per Android online:
https://riptutorial.com/it/android/topic/9212/crea-rom-personalizzate-per-android

Capitolo 69: Crea una classe Singleton per il messaggio Toast

introduzione

I messaggi di toast sono il modo più semplice per fornire feedback all'utente. Per impostazione predefinita, Android fornisce un messaggio con un messaggio di colore grigio in cui è possibile impostare il messaggio e la durata del messaggio. Se abbiamo bisogno di creare un messaggio di toast più personalizzabile e riutilizzabile, possiamo implementarlo da soli con l'uso di un layout personalizzato. Ancora più importante quando lo stiamo implementando, l'uso del modello di progettazione di Singelton renderà più semplice il mantenimento e lo sviluppo della classe di messaggi di brindisi personalizzati.

Sintassi

- Toast Toast (Contesto contesto)
- void setDuration (int duration)
- void setGravity (int gravity, int xOffset, int yOffset)
- void setView (Visualizza vista)
- void show ()

Parametri

Parametro	dettagli
contesto	Contesto rilevante che deve visualizzare il messaggio di brindisi. Se si utilizza questo passaggio di attività "questo" parola chiave o Se si utilizza in passaggio di gestione come "getActivity ()".
vista	Creare una vista personalizzata e passare questo oggetto vista a questo.
gravità	Passa la posizione di gravità del tostapane. Tutte le posizioni sono state aggiunte sotto la classe Gravity come variabili statiche. Le posizioni più comuni sono Gravity.TOP, Gravity.BOTTOM, Gravity.LEFT, Gravity.RIGHT.
xOffset	Offset orizzontale del messaggio di toast.
YOffset	Offset verticale del messaggio di toast.
durata	Durata dello spettacolo di brindisi Possiamo impostare Toast.LENGTH_SHORT o Toast.LENGTH_LONG

Osservazioni

Il messaggio Toast è un modo semplice per fornire feedback all'utente su qualcosa che sta accadendo. Se hai bisogno di un modo più avanzato per dare feedback puoi usare i dialoghi o la barra degli snack.

Per ulteriori dettagli sul messaggio di brindisi, consultare questa documentazione.
https://developer.android.com/reference/android/widget/Toast.html

Examples

Crea la tua classe singleton per i massaggi ai toast

Ecco come creare la tua classe singleton per i messaggi di brindisi, Se la tua applicazione deve mostrare successo, avvertimenti e messaggi di pericolo per diversi casi d'uso, puoi usare questa classe dopo averla modificata secondo le tue specifiche.

```
    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();
        }
    }
```

Leggi Crea una classe Singleton per il messaggio Toast online:
https://riptutorial.com/it/android/topic/10843/crea-una-classe-singleton-per-il-messaggio-toast

Capitolo 70: Creare le tue librerie per le applicazioni Android

Examples

Creazione del progetto di libreria

Per creare una libreria, devi utilizzare `File -> New -> New Module -> Android Library` . Questo creerà un progetto di libreria di base.

Al termine, è necessario disporre di un progetto impostato come segue:

```
[project root directory]
    [library root directory]
    [gradle]
    build.gradle //project level
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    settings.gradle //this is important!
```

Il tuo file `settings.gradle` deve contenere quanto segue:

```
include ':[library root directory]'
```

La tua `[library root directory]` deve contenere quanto segue:

```
[libs]
[src]
   [main]
      [java]
         [library package]
   [test]
      [java]
         [library package]
build.gradle //"app"-level
proguard-rules.pro
```

Il tuo file " `build.gradle` " `build.gradle` deve contenere quanto segue:

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

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.2"

    defaultConfig {
        minSdkVersion 14
        targetSdkVersion 23
```

```
    }
}
```

Con quello, il tuo progetto dovrebbe funzionare bene!

Uso della libreria nel progetto come modulo

Per utilizzare la libreria, è necessario includerla come dipendenza con la seguente riga:

```
compile project(':[library root directory]')
```

Crea una libreria disponibile su Jitpack.io

Effettuare le seguenti operazioni per creare la libreria:

1. Crea un account GitHub.
2. Crea un repository Git contenente il tuo progetto di libreria.
3. Modifica il file `build.gradle` del tuo progetto di `build.gradle` aggiungendo il seguente codice:

   ```
   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
   }
   ```

   Assicurati di impegnare / spingere le modifiche precedenti su GitHub.

4. Crea una versione dal codice corrente su Github.
5. Esegui `gradlew install` sul tuo codice.

6. La tua libreria è ora disponibile dalla seguente dipendenza:

```
compile 'com.github.[YourUser]:[github repository name]:[release tag]'
```

Leggi Creare le tue librerie per le applicazioni Android online:
https://riptutorial.com/it/android/topic/4118/creare-le-tue-librerie-per-le-applicazioni-android

Capitolo 71: Creazione della schermata Splash

Osservazioni

Il primo esempio (una schermata iniziale di base) non è il modo più efficiente per gestirlo. In quanto tale, è la schermata iniziale di base.

Examples

Una schermata iniziale di base

Una schermata iniziale è come qualsiasi altra attività, ma può gestire tutte le esigenze di avvio in background. Esempio:

Manifesto:

```
<?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>
```

Ora il nostro splash-screen sarà chiamato come prima attività.

Ecco un esempio di splashscreen che gestisce anche alcuni elementi critici dell'app:

```
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(){
```

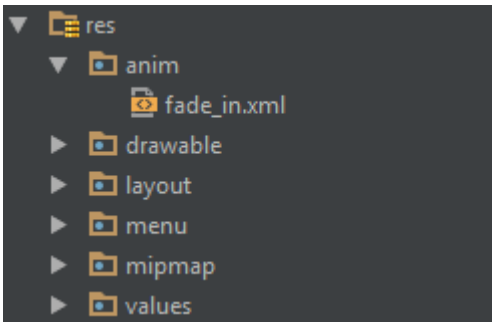
```
        super.onPause();
        finish();
    }

}
```

Schermata iniziale con animazione

Questo esempio mostra una schermata iniziale semplice ma efficace con animazione che può essere creata utilizzando Android Studio.

Passaggio 1: crea un'animazione

Crea una nuova directory denominata *anim* nella directory *res* . Fare clic con il tasto destro del mouse e creare un nuovo file di *risorse di animazione* denominato *fade_in.xml* :



Quindi, inserisci il seguente codice nel file *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>
```

Passaggio 2: crea un'attività

Crea un'attività *vuota* utilizzando Android Studio chiamato `Splash` . Quindi, inserisci il seguente codice:

```
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);
    }
}
```

Quindi, inserisci il seguente codice nel file di layout:

```
<?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>
```

Passaggio 3: sostituire il programma di avvio predefinito

Trasforma la tua attività `Splash` in un launcher aggiungendo il seguente codice al file *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>
```

Quindi, rimuovi l'attività di avvio predefinita rimuovendo il seguente codice dal file *AndroidManifest* :

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />

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

Leggi Creazione della schermata Splash online:
https://riptutorial.com/it/android/topic/9316/creazione-della-schermata-splash

Capitolo 72: Creazione di finestre sovrapposte (sempre in primo piano)

Examples

Sovrapposizione popup

Per mettere la tua vista sopra ogni applicazione, devi assegnare la tua vista al gestore di finestre corrispondente. Per questo è necessaria l'autorizzazione di avviso di sistema, che può essere richiesta aggiungendo la seguente riga al file manifest:

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

Nota: se l'applicazione viene distrutta, la visualizzazione verrà rimossa dal gestore di finestre. Pertanto, è preferibile creare la vista e assegnarla al gestore di finestre tramite un servizio in primo piano.

Assegnazione di una vista a WindowManager

È possibile recuperare un'istanza del gestore di finestre come segue:

```
WindowManager mWindowManager = (WindowManager)
mContext.getSystemService(Context.WINDOW_SERVICE);
```

Per definire la posizione della tua vista, devi creare alcuni parametri di layout come segue:

```
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;
```

Ora è possibile assegnare la vista insieme ai parametri di layout creati all'istanza del gestore finestre come segue:

```
mWindowManager.addView(yourView, mLayoutParams);
```

Ecco! La tua vista è stata posizionata con successo in cima a tutte le altre applicazioni.

Nota: la vista non verrà posizionata sopra il blocco tasti.

Concessione dell'autorizzazione SYSTEM_ALERT_WINDOW su Android 6.0 e

versioni successive

Da Android 6.0 questa autorizzazione deve essere concessa in modo dinamico,

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

Gettare sotto l'autorizzazione ha negato l'errore su 6.0,

```
Caused by: android.view.WindowManager$BadTokenException: Unable to add window
android.view.ViewRootImpl$W@86fb55b -- permission denied for this window type
```

Soluzione: -

Richiesta di permesso Overlay come di seguito,

```
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);
}
```

Controlla il risultato,

```
@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...
        }
    }
}
```

Leggi Creazione di finestre sovrapposte (sempre in primo piano) online: https://riptutorial.com/it/android/topic/6214/creazione-di-finestre-sovrapposte--sempre-in-primo-piano-

Capitolo 73: Creazione di viste personalizzate

Examples

Creazione di viste personalizzate

Se hai bisogno di una vista completamente personalizzata, dovrai creare sottoclassi `View` (la superclasse di tutte le viste Android) e fornire i tuoi `onMeasure(...)` ridimensionamento personalizzato (`onMeasure(...)`) e drawing (`onDraw(...)`):

1. **Crea il tuo scheletro di visualizzazione personalizzato:** questo è fondamentalmente lo stesso per ogni vista personalizzata. Qui creiamo lo scheletro per una vista personalizzata che può disegnare uno smiley, chiamato `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. **Inizializza le tue pitture:** gli oggetti `Paint` sono i pennelli della tua tela virtuale che definiscono il modo in cui i tuoi oggetti geometrici sono resi (es. Colore, stile di riempimento e tratto, ecc.). Qui creiamo due `Paint` , una vernice gialla riempita per il cerchio e una vernice nera per gli occhi e la bocca:

```
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. **Implementa il tuo metodo `onMeasure(...)` :** è necessario in modo che i layout principali (ad esempio `FrameLayout`) possano allineare correttamente la vista personalizzata. Fornisce un set di `measureSpecs` che è possibile utilizzare per determinare l'altezza e la larghezza della vista. Qui creiamo un quadrato assicurandoti che l'altezza e la larghezza siano le stesse:

```
@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);
}
```

   Nota che `onMeasure(...)` deve contenere almeno una chiamata a `setMeasuredDimension(..)` altrimenti la tua vista personalizzata si bloccherà con un `IllegalStateException` .

4. **Implementa il tuo metodo `onSizeChanged(...)` :** questo ti consente di catturare l'altezza e la larghezza attuali della tua vista personalizzata per regolare correttamente il tuo codice di rendering. Qui calcoliamo solo il nostro centro e il nostro raggio:

```
@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. **Implementa il tuo metodo `onDraw(...)` :** è qui che implementa il rendering effettivo della tua vista. Fornisce un oggetto `Canvas` cui puoi disegnare (consulta la documentazione ufficiale su `Canvas` per tutti i metodi di disegno disponibili).

```
@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. **Aggiungi la tua vista personalizzata a un layout:** la visualizzazione personalizzata ora può essere inclusa in qualsiasi file di layout che hai. Qui lo avvolgiamo in 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>
```

Si noti che è consigliabile creare il progetto dopo aver completato il codice di visualizzazione. Senza costruirlo, non potrai vedere la vista su una schermata di anteprima in Android Studio.

Dopo aver messo tutto insieme, dovresti essere accolto con la seguente schermata dopo aver avviato l'attività contenente il layout di cui sopra:

Aggiunta di attributi alle viste

Le visualizzazioni personalizzate possono anche utilizzare attributi personalizzati che possono essere utilizzati nei file di risorse del layout Android. Per aggiungere attributi alla tua vista personalizzata devi fare quanto segue:

1. **Definisci il nome e il tipo dei tuoi attributi:** questo viene fatto all'interno di `res/values/attrs.xml` (`res/values/attrs.xml` se necessario). Il seguente file definisce un attributo colore per il colore del viso della nostra emoticon e un attributo enum per l'espressione della faccina:

```
<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. **Usa i tuoi attributi all'interno del tuo layout:** questo può essere fatto all'interno di qualsiasi file di layout che usa la tua vista personalizzata. Il seguente file di layout crea uno schermo con un sorriso giallo felice:

```
<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>
```

   Suggerimento: gli attributi personalizzati non funzionano con gli `tools:` prefisso in Android Studio 2.1 e versioni precedenti (e possibilmente nelle versioni future). In questo esempio, sostituendo l' `app:smileyColor` con gli `tools:smileyColor` comporterebbe che `smileyColor` non venisse impostato né in fase di esecuzione né in fase di progettazione.

3. **Leggi i tuoi attributi:** questo viene fatto all'interno del tuo codice sorgente di visualizzazione personalizzato. Il seguente frammento di `SmileyView` mostra come `SmileyView` gli attributi:

```
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. **(Facoltativo) Aggiungi stile predefinito:** questo viene fatto aggiungendo uno stile con i valori predefiniti e caricandolo all'interno della tua vista personalizzata. Il seguente stile predefinito di smiley rappresenta un felice giallo:

   ```
   <!-- styles.xml -->
   <style name="DefaultSmileyStyle">
       <item name="smileyColor">#ffff00</item>
       <item name="smileyExpression">happy</item>
   </style>
   ```

   Che viene applicato nel nostro `SmileyView` aggiungendolo come l'ultimo parametro della chiamata per `obtainStyledAttributes` (vedere codice nel passaggio 3):

   ```
   TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView,
   defStyleAttr, R.style.DefaultSmileyViewStyle);
   ```

   Si noti che qualsiasi valore di attributo impostato nel file di layout gonfiato (vedere il codice nel passaggio 2) sostituirà i valori corrispondenti dello stile predefinito.

5. **(Facoltativo) Fornire stili all'interno di temi:** ciò viene fatto aggiungendo un nuovo attributo di riferimento di stile che può essere utilizzato all'interno dei temi e fornendo uno stile per tale attributo. Qui semplicemente `smileyStyle` nostro attributo di riferimento `smileyStyle` :

   ```
   <!-- attrs.xml -->
   <attr name="smileyStyle" format="reference" />
   ```

   Che quindi forniamo uno stile per il nostro tema dell'app (qui riutilizziamo solo lo stile predefinito del passaggio 4):

   ```
   <!-- themes.xml -->
   <style name="AppTheme" parent="AppBaseTheme">
       <item name="smileyStyle">@style/DefaultSmileyStyle</item>
   </style>
   ```

Creazione di una vista composta

Una **vista composta** è un `ViewGroup` personalizzato trattato come una singola vista dal codice del programma circostante. Tale ViewGroup può essere davvero utile nella progettazione DDD ,

perché può corrispondere a un aggregato, in questo esempio, un contatto. Può essere riutilizzato ovunque venga visualizzato il contatto.

Ciò significa che il codice del controller circostante, un'attività, un frammento o un adattatore, può semplicemente passare l'oggetto dati alla vista senza selezionarlo in un numero di diversi widget dell'interfaccia utente.

Ciò facilita il riutilizzo del codice e rende il design migliore secondo i principi SOLID .

Il layout XML

Questo di solito è dove inizi. Hai un po 'di XML esistente che ti ritrovi a riutilizzare, forse come un `<include/>` . Estraetelo in un file XML separato e avvolgete il tag radice in 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>
```

Questo file XML continua a funzionare perfettamente nell'editor di layout di Android Studio. Puoi trattarlo come qualsiasi altro layout.

Il composto ViewGroup

Una volta ottenuto il file XML, creare il gruppo di viste personalizzato.

```
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);
        }
    }
}
```

Il metodo `init(Context, AttributeSet)` è il punto in cui leggere gli attributi XML personalizzati come spiegato in Aggiunta di attributi alle viste .

Con questi pezzi sul posto, puoi usarlo nella tua app.

Utilizzo in XML

Ecco un esempio di `fragment_contact_info.xml` che illustra come mettere un singolo ContactView in cima a un elenco di messaggi:

```
<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 nel codice

Ecco un esempio di `RecyclerView.Adapter` che mostra un elenco di contatti. Questo esempio illustra quanto è più pulito il codice del controller quando è completamente privo di manipolazione della vista.

```
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
        }
    }
}
```

Suggerimenti sulle prestazioni di CustomView

Non assegnare nuovi oggetti in **onDraw**

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = new Paint(); //Do not allocate here
}
```

Invece di disegnare drawable su tela ...

```
drawable.setBounds(boundsRect);

drawable.draw(canvas);
```

Usa una bitmap per disegnare più velocemente:

```
canvas.drawBitmap(bitmap, srcRect, boundsRect, paint);
```

Non ridisegnare l'intera vista per aggiornarne solo una piccola parte. Invece ridisegna la parte specifica della vista.

```
invalidate(boundToBeRefreshed);
```

Se la tua vista sta facendo un'animazione continua, ad esempio un quadrante che mostra ogni secondo, almeno ferma l'animazione su `onStop()` dell'attività e ricominciala su `onStart()` dell'attività.

Non eseguire calcoli all'interno del metodo `onDraw` di una vista, dovresti invece terminare il disegno prima di chiamare `invalidate()` . Usando questa tecnica puoi evitare la caduta del fotogramma nella tua vista.

rotazioni

Le operazioni di base di una vista sono traslazione, rotazione, ecc ... Quasi tutti gli sviluppatori

hanno affrontato questo problema quando usano bitmap o gradienti nella loro vista personalizzata. Se la vista mostra una vista ruotata e la bitmap deve essere ruotata in quella visualizzazione personalizzata, molti di noi penseranno che sarà costoso. Molti pensano che la rotazione di una bitmap sia molto costosa perché per farlo è necessario tradurre la matrice di pixel della bitmap. Ma la verità è che non è così difficile! Invece di ruotare la bitmap, basta ruotare la tela stessa!

```
// 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 composta per SVG / VectorDrawable come drawableRight

Motivo principale per sviluppare questa vista composta è, sotto 5.0 dispositivi non supporta svg in disegnabile all'interno di TextView / EditText. Un altro pro è che possiamo impostare `height` e `width` del `drawableRight` all'interno di `EditText` . L'ho separato dal mio progetto e creato in un modulo separato.

Nome del modulo: custom_edit_drawable (nome breve per il prefisso-c_d_e)

prefisso "c_d_e_" da utilizzare in modo che le risorse del modulo app non le sovrascrivano per errore. Esempio: il prefisso "abc" è utilizzato da google nella libreria di supporto.

build.gradle

```
dependencies {
   compile 'com.android.support:appcompat-v7:25.3.1'
}
```

usa AppCompat> = 23

File di layout: 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>
```

Attributi personalizzati: 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>
```

Codice: 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();
        }
    }
}
```

Esempio: come usare la vista sopra

Layout: 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>
```

Attività: 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);
    }
}
```

Risposta a Touch Events

Molte visualizzazioni personalizzate devono accettare l'interazione dell'utente sotto forma di eventi tattili. Puoi accedere agli eventi touch ignorando `onTouchEvent` . Ci sono un certo numero di azioni che puoi filtrare. I principali sono

- `ACTION_DOWN` : viene attivato una volta quando il dito tocca per la prima volta la vista.
- `ACTION_MOVE` : viene chiamato ogni volta che il dito si sposta leggermente sulla vista. Viene chiamato molte volte.
- `ACTION_UP` : questa è l'ultima azione da chiamare quando si solleva il dito dallo schermo.

È possibile aggiungere il seguente metodo alla vista e osservare l'output del registro quando si tocca e si sposta il dito sulla 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;
}
```

Ulteriori letture:

- Documentazione ufficiale di Android: Risposta a Touch Events

Leggi Creazione di viste personalizzate online:
https://riptutorial.com/it/android/topic/1446/creazione-di-viste-personalizzate

Capitolo 74: Criteri della modalità rigorosa: uno strumento per catturare l'errore nel tempo di compilazione.

introduzione

La modalità rigorosa è una classe speciale introdotta in Android 2.3 per il debug. Questi strumenti di sviluppo rilevano le cose fatte accidentalmente e le portano alla nostra attenzione in modo che possiamo ripararle. È più comunemente utilizzato per catturare il disco accidentale o l'accesso alla rete sul thread principale delle applicazioni, dove vengono ricevute le operazioni dell'interfaccia utente e vengono eseguite le animazioni. StrictMode è fondamentalmente uno strumento per catturare il bug in modalità Compile Time.

Osservazioni

StrictMode è fondamentalmente uno strumento per catturare il bug in modalità Compile Time. Usando questo possiamo evitare perdite di memoria nelle nostre applicazioni.

Examples

Il seguente Snippet di codice consiste nell'impostare StrictMode per i criteri di thread. Questo codice deve essere impostato nei punti di ingresso della nostra applicazione.

```
StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
    .detectDiskWrites()
    .penaltyLog() //Logs a message to LogCat
    .build())
```

Il codice qui sotto riguarda le perdite di memoria, come quando viene rilevato o meno in SQLLite finalize.

```
StrictMode.setVmPolicy(new StrictMode.VmPolicy.Builder()
    .detectActivityLeaks()
    .detectLeakedClosableObjects()
    .penaltyLog()
    .build());
```

Leggi Criteri della modalità rigorosa: uno strumento per catturare l'errore nel tempo di compilazione. online: https://riptutorial.com/it/android/topic/8756/criteri-della-modalita-rigorosa--uno-strumento-per-catturare-l-errore-nel-tempo-di-compilazione-

Capitolo 75: Crittografia / decrittografia dei dati

introduzione

Questo argomento discute di come la crittografia e la decrittografia funzionano in Android.

Examples

Crittografia AES dei dati utilizzando la password in modo sicuro

Il seguente esempio crittografa un dato blocco di dati usando AES . La chiave di crittografia è derivata in modo sicuro (sale casuale, 1000 giri di SHA-256). La crittografia utilizza AES in modalità CBC con IV casuale.

Si noti che i dati memorizzati nella classe `EncryptedData` (`salt` , `iv` e `encryptedData`) possono essere concatenati a un singolo array di byte. È quindi possibile salvare i dati o trasmetterli 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;
}
```

Il seguente codice di esempio mostra come testare la crittografia e la decrittografia:

```
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();
}
```

Leggi Crittografia / decrittografia dei dati online:
https://riptutorial.com/it/android/topic/3471/crittografia---decrittografia-dei-dati

Capitolo 76: Crostini

introduzione

Un toast fornisce un feedback semplice su un'operazione in un piccolo popup e scompare automaticamente dopo un timeout. Riempie solo la quantità di spazio richiesta per il messaggio e l'attività corrente rimane visibile e interattiva.

Sintassi

- Toast makeText (Contesto contesto, testo CharSequence, durata int)
- Toast makeText (Contesto contesto, int resId, int durata)
- void setGravity (int gravity, int xOffset, int yOffset)
- void show ()

Parametri

Parametro	Dettagli
contesto	Il contesto per visualizzare il tuo Toast in. `this` è comunemente usato in un'attività e `getActivity()` è comunemente usato in un frammento
testo	Un CharSequence che specifica quale testo verrà mostrato nel Toast. È possibile utilizzare qualsiasi oggetto che implementa CharSequence, inclusa una stringa
resid	Un ID risorsa che può essere utilizzato per fornire una risorsa Stringa da visualizzare nel Toast
durata	Flag di interi che rappresentano per quanto tempo il Toast mostrerà. Le opzioni sono `Toast.LENGTH_SHORT` e `Toast.LENGTH_LONG`
gravità	Numero intero che specifica la posizione, o "gravità" del pane tostato. Vedi le opzioni qui
xOffset	Specifica l'offset orizzontale per la posizione Toast
YOffset	Specifica l'offset verticale per la posizione di Toast

Osservazioni

Un brindisi fornisce un semplice feedback su un'operazione in un piccolo popup. Riempie solo la quantità di spazio richiesta per il messaggio e l'attività corrente rimane visibile e interattiva.

Un'alternativa più recente a Toast è SnackBar. SnackBar offre uno stile visivo aggiornato e consente all'utente di ignorare il messaggio o intraprendere ulteriori azioni. Vedere la documentazione SnackBar per i dettagli.

Documentazione ufficiale:

https://developer.android.com/reference/android/widget/Toast.html

Examples

Imposta la posizione di un toast

Una notifica standard per il brindisi appare nella parte inferiore dello schermo allineata al centro orizzontale. Puoi cambiare questa posizione con `setGravity(int, int, int)` . Questo accetta tre parametri: una costante di gravità, uno spostamento di posizione x e uno spostamento di posizione y.

Ad esempio, se decidi che il toast dovrebbe apparire nell'angolo in alto a sinistra, puoi impostare la gravità in questo modo:

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

Mostrare un messaggio Toast

In Android, un Toast è un semplice elemento dell'interfaccia utente che può essere utilizzato per fornire un feedback contestuale a un utente.

Per visualizzare un semplice messaggio di Toast, possiamo fare quanto segue.

```
// 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();
```

Oppure, per mostrare un Toast in linea, senza aggrapparti all'oggetto Toast puoi:

```
Toast.makeText(context, "Ding! Your Toast is ready.", Toast.LENGTH_SHORT).show();
```

IMPORTANTE: assicurati che il metodo `show()` sia chiamato dal thread dell'interfaccia utente. Se stai cercando di mostrare un `Toast` da un thread diverso è possibile ad esempio utilizzare `runOnUiThread` metodo di un `Activity` .

Non riuscendo a farlo, il che significa provare a modificare l'interfaccia utente creando un Toast, genererà una `RuntimeException` che avrà il seguente aspetto:

```
java.lang.RuntimeException: Can't create handler inside thread that has not called
Looper.prepare()
```

Il modo più semplice per gestire questa eccezione è semplicemente usando runOnUiThread: la sintassi è mostrata di seguito.

```
   runOnUiThread(new Runnable() {
        @Override
        public void run() {
            // Your code here
        }
    });
```

Creare un Toast personalizzato

Se non si desidera utilizzare la vista Toast predefinita, è possibile fornire la propria utilizzando il `setView(View)` su un oggetto `Toast` .

Innanzitutto, crea il layout XML che vorresti usare nel tuo 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>
```

Quindi, quando crei il tuo Toast, gonfia la tua vista personalizzata da XML e chiama `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();
```

Modo sicuro per la visualizzazione dei thread Toast (Application Wide)

```
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();
                }
            }
        });
    }
```

Ricordati di aggiungere `MainApplication` in `manifest` .

Ora chiamalo da qualsiasi thread per visualizzare un messaggio di brindisi.

```
MainApplication.getApp().showToast("Some message", Toast.LENGTH_LONG);
```

Mostra il messaggio Toast sopra la tastiera Soft

Per impostazione predefinita, Android visualizza i messaggi Toast nella parte inferiore dello schermo, anche se la tastiera è visualizzata. Questo mostrerà un messaggio Toast appena sopra la tastiera.

```
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();
}
```

Thread modo sicuro di visualizzare un messaggio Toast (per AsyncTask)

Se non vuoi estendere l'applicazione e mantenere i thread dei tuoi toast sicuri, assicurati di mostrarli nella sezione post execute dei tuoi AsyncTasks.

```
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();
    }

}
```

Leggi Crostini online: https://riptutorial.com/it/android/topic/1741/crostini

Capitolo 77: Data / ora localizzate in Android

Osservazioni

Si consiglia di utilizzare i metodi della classe DateUtils per formattare le date che sono a conoscenza delle impostazioni internazionali, vale a dire che considerano le preferenze dell'utente (ad esempio, i formati di ora 12h / 24h). Questi metodi sono più appropriati per le date visualizzate all'utente.

Per le rappresentazioni di date completamente personalizzate, si consiglia di utilizzare la classe SimpleDateFormat , in quanto consente di controllare completamente tutti gli elementi di data.

Examples

Formato di data localizzato personalizzato con DateUtils.formatDateTime ()

DateUtils.formatDateTime () consente di fornire un tempo e, in base ai flag forniti, crea una stringa datetime localizzata. I flag consentono di specificare se includere elementi specifici (come il giorno della settimana).

```
Date date = new Date();
String localizedDate = DateUtils.formatDateTime(context, date.getTime(),
DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_WEEKDAY);
```

formatDateTime () si occupa automaticamente dei formati data appropriati.

Formattazione standard di data / ora in Android

Formatta una data:

```
Date date = new Date();
DateFormat df = DateFormat.getDateInstance(DateFormat.MEDIUM);
String localizedDate = df.format(date)
```

Formatta una data e un'ora. La data è in formato breve, il tempo è in formato lungo:

```
Date date = new Date();
DateFormat df = DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.LONG);
String localizedDate = df.format(date)
```

Data / ora completamente personalizzate

```
Date date = new Date();
df = new SimpleDateFormat("HH:mm", Locale.US);
String localizedDate = df.format(date)
```

Modelli comunemente usati:

- HH: ora (0-23)
- hh: hour (1-12)
- a: indicatore AM / PM
- mm: minuto (0-59)
- ss: secondo
- dd: giorno del mese (1-31)
- MM: mese
- aaaa: anno

Leggi Data / ora localizzate in Android online: https://riptutorial.com/it/android/topic/6057/data---ora-localizzate-in-android

Capitolo 78: Decorazioni RecyclerView

Sintassi

- RecyclerView addItemDecoration (decorazione RecyclerView.ItemDecoration)
- RecyclerView addItemDecoration (RecyclerView.ItemDecoration decoration, int index)

Parametri

Parametro	Dettagli
decorazione	la decorazione dell'oggetto da aggiungere a `RecyclerView`
indice	l'indice nella lista delle decorazioni per questo `RecyclerView` . Questo è l'ordine in cui vengono chiamati `getItemOffset` e `onDraw` . Le chiamate successive potrebbero sovrascrivere quelle precedenti.

Osservazioni

Le decorazioni sono statiche

Poiché le decorazioni sono solo disegnate, non è possibile aggiungere ad esse clicker o altre funzionalità dell'interfaccia utente.

Decorazioni multiple

In alcuni casi, l'aggiunta di decorazioni multiple a `RecyclerView` funzionerà, ma attualmente non ci sono API pubbliche per tenere conto di altre decorazioni possibili durante la misurazione o il disegno. È possibile ottenere i limiti della vista o i bordi decorati della vista, in cui i bordi decorati sono la somma di tutti gli offset decorativi applicati.

Altri argomenti correlati:

RecyclerView
RecyclerView onClickListeners

Javadoc ufficiale

https://developer.android.com/reference/android/support/v7/widget/RecyclerView.ItemDecoration.html

Examples

Disegnare un separatore

Questo disegnerà una linea in fondo a ogni vista ma l'ultima a fare da separatore tra gli elementi.

```
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);
            }
        }
    }
}
```

Margini per articolo con ItemDecoration

È possibile utilizzare `RecyclerView.ItemDecoration` per aggiungere margini aggiuntivi a ciascun elemento in RecyclerView. In alcuni casi questo può ripulire sia l'implementazione dell'adattatore che il codice XML della vista articolo.

```
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);
    }
}
```

Per abilitare la decorazione, è sufficiente aggiungerla a RecyclerView:

```
// in your onCreate()
RecyclerView rv = (RecyclerView) findItemById(R.id.myList);
rv.addItemDecoration(new MyItemDecoration(context));
```

Aggiungi divisore a RecyclerView

Prima di tutto è necessario creare una classe che estenda `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);
    }
}

}
```

Aggiungi `divider_blue.xml` alla tua cartella drawable:

```
<?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>
```

Quindi usarlo come:

```
recyclerView.addItemDecoration(new SimpleBlueDivider(context));
```

Il risultato sarà come:


28% 14:20
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)

Questa immagine è solo un esempio di come funzionano i divisori, se si desidera seguire le specifiche di Material Design quando si aggiungono i divisori si prega di dare un'occhiata a questo link: divisori e grazie @Brenden Kromhout fornendo link.

Come aggiungere divisori usando e DividerItemDecoration

`DividerItemDecoration` è un `RecyclerView.ItemDecoration` che può essere utilizzato come divisore tra gli elementi.

```
DividerItemDecoration mDividerItemDecoration = new DividerItemDecoration(context,
             mLayoutManager.getOrientation());
recyclerView.addItemDecoration(mDividerItemDecoration);
```

Supporta entrambi gli orientamenti utilizzando `DividerItemDecoration.VERTICAL` e `DividerItemDecoration.HORIZONTAL` .

ItemOffsetDecoration per GridLayoutManager in RecycleView

L'esempio seguente aiuterà a dare lo stesso spazio a un oggetto in 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;
    }
```

```
}
```

Puoi chiamare ItemDecoration come sotto il codice.

```
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);
```

e offset dell'elemento di esempio

```
<dimen name="item_offset">5dp</dimen>
```

Leggi Decorazioni RecyclerView online: https://riptutorial.com/it/android/topic/506/decorazioni-recyclerview

Capitolo 79: Definire il valore del passo (incremento) per RangeSeekBar personalizzato

introduzione

Una personalizzazione di Android RangeSeekBar proposta da Alex Florescu su https://github.com/anothem/android-range-seek-bar

Permette di definire un valore di passo (incremento), quando si sposta la barra di ricerca

Osservazioni

1- Aggiungi l'attributo increment in attrs.xml

```
<attr name="increment" format="integer|float"/>
```

2- Definire un valore predefinito in RangeSeekBar.java e creare anche l'attributo

```
private static final int DEFAULT_INCREMENT = 1;
private int increment;
```

3- Inizia il valore di incremento in privato void init (Contesto contesto, AttributeSet attrs)

```
if (attrs == null)
    increment = DEFAULT_INCREMENT;
else
    increment = a.getInt(R.styleable.RangeSeekBar_increment, DEFAULT_INCREMENT);
```

4- Definire il valore dell'incremento nel vuoto sincronizzato protetto onDraw (@NonNull Canvas canvas)

Dovrai sostituire il valore minText e maxText. Quindi, invece di:

- minText = valueToString (getSelectedMinValue ());
- maxText = valueToString (getSelectedMaxValue ());

Avrai: 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 - Ora devi solo usarlo. Spero che sia d'aiuto

Examples

Definire un valore di passo di 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" />
```

Leggi Definire il valore del passo (incremento) per RangeSeekBar personalizzato online: https://riptutorial.com/it/android/topic/8627/definire-il-valore-del-passo--incremento--per-rangeseekbar-personalizzato

Capitolo 80: Design dei materiali

introduzione

Material Design è una guida completa per la progettazione visuale, di movimento e di interazione tra piattaforme e dispositivi.

Osservazioni

Vedi anche il post del blog Android originale che introduce la libreria di supporto del design

Documentazione ufficiale

https://developer.android.com/design/material/index.html

Linee guida per la progettazione dei materiali

https://material.io/guidelines

Altre risorse e librerie di progettazione

https://design.google.com/resources/

Examples

Applicare un tema AppCompat

La libreria di supporto AppCompat fornisce temi per creare app con le specifiche di Material Design . Un tema con un genitore di `Theme.AppCompat` è necessario anche per un'attività per estendere `AppCompatActivity` .

Il primo passo è personalizzare la tavolozza dei colori del tuo tema per colorare automaticamente la tua app.
Nella tua app `res/styles.xml` puoi definire:

```
<!-- 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>
```

Invece di `Theme.AppCompat` , che ha uno sfondo scuro, puoi anche utilizzare `Theme.AppCompat.Light` o `Theme.AppCompat.Light.DarkActionBar` .

Puoi personalizzare il tema con i tuoi colori. Buone scelte sono nella tabella dei colori delle specifiche di progettazione dei materiali e nella tavolozza dei materiali . I colori "500" sono buone scelte per il primario (blu 500 in questo esempio); scegli "700" della stessa tonalità per quella scura; e un'ombra da una tonalità diversa come il colore dell'accento. Il colore principale viene utilizzato per la barra degli strumenti dell'app e la relativa voce nella schermata panoramica (app recenti), la variante più scura per colorare la barra di stato e il colore per evidenziare alcuni controlli.

Dopo aver creato questo tema, applicalo alla tua app in `AndroidManifest.xml` e applica il tema a qualsiasi attività specifica. Ciò è utile per applicare un tema `AppTheme.NoActionBar` , che consente di implementare configurazioni di barre degli strumenti non predefinite.

```
<application android:theme="@style/AppTheme"
    ...>
    <activity
        android:name=".MainActivity"
        android:theme="@style/AppTheme" />
</application>
```

Puoi anche applicare temi alle singole visualizzazioni usando `android:theme` e un tema `ThemeOverlay` . Ad esempio 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>
```

Aggiunta di una barra degli strumenti

Una `Toolbar` è una generalizzazione di `ActionBar` da utilizzare all'interno dei layout dell'applicazione. Mentre un `ActionBar` è tradizionalmente parte di una `Activity's` opaca finestra arredamento controllata dal quadro, una `Toolbar` può essere posta a qualsiasi livello arbitrario di nidificazione all'interno di una gerarchia vista. Può essere aggiunto effettuando le seguenti operazioni:

1. Assicurati che la seguente dipendenza sia aggiunta al file **build.gradle del** tuo modulo (ad esempio app) sotto dipendenze:

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

2. Imposta il tema della tua app su uno che **non** ha un `ActionBar` . Per farlo, modifica il tuo file **styles.xml** sotto `res/values` e imposta un tema `Theme.AppCompat` .
   In questo esempio stiamo usando `Theme.AppCompat.NoActionBar` come genitore del tuo `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>
```

   Puoi anche utilizzare `Theme.AppCompat.Light.NoActionBar` o `Theme.AppCompat.DayNight.NoActionBar` o qualsiasi altro tema che non abbia intrinsecamente un `ActionBar`

3. Aggiungi la `Toolbar` degli `Toolbar` al tuo layout delle attività:

```
<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"/>
```

   Sotto la `Toolbar` degli `Toolbar` puoi aggiungere il resto del tuo layout.

4. Nella tua `Activity` , imposta la `Toolbar` degli `Toolbar` come `ActionBar` per questa `Activity` . A condizione che si stia utilizzando la libreria *app* e una `AppCompatActivity` , si utilizzerà il metodo `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);

    //...
}
```

Dopo aver eseguito i passaggi precedenti, è possibile utilizzare il metodo `getSupportActionBar()` per manipolare la `Toolbar` degli `Toolbar` impostata come `ActionBar` .

Ad esempio, puoi impostare il titolo come mostrato di seguito:

```
getSupportActionBar().setTitle("Activity Title");
```

Ad esempio, puoi anche impostare il colore del titolo e dello sfondo come mostrato di seguito:

```
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)));
```

Aggiunta di un FloatingActionButton (FAB)

Nella progettazione del materiale, un pulsante di azione Mobile rappresenta l'azione principale in un'attività.
Sono caratterizzati da un'icona circolare che galleggia sopra l'interfaccia utente e presentano comportamenti di movimento che includono morphing, lancio e un punto di ancoraggio di trasferimento.

Assicurati che la seguente dipendenza venga aggiunta al file build.gradle dell'app in dipendenze:

```
compile 'com.android.support:design:25.3.1'
```

Ora aggiungi il `FloatingActionButton` al tuo file di layout:

```
<android.support.design.widget.FloatingActionButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_margin="16dp"
      android:src="@drawable/some_icon"/>
```

dove l'attributo `src` riferimento all'icona che deve essere utilizzata per l'azione mobile.
Il risultato dovrebbe assomigliare a questo (presumendo che il tuo colore di accento sia Material Pink):

Per impostazione predefinita, il colore di sfondo di `FloatingActionButton` verrà impostato sul colore dell'accento del tema. Inoltre, si noti che un `FloatingActionButton` richiede un margine attorno per funzionare correttamente. Il margine consigliato per il fondo è `16dp` per i telefoni e `24dp` per i tablet.

Qui ci sono le proprietà che puoi usare per personalizzare ulteriormente il `FloatingActionButton` (assumendo `xmlns:app="http://schemas.android.com/apk/res-auto` è dichiarato come namespace in cima al tuo layout):

- **`app:fabSize`** : può essere impostato su `normal` o `mini` per passare da una versione normale a una più piccola.
- **`app:rippleColor`** : imposta il colore dell'effetto di ripple del tuo `FloatingActionButton` . Può

essere una risorsa colore o una stringa esadecimale.
- `app:elevation` : può essere una stringa, un numero intero, un valore booleano, un valore a virgola mobile, un valore a virgola mobile.
- `app:useCompatPadding` : abilita il riempimento `app:useCompatPadding` . Forse un valore booleano, come `true` o `false` . Impostare su `true` per utilizzare compat pad su api-21 e versioni successive, in modo da mantenere un aspetto coerente con livelli API più vecchi.

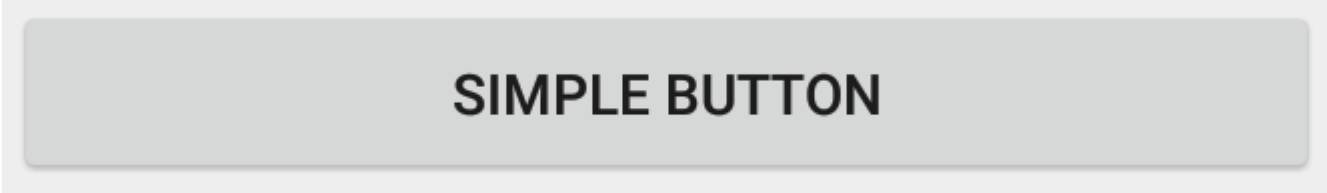
Si possono trovare altri esempi su FAB qui .

Bottoni con design materiale

La libreria di supporto AppCompat definisce diversi stili utili per i pulsanti , ognuno dei quali estende uno stile base `Widget.AppCompat.Button` che viene applicato a tutti i pulsanti per impostazione predefinita se si utilizza un tema `AppCompat` . Questo stile garantisce che tutti i pulsanti abbiano lo stesso aspetto per impostazione predefinita seguendo le specifiche di Material Design .

In questo caso il colore dell'accento è rosa.

1. Pulsante semplice: `@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. Pulsante colorato: `@style/Widget.AppCompat.Button.Colored`
   Lo stile `Widget.AppCompat.Button.Colored` estende lo stile `Widget.AppCompat.Button` e applica automaticamente il **colore dell'accento** selezionato nel tema dell'app.


```
<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"/>
```

   Se si desidera personalizzare il colore di sfondo senza modificare il colore dell'accento nel

tema principale, è possibile creare un *tema personalizzato* (estendendo il tema `ThemeOverlay`) per il proprio `Button` e assegnarlo `android:theme` del pulsante `android:theme` attributo `android:theme` :

```
<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"/>
```

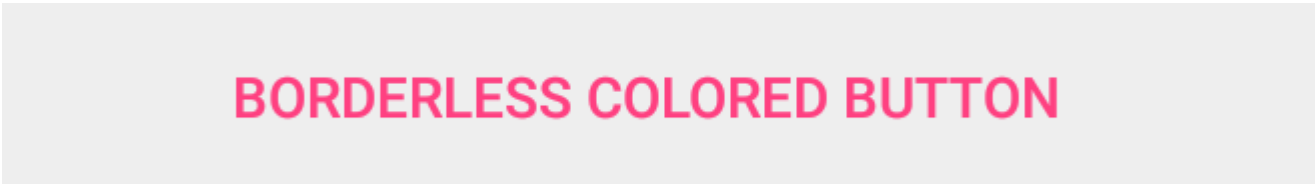
Definisci il tema in `res/values/themes.xml` :

```
 <style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
      <item name="colorAccent">@color/my_color</item>
 </style>
```

3. Pulsante senza bordi: `@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. Pulsante colorato senza bordi: `@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"/>
```

Come usare TextInputLayout

Assicurati che la seguente dipendenza venga aggiunta al file `build.gradle` dell'app in dipendenze:

```
compile 'com.android.support:design:25.3.1'
```

Mostra il suggerimento da un EditText come etichetta mobile quando viene inserito un valore.

```
<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>
```

Per visualizzare l'icona occhio di visualizzazione della password con TextInputLayout, possiamo utilizzare il seguente codice:

```
<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>
```

dove `app:passwordToggleEnabled="true"` & `android:inputType="textPassword"` sono richiesti i parametri.

`app` dovrebbe utilizzare lo spazio dei nomi `xmlns:app="http://schemas.android.com/apk/res-auto"`

Puoi trovare maggiori dettagli ed esempi nell'argomento dedicato.

Aggiungere un TabLayout

TabLayout fornisce un layout orizzontale per visualizzare le schede ed è comunemente usato in combinazione con un ViewPager .

Assicurati che la seguente dipendenza venga aggiunta al file `build.gradle` dell'app in dipendenze:

```
compile 'com.android.support:design:25.3.1'
```

Ora puoi aggiungere elementi a un TabLayout nel tuo layout usando la classe TabItem .

Per esempio:

```
<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>
```

Aggiungi un `OnTabSelectedListener` per ricevere una notifica quando una scheda in `TabLayout` è selezionata / deselezionata / riselezionata:

```
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) {

    }
});
```

Le schede possono anche essere aggiunte / rimosse dal `TabLayout` programmatico.

```
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` ha due modalità, fisse e scorrevoli.

```
tabLayout.setTabMode(TabLayout.MODE_FIXED);
tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);
```

Questi possono anche essere applicati in 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: i modi `TabLayout` si escludono a vicenda, il che significa che solo uno può essere attivo alla volta.

Il colore dell'indicatore di tabulazione è il colore di accento definito per il tema di Material Design. Puoi sovrascrivere questo colore definendo uno stile personalizzato in `styles.xml` e quindi applicando lo stile al tuo TabLayout:

```
<style name="MyCustomTabLayoutStyle" parent="Widget.Design.TabLayout">
    <item name="tabIndicatorColor">@color/your_color</item>
</style>
```

Quindi puoi applicare lo stile alla 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

Effetto tocco ondulato è stato introdotto con il design dei materiali in Android 5.0 (livello API 21) e l'animazione è implementata dalla nuova classe RippleDrawable .

> Disegnabile che mostra un effetto a catena in risposta ai cambiamenti di stato. La posizione di ancoraggio dell'ondulazione per un dato stato può essere specificata chiamando `setHotspot(float x, float y)` con l'identificatore dell'attributo stato corrispondente.

5.0

In generale, l'effetto a catena per **i pulsanti normali funziona per impostazione predefinita** nell'API 21 e sopra e, per altre visualizzazioni tangibili, può essere ottenuto specificando:

```
android:background="?android:attr/selectableItemBackground">
```

per le increspature contenute nella vista o:

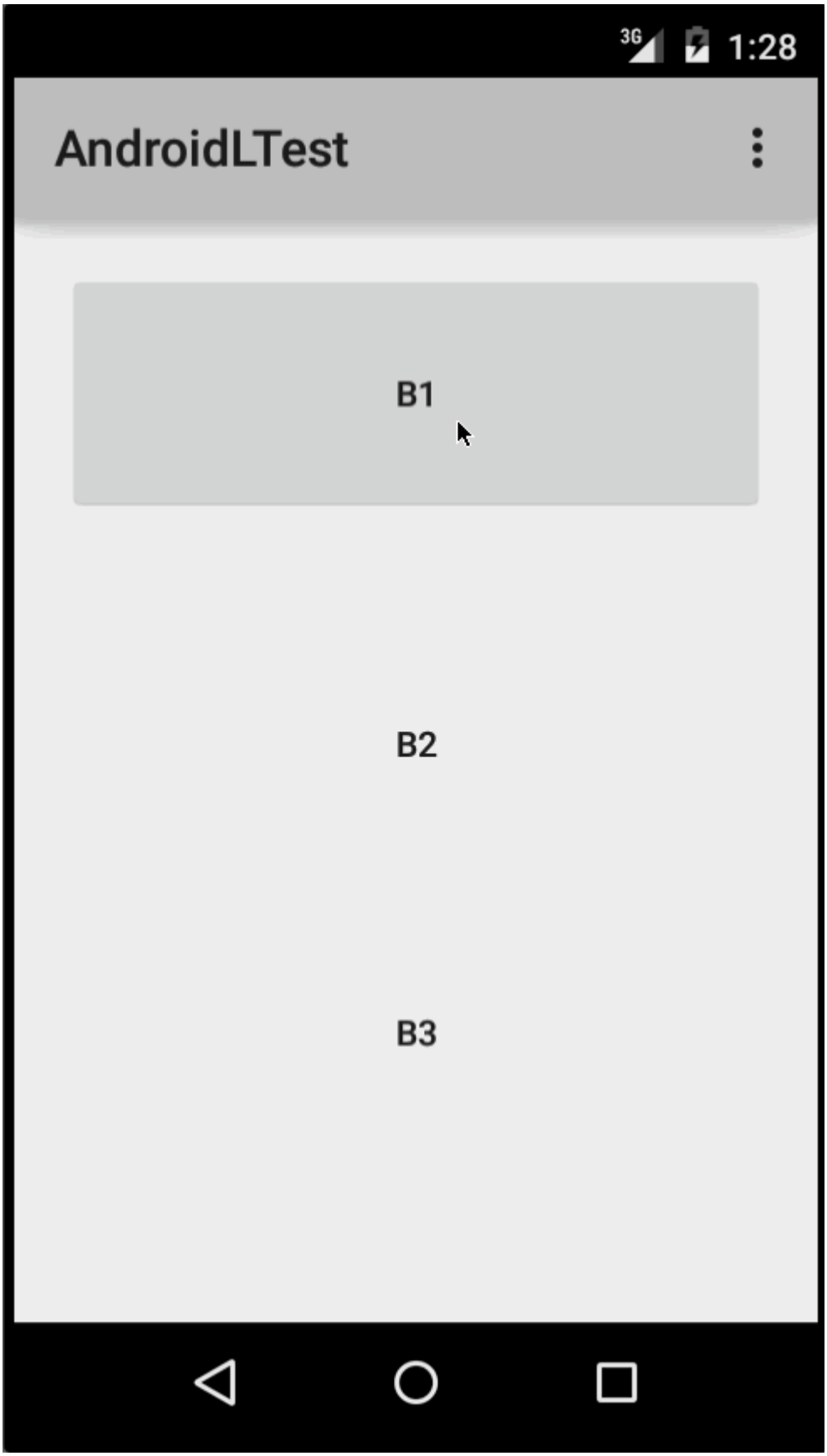
```
android:background="?android:attr/selectableItemBackgroundBorderless"
```

per le increspature che si estendono oltre i limiti della vista.

Ad esempio, nell'immagine qui sotto,

- B1 è un pulsante che non ha alcun background,

- B2 è impostato con `android:background="android:attr/selectableItemBackground"`
- B3 è impostato con `android:background="android:attr/selectableItemBackgroundBorderless"`



(Immagine per gentile concessione: http://blog.csdn.net/a396901990/article/details/40187203)

Puoi ottenere lo stesso codice usando:

```
int[] attrs = new int[]{R.attr.selectableItemBackground};
TypedArray typedArray = getActivity().obtainStyledAttributes(attrs);
int backgroundResource = typedArray.getResourceId(0, 0);
myView.setBackgroundResource(backgroundResource);
```

Ripples può anche essere aggiunto ad una vista usando l'attributo `android:foreground` allo stesso modo di sopra. Come suggerisce il nome, nel caso in cui l'ondulazione venga aggiunta al primo piano, il ripple verrà visualizzato sopra qualsiasi vista a cui è stato aggiunto (ad esempio, `ImageView` , un `LinearLayout` contiene più viste, ecc.).

Se si desidera personalizzare l'effetto di ripple in una vista, è necessario creare un nuovo file `XML` , all'interno della directory drawable.

Ecco alcuni esempi:

Esempio 1 : un'ondulazione illimitata

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#ffff0000" />
```

Esempio 2 : ripple con maschera e colore di sfondo

```
<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>
```

Se c'è una `view` con uno sfondo *già specificato* con una `shape` , `corners` e altri tag, per aggiungere un'increspatura a quella vista usa un `mask layer` e imposta il ripple come sfondo della vista.

Esempio :

```
 <?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>
```

Esempio 3 : Ripple in cima a una risorsa drawable

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#ff0000ff">
    <item android:drawable="@drawable/my_drawable" />
</ripple>
```

Utilizzo: per allegare il file ripple xml a qualsiasi vista, `my_ripple.xml` come sfondo come segue (supponendo che il tuo file ripple sia denominato `my_ripple.xml`):

```
<View
    android:id="@+id/myViewId"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/my_ripple" />
```

Selettore:

Il ripple drawable può anche essere utilizzato al posto dei selettori di elenchi di stati colore se la versione di destinazione è v21 o superiore (è anche possibile posizionare il selettore di ripple nella cartella `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>
```

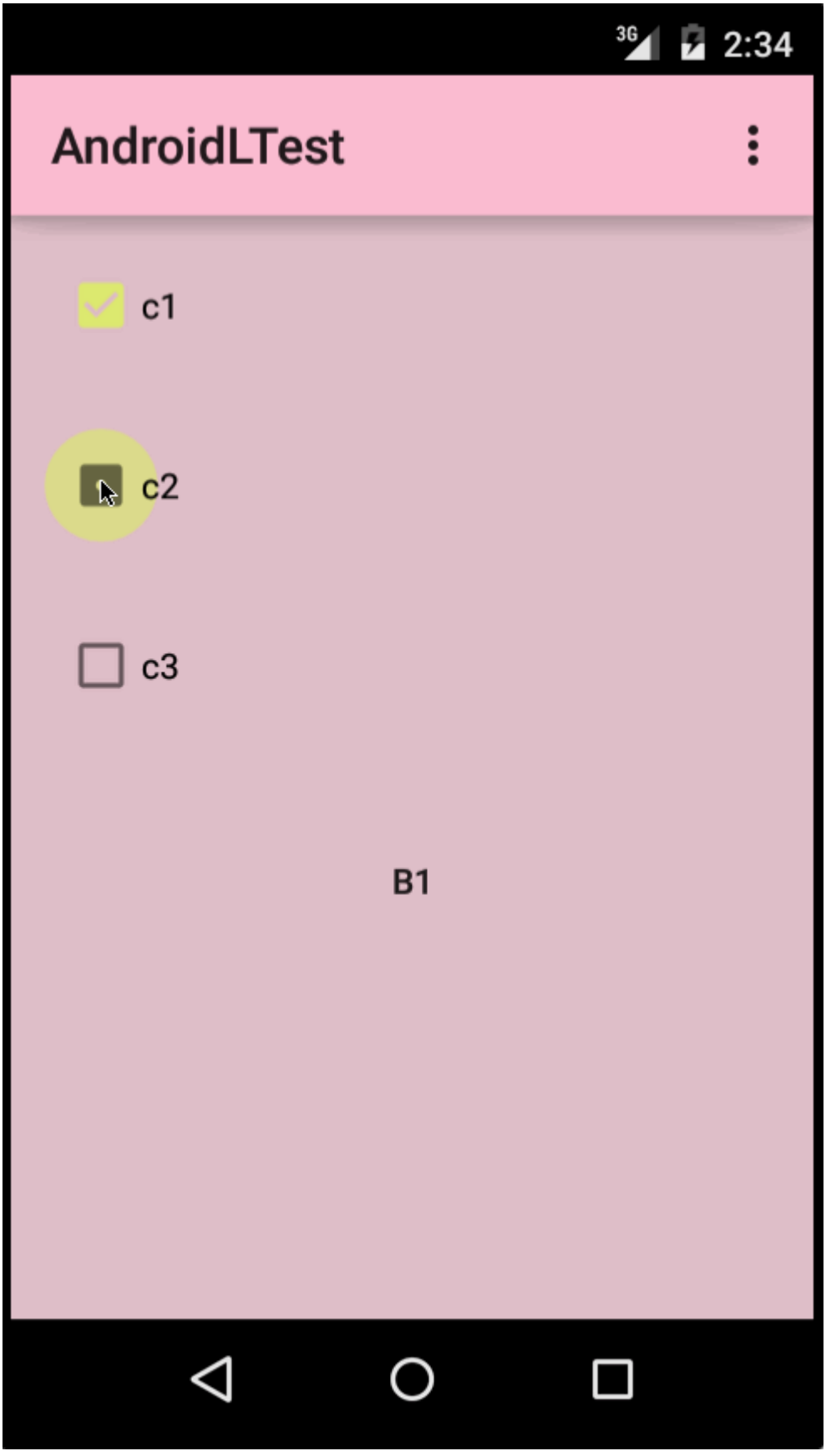
In questo caso, il colore dello stato predefinito della vista sarebbe bianco e lo stato premuto mostrerebbe il ripple disegnabile.

Nota: utilizzando `?android:colorControlHighlight` darà lo sfarfallio dello stesso colore delle increspature incorporate nella tua app.

Per cambiare solo il colore dell'ondulazione, puoi personalizzare il colore `android:colorControlHighlight` nel tuo tema in questo modo:

```
<?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>
```

e poi usa questo tema nelle tue attività, ecc. L'effetto sarebbe come l'immagine qui sotto:



(Immagine per gentile concessione: http://blog.csdn.net/a396901990/article/details/40187203)

Aggiungi un cassetto di navigazione

I cassetti di navigazione vengono utilizzati per spostarsi verso destinazioni di livello superiore in un'app.

Assicurarsi di aver aggiunto la libreria di supporto alla progettazione nel file `build.gradle` nelle dipendenze:

```
 dependencies {
    // ...
    compile 'com.android.support:design:25.3.1'
}
```

Successivamente, aggiungi `DrawerLayout` e `NavigationView` nel file di risorse del layout XML. `DrawerLayout` è solo un contenitore di fantasia che consente a `NavigationView` , l'attuale cassetto di navigazione, di scorrere verso sinistra o destra dello schermo. Nota: per i dispositivi mobili, la dimensione del cassetto standard è 320 dpi.

```
<!-- 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>
```

Ora, se lo desideri, crea un **file di intestazione** che servirà come parte superiore del tuo cassetto di navigazione. Questo è usato per dare un aspetto molto più elegante al cassetto.

```
<!-- 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>
```

Si fa riferimento nel tag `NavigationView app:headerLayout="@layout/drawer_header"` .
Questa `app:headerLayout` gonfia automaticamente il layout specificato nell'intestazione. Questo può essere fatto alternativamente in runtime 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);
```

Per popolare automaticamente il cassetto di navigazione con elementi di navigazione compatibili con il design del materiale, creare un file di menu e aggiungere elementi secondo necessità. Nota: mentre le icone per gli articoli non sono richieste, sono suggerite nelle specifiche di Material Design .
Si fa riferimento nel tag `NavigationView 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>
```

Per separare gli elementi in gruppi, inseriscili in un `<menu>` nidificato in un altro `<item>` con un attributo `android:title` o avvolgili con il tag `<group>` .

Ora che il layout è terminato, vai al codice `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();
```

Ora puoi fare tutto ciò che vuoi nella vista dell'header di `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 intestazione si comporta come qualsiasi altro `View` , quindi una volta che si utilizza `findViewById()` e aggiungere qualche altro `View` s al file di layout, è possibile impostare le proprietà di qualsiasi cosa in esso.

Puoi trovare maggiori dettagli ed esempi nell'argomento dedicato .

Fogli in fondo nella libreria di supporto del design

I fogli in basso scorrono verso l'alto dalla parte inferiore dello schermo per rivelare più contenuti. Sono stati aggiunti alla libreria di supporto Android nella versione v25.1.0 e supportano soprattutto

le versioni.

Assicurati che la seguente dipendenza venga aggiunta al file build.gradle dell'app in dipendenze:

```
compile 'com.android.support:design:25.3.1'
```

Fogli inferiori persistenti

È possibile ottenere un foglio inferiore persistente allegando un `BottomSheetBehavior` foglio inferiore a un bambino Vista di un `CoordinatorLayout BottomSheetBehavior` :

```
<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>
```

Quindi nel tuo codice puoi creare un riferimento usando:

```
// The View with the BottomSheetBehavior
View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

Puoi impostare lo stato di BottomSheetBehavior usando il metodo setState () :

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

Puoi utilizzare uno di questi stati:

- `STATE_COLLAPSED` : questo stato compresso è l'impostazione predefinita e mostra solo una parte del layout lungo il fondo. L'altezza può essere controllata con l' `app:behavior_peekHeight` attributo `app:behavior_peekHeight` (predefinito su 0)
- `STATE_EXPANDED` : lo stato completamente espanso del foglio inferiore, dove è visibile l'intero foglio inferiore (se la sua altezza è inferiore al `CoordinatorLayout` contenente) o l'intero `CoordinatorLayout` è pieno
- `STATE_HIDDEN` : disabilitato per impostazione predefinita (e abilitato con l' `app:behavior_hideable`

attributo `app:behavior_hideable`), abilitando questo consente agli utenti di scorrere verso il basso sul foglio inferiore per nascondere completamente il foglio inferiore

Oltre ad aprire o chiudere il BottomSheet al clic di una vista a scelta, un pulsante diciamo, ecco come attivare il comportamento del foglio e aggiornare la vista.

```
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);
            }
        }
    });
```

Ma il comportamento di `BottomSheet` ha anche una caratteristica in cui l'utente può interagire con lo scorrimento verso l'alto o verso il basso con un movimento DRAG. In tal caso, potremmo non essere in grado di aggiornare la Vista dipendente (come il pulsante sopra) Se lo stato del Foglio è cambiato. Per questo, ti piacerebbe ricevere i callback delle modifiche di stato, quindi puoi aggiungere un `BottomSheetCallback` per ascoltare gli eventi di scorrimento degli utenti:

```
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
   }
 });
```

E se vuoi che il tuo Bottom Sheet sia visibile solo in modalità COLLAPSED ed EXPANDED, non usare mai HIDE:

```
mBottomSheetBehavior2.setHideable(false);
```

DialogFragment di foglio inferiore

È inoltre possibile visualizzare un oggetto BottomSheetDialogFragment al posto di una vista nel foglio inferiore. Per fare ciò, devi prima creare una nuova classe che estenda BottomSheetDialogFragment.

All'interno del metodo `setupDialog()` , puoi gonfiare un nuovo file di layout e recuperare il BottomSheetBehavior della vista del contenitore nella tua attività. Una volta che hai il comportamento, puoi creare e associare un BottomSheetCallback con esso per eliminare il Fragment quando il foglio è nascosto.

```
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);
        }
    }
}
```

Infine, puoi chiamare show () su un'istanza del tuo frammento per visualizzarlo nel foglio inferiore.

```
BottomSheetDialogFragment bottomSheetDialogFragment = new BottomSheetDialogFragmentExample();
bottomSheetDialogFragment.show(getSupportFragmentManager(),
bottomSheetDialogFragment.getTag());
```

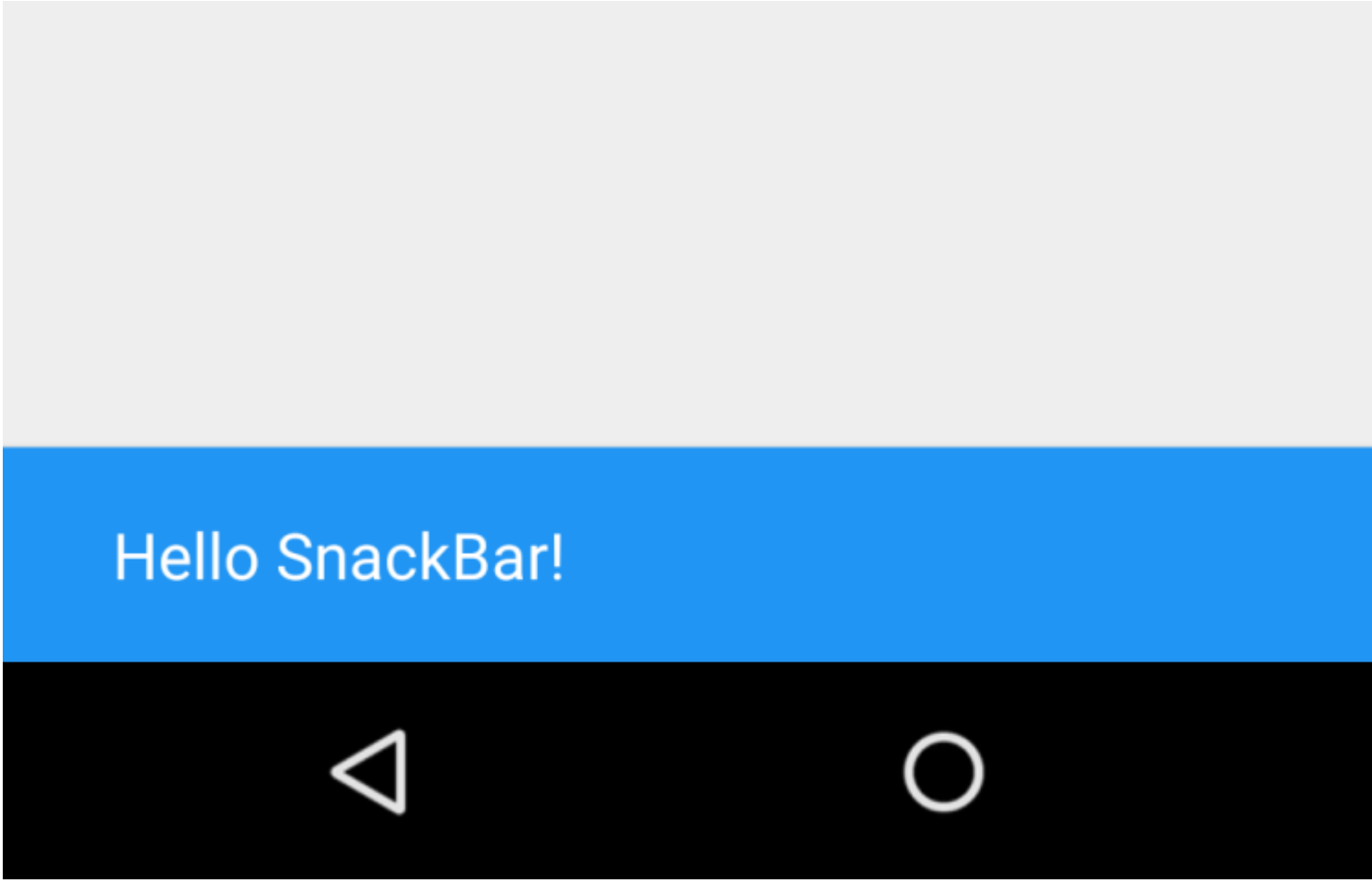
Puoi trovare maggiori dettagli nell'argomento dedicato

Aggiungi uno snack bar

Una delle caratteristiche principali di Material Design è l'aggiunta di uno `Snackbar` , che in teoria

sostituisce il precedente `Toast` . Come da documentazione di Android:

> Gli snack contengono una singola riga di testo direttamente correlata all'operazione eseguita. Possono contenere un'azione di testo, ma nessuna icona. I toast sono utilizzati principalmente per la messaggistica di sistema. Vengono visualizzati anche nella parte inferiore dello schermo, ma potrebbero non essere trascinati fuori dallo schermo.



I toast possono ancora essere utilizzati in Android per visualizzare i messaggi agli utenti, tuttavia, se hai deciso di optare per l'utilizzo del design dei materiali nella tua app, ti consigliamo di utilizzare effettivamente una barra di snack. Invece di essere visualizzato come una sovrapposizione sullo schermo, uno `Snackbar` apre dal basso.

Ecco come è fatto:

```
Snackbar snackbar = Snackbar
        .make(coordinatorLayout, "Here is your new Snackbar", Snackbar.LENGTH_LONG);
snackbar.show();
```

Per quanto riguarda il tempo necessario per mostrare lo `Snackbar` , abbiamo le opzioni simili a quelle offerte da un `Toast` o potremmo impostare una durata personalizzata in millisecondi:

- `LENGTH_SHORT`
- `LENGTH_LONG`
- `LENGTH_INDEFINITE`

- `setDuration()` (dalla versione `22.2.1`)

Puoi anche aggiungere funzionalità dinamiche al tuo `Snackbar` come `ActionCallback` o colore personalizzato. Tuttavia, prestate attenzione alle linee guida progettuali offerte da Android durante la personalizzazione di uno `Snackbar` .

L'implementazione dello `Snackbar` ha tuttavia una limitazione. Il layout principale della vista in cui implementerai uno `Snackbar` deve essere un `CoordinatorLayout` . Questo è così che può essere fatto il popup attuale dal basso.

Ecco come definire un `CoordinatorLayout` nel file xml del layout:

```
<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>
```

Il `CoordinatorLayout` deve quindi essere definito nel metodo `onCreate` della tua attività e quindi utilizzato durante la creazione dello `Snackbar` stesso.

Per ulteriori informazioni su `Snackbar` , consultare la documentazione ufficiale o l' argomento dedicato nella documentazione.

Leggi Design dei materiali online: https://riptutorial.com/it/android/topic/124/design-dei-materiali

Capitolo 81: Dialogo

Parametri

Linea	Descrizione
mostrare();	Mostra la finestra di dialogo
setContentView (R.layout.yourlayout);	imposta ContentView della finestra di dialogo sul layout personalizzato.
respingere()	Chiude la finestra di dialogo

Osservazioni

- La finestra di dialogo nel primo esempio (Dialog) non ha bisogno di chiamare `show()` quando viene creata man mano che viene gestita nel costruttore
- Le finestre di dialogo di avviso devono essere costruite tramite una nuova istanza della classe `AlertDialog.Builder()` . Seguendo il Pattern Builder , tutti i membri di AlertDialog.Builder possono essere concatenati con il metodo per "creare" l'istanza di dialogo.
- Il builder Alert Dialog può `show()` direttamente `show()` la finestra di dialogo - non è necessario chiamare `create()` then `show()` nell'istanza AlertDialog

Examples

Alert Dialog

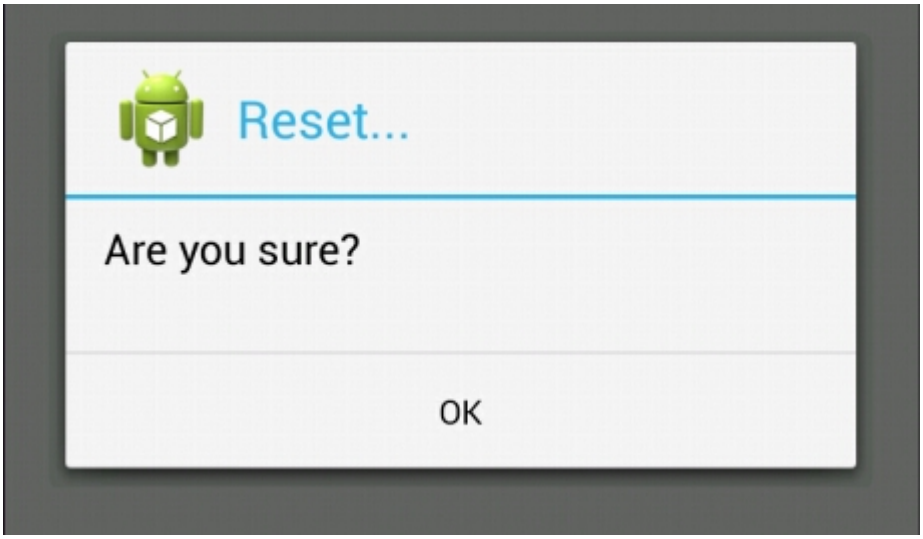
```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
                MainActivity.this);

        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();
```

Una finestra di dialogo di avviso di base

```
    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.
```

Ora questo codice raggiungerà questo:



(*Fonte immagine: WikiHow*)

Selezione data all'interno di DialogFragment

xml della finestra di dialogo:

```
<?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>
```

Classe di dialogo:

```
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;
    }

}
```

Attività che chiama il dialogo:

```
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` è il modo più semplice di usare `DatePicker` , perché puoi mostrare la finestra di dialogo ovunque nella tua app. Non è necessario implementare il proprio layout con il widget `DatePicker` .

Come mostrare il dialogo:

```
DatePickerDialog datePickerDialog = new DatePickerDialog(context, listener, year, month, day);
datePickerDialog.show();
```

È possibile ottenere il widget `DataPicker` dalla finestra di dialogo in alto, per ottenere l'accesso a più funzioni e, ad esempio, impostare la data minima in millisecondi:

```
DatePicker datePicker = datePickerDialog.getDatePicker();
datePicker.setMinDate(System.currentTimeMillis());
```

Date picker

`DatePicker` consente all'utente di scegliere la data. Quando creiamo una nuova istanza di `DatePicker` , possiamo impostare la data iniziale. Se non impostiamo la data iniziale, la data corrente verrà impostata per impostazione predefinita.

Possiamo mostrare `DatePicker` all'utente utilizzando `DatePickerDialog` o creando il nostro layout con il widget `DatePicker` .

Inoltre possiamo limitare l'intervallo di date, che l'utente può scegliere.

Impostando la data minima in millisecondi

```
//In this case user can pick date only from future
datePicker.setMinDate(System.currentTimeMillis());
```

Impostando la data massima in millisecondi

```
//In this case user can pick date only, before following week.
datePicker.setMaxDate(System.currentTimeMillis() + TimeUnit.DAYS.toMillis(7));
```

Per ricevere informazioni, su quale data è stata scelta dall'utente, dobbiamo usare `Listener` .

Se utilizziamo `DatePickerDialog` , possiamo impostare `OnDateSetListener` nel costruttore quando stiamo creando una nuova istanza di `DatePickerDialog` :

Esempio di utilizzo di `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) {

    }
}
```

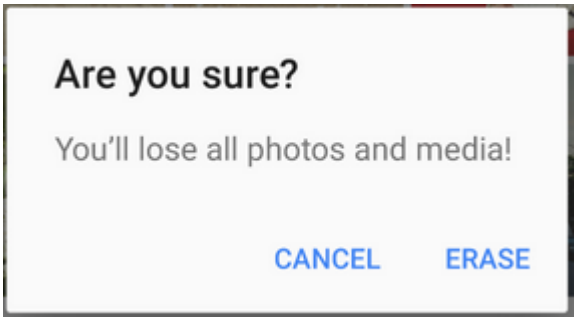
Altrimenti, se stiamo creando il nostro layout con il widget `DatePicker` , dobbiamo anche creare il nostro listener come mostrato in un altro esempio

Aggiunta di materiale Design AlertDialog all'app utilizzando Appcompat

`AlertDialog` è una sottoclasse di `Dialog` grado di visualizzare uno, due o tre pulsanti. Se si desidera solo visualizzare una stringa in questa finestra di dialogo, utilizzare il metodo `setMessage()` .

Il `AlertDialog` dal pacchetto `android.app` viene visualizzato in modo diverso su diverse versioni del sistema operativo Android.

La libreria Appcompat di Android V7 fornisce un'implementazione di `AlertDialog` che verrà visualizzata con Material Design su tutte le versioni del sistema operativo Android supportato, come mostrato di seguito:



Per prima cosa è necessario aggiungere la libreria V7 Appcompat al progetto. puoi farlo nel file build.gradle a livello di app:

```
dependencies {
    compile 'com.android.support:appcompat-v7:24.2.1'
```

```
    //........
}
```

Assicurati di importare la classe corretta:

```
import android.support.v7.app.AlertDialog;
```

Quindi creare AlertDialog come questo:

```
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 in AlertDialog

Possiamo sempre usare `ListView` o `RecyclerView` per la selezione dalla lista di elementi, ma se abbiamo una piccola quantità di scelte e tra quelle scelte vogliamo che l'utente ne selezioni una, possiamo usare `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();
    }
```

Forse, se non abbiamo bisogno di alcun particolare `ListView` , possiamo usare un modo base:

```
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();
```

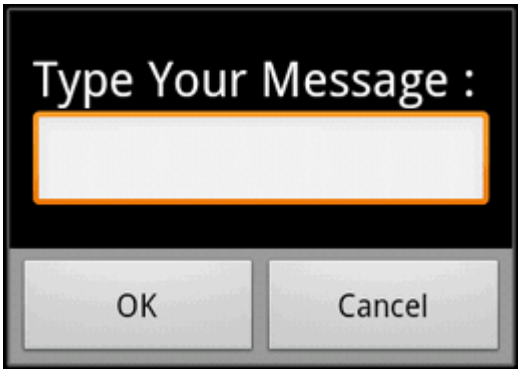
Finestra di dialogo di avviso personalizzato con EditText

```
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();
    }
```

File 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>
```



Finestra di dialogo personalizzata a schermo intero senza sfondo e senza titolo

in `styles.xml` aggiungi il tuo stile personalizzato:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppBaseTheme" parent="@android:style/Theme.Light.NoTitleBar.Fullscreen">
    </style>
</resources>
```

Crea il tuo layout personalizzato per la finestra di dialogo: `fullscreen.xml` :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

</RelativeLayout>
```

Quindi nel file java è possibile utilizzarlo per un'attività o finestra di dialogo ecc.

```
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();
    }
}
```

Finestra di dialogo di avviso con titolo multilinea

Il metodo setCustomTitle () di AlertDialog.Builder consente di specificare una vista arbitraria da utilizzare per il titolo della finestra di dialogo. Un uso comune di questo metodo è creare una finestra di avviso con un titolo lungo.

```
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();
```

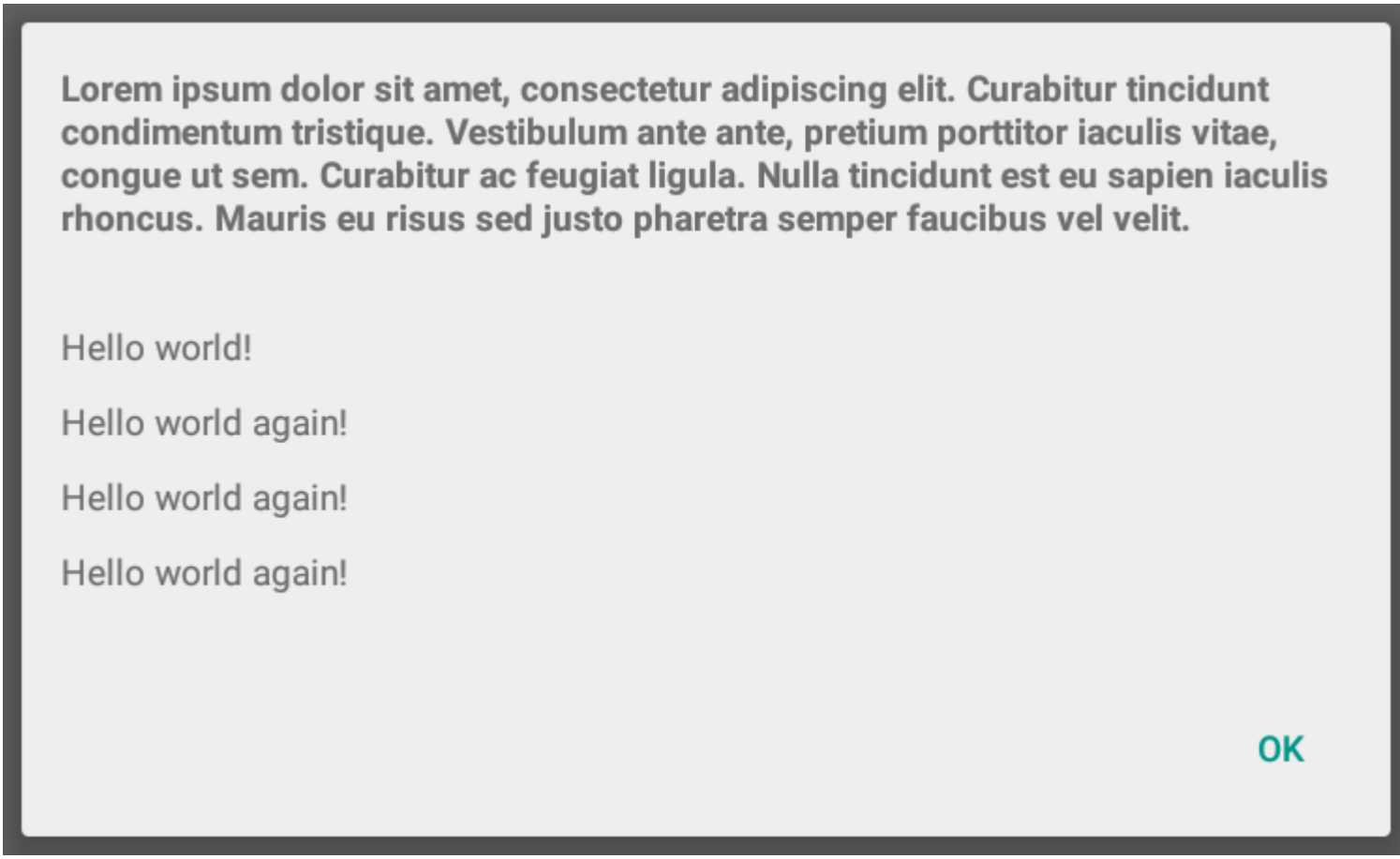
my_dialog_title.xml:

```
<?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>
```



Leggi Dialogo online: https://riptutorial.com/it/android/topic/1225/dialogo

Capitolo 82: Dipingere

introduzione

Una vernice è uno dei quattro oggetti necessari per disegnare, insieme a una tela (trattiene le chiamate di disegno), una bitmap (contiene i pixel) e una primitiva di disegno (Rect, Path, Bitmap ...)

Examples

Creare un dipinto

Puoi creare una nuova vernice con uno di questi 3 costruttori:

- `new Paint()` Crea con le impostazioni predefinite
- `new Paint(int flags)` Crea con flag
- `new Paint(Paint from)` Copia le impostazioni da un'altra vernice

In genere si consiglia di non creare mai un oggetto paint o qualsiasi altro oggetto in onDraw () poiché può causare problemi di prestazioni. (Android Studio probabilmente ti avviserà) Invece, rendilo globale e inizializzalo nel tuo costruttore di classe in questo modo:

```
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);
        // ...
    }
}
```

Impostazione di Paint per il testo

Impostazioni di disegno del testo

- `setTypeface(Typeface typeface)` Imposta la faccia del font. Vedi il carattere
- `setTextSize(int size)` Imposta la dimensione del carattere, in pixel.
- `setColor(int color)` Imposta il colore del disegno della pittura, incluso il colore del testo. Puoi anche usare `setARGB(int a, int r, int g, int b` e `setAlpha(int alpha)`

- `setLetterSpacing(float size)` Imposta la spaziatura tra i caratteri, in ems. Il valore predefinito è 0, un valore negativo stringerà il testo, mentre uno positivo lo espanderà.
- `setTextAlign(Paint.Align align)` Imposta l'allineamento del testo rispetto all'origine. `Paint.Align.LEFT` lo disegnerà a destra dell'origine, `RIGHT` lo disegnerà a sinistra, e `CENTER` trarrà centrato sull'origine (orizzontalmente)
- `setTextSkewX(float skewX)` Questo potrebbe essere considerato come falso in corsivo. SkewX rappresenta l'offset orizzontale del testo in basso. (usa -0.25 per corsivo)
- `setStyle(Paint.Style style)` Riempi il testo `FILL` , il testo tratto `STROKE` o entrambi `FILL_AND_STROKE`

Si noti che è possibile utilizzare `TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, size, getResources().getDisplayMetrics())` per convertire da SP o DP a pixel.

Testo di misurazione

- `float width = paint.measureText(String text)` Misura la larghezza del testo
- `float height = paint.ascent()` Misura l'altezza del testo
- `paint.getTextBounds(String text, int start, int end, Rect bounds` Memorizza le dimensioni del testo. Hai assegnato il Rect, non può essere nullo:

```
String text = "Hello world!";
Rect bounds = new Rect();
paint.getTextBounds(text, 0, text.length(), bounds);
```

Esistono altri metodi per misurare, tuttavia questi tre dovrebbero adattarsi alla maggior parte degli scopi.

Impostazione di Paint per disegnare forme

- `setStyle(Paint.Style style)` Forma riempita `FILL` , forma Stroke `STROKE` o entrambi `FILL_AND_STROKE`
- `setColor(int color)` Imposta il colore del disegno della pittura. Puoi anche usare `setARGB(int a, int r, int g, int b` e `setAlpha(int alpha)`
- `setStrokeCap(Paint.Cap cap)` Imposta i limiti di riga, `ROUND` , `SQUARE` o `BUTT` (nessuno) Vedi questo .
- `setStrokeJoin(Paint.Join join)` Imposta i join di linea, `MITER` (a punta), `ROUND` o `BEVEL` . Vedi questo .
- `setStrokeMiter(float miter)` Imposta il limite del join mitra. Questo può impedire al mitre join di andare avanti indefinitamente, trasformandolo in un join smussato dopo x pixel. Vedi questo .
- `setStrokeWidth(float width)` Imposta la larghezza del tratto. `0` disegnerà in modalità attaccatura dei capelli, indipendente dalla matrice di tela. (sempre 1 pixel)

Impostazione delle bandiere

È possibile impostare i seguenti flag nel costruttore o con `setFlags(int flags)`

- `Paint.ANTI_ALIAS_FLAG` Abilita l'antialiasing, `Paint.ANTI_ALIAS_FLAG` il disegno.

- `Paint.DITHER_FLAG` Abilita il dithering. Se la precisione del colore è superiore a quella del dispositivo, ciò avverrà .
- `Paint.EMBEDDED_BITMAP_TEXT_FLAG` Abilita l'uso di caratteri bitmap.
- `Paint.FAKE_BOLD_TEXT_FLAG` disegnerà il testo con un effetto audace falso, può essere usato invece di usare un carattere grassetto. Alcuni tipi di carattere hanno uno stile audace, non lo sono in grassetto falso
- `Paint.FILTER_BITMAP_FLAG` sul campionamento di bitmap quando viene trasformato.
- `Paint.HINTING_OFF` , `Paint.HINTING_ON` Attiva `Paint.HINTING_ON` disattiva il suggerimento sui font, guarda questo
- `Paint.LINEAR_TEXT_FLAG` Disabilita il ridimensionamento dei caratteri, mentre le operazioni di disegno vengono ridimensionate
- `Paint.SUBPIXEL_TEXT_FLAG` testo verrà calcolato utilizzando la precisione subpixel.
- `Paint.STRIKE_THRU_TEXT_FLAG` testo disegnato verrà colpito
- `Paint.UNDERLINE_TEXT_FLAG` testo disegnato sarà sottolineato

Puoi aggiungere un flag e rimuovere flag come questo:

```
Paint paint = new Paint();
paint.setFlags(paint.getFlags() | Paint.FLAG);   // Add flag
paint.setFlags(paint.getFlags() & ~Paint.FLAG);  // Remove flag
```

Cercando di rimuovere una bandiera che non c'è o aggiungendo una bandiera che è già lì non cambierà nulla. Si noti inoltre che la maggior parte dei flag può anche essere impostata usando `set<Flag>(boolean enabled)` , ad esempio `setAntialias(true)` .

È possibile utilizzare `paint.reset()` per ripristinare il `paint.reset()` impostazioni predefinite. L'unico flag predefinito è `EMBEDDED_BITMAP_TEXT_FLAG` . Verrà impostato anche se si utilizza il `new Paint(0)` , lo si avrà

Leggi Dipingere online: https://riptutorial.com/it/android/topic/9141/dipingere

Capitolo 83: Disegni vettoriali

introduzione

Come suggerisce il nome, i drawable vettoriali si basano sulla grafica vettoriale. La grafica vettoriale è un modo di descrivere elementi grafici usando forme geometriche. Questo ti consente di creare un drawable basato su un grafico vettoriale XML. Ora non è necessario progettare immagini di dimensioni diverse per mdpi, hdpi, xhdpi e così via. Con Vector Drawable è necessario creare un'immagine una sola volta come file xml e ridimensionarla per tutti i dpi e per diversi dispositivi. Questo inoltre non risparmia spazio ma semplifica anche la manutenzione.

Parametri

Parametro	Dettagli
`<vector>`	Utilizzato per definire un vettore disegnabile
`<group>`	Definisce un gruppo di percorsi o sottogruppi, oltre a informazioni sulla trasformazione. Le trasformazioni sono definite nelle stesse coordinate del viewport. E le trasformazioni vengono applicate nell'ordine di scala, ruotano poi traducono.
`<path>`	Definisce i percorsi da tracciare.
`<clip-path>`	Definisce il percorso per essere la clip corrente. Si noti che il percorso della clip si applica solo al gruppo corrente e ai relativi figli.

Osservazioni

Aggiorna il file **build.gradle** .

```
dependencies {
    ...
   compile 'com.android.support:appcompat-v7:23.2.1'
}
```

Se stai usando la versione **2.0 o successiva** del **plugin Gradle** , aggiungi il seguente codice.

```
// Gradle Plugin 2.0+
 android {
   defaultConfig {
     vectorDrawables.useSupportLibrary = true
    }
 }
```

Se stai usando la versione **1.5 o successiva** del **plugin Gradle** , aggiungi il seguente codice.

```
// Gradle Plugin 1.5
 android {
   defaultConfig {
     generatedDensities = []
  }

  // This is handled for you by the 2.0+ Gradle Plugin
  aaptOptions {
    additionalParameters "--no-version-vectors"
  }
 }
```

Leggi le note di rilascio della libreria di supporto Android 23.2 per maggiori informazioni.

NOTA: Anche con *AppCompat* , i vettoriali Drawable non funzioneranno al di fuori della tua app nelle versioni precedenti di Android. Ad esempio, non è possibile passare i vettoriali drawable come icone di notifica man mano che vengono gestiti dal sistema e non dall'app. Vedi questa risposta per una soluzione.

Examples

Esempio di utilizzo di VectorDrawable

Ecco un asset vettoriale di esempio che stiamo effettivamente utilizzando in 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 questo drawable, una dichiarazione `ImageView` esempio potrebbe essere:

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@drawable/ic_search"/>
```

Puoi anche impostarlo in fase di esecuzione:

```
ImageView iv = (ImageView) findViewById(...);
iv.setImageResource(R.drawable.ic_search);
```

Lo stesso attributo e le chiamate funzionano anche per `ImageButton` .

Esempio di VectorDrawable xml

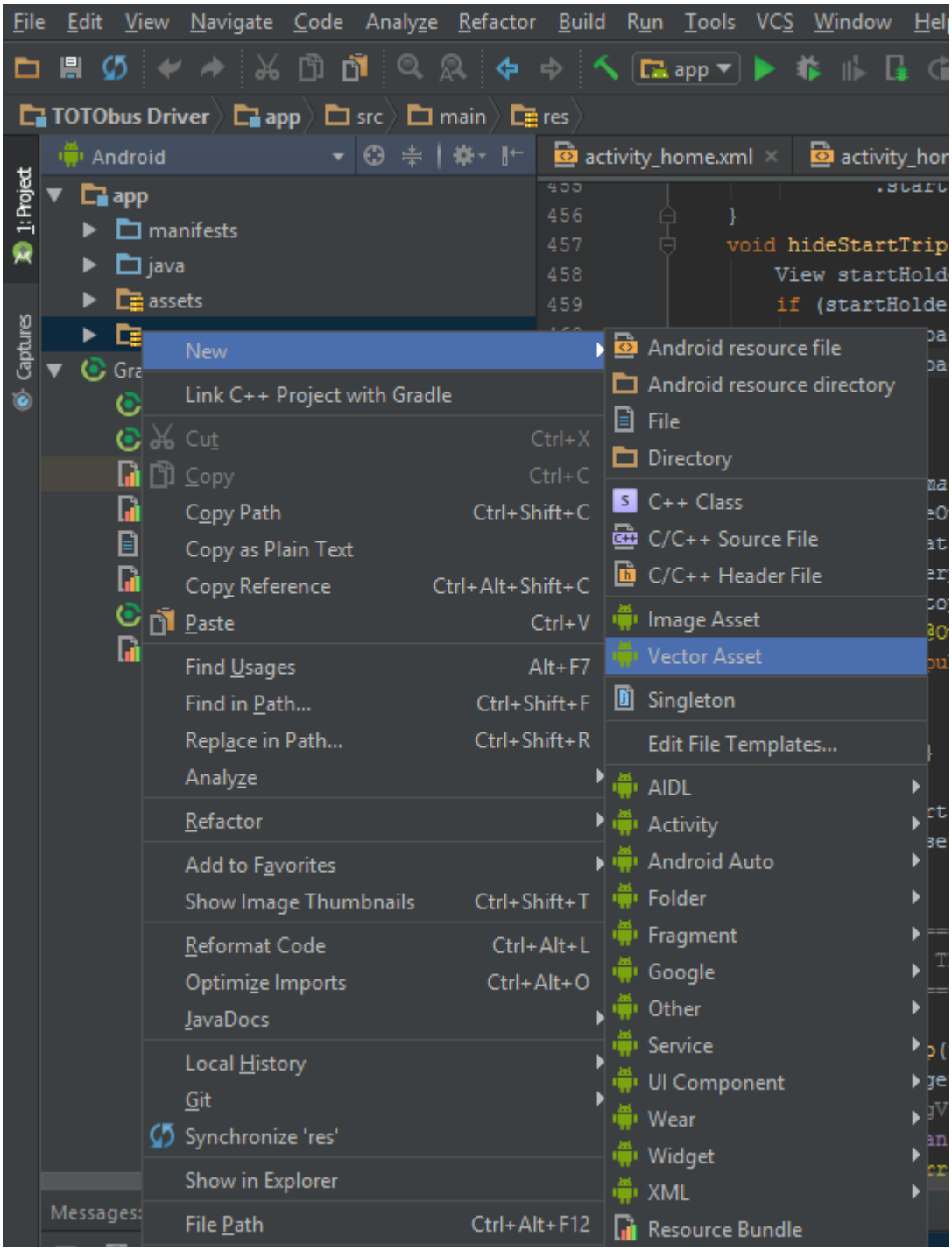
Ecco un semplice `VectorDrawable` in questo file **vectordrawable.xml** .

```
<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>
```

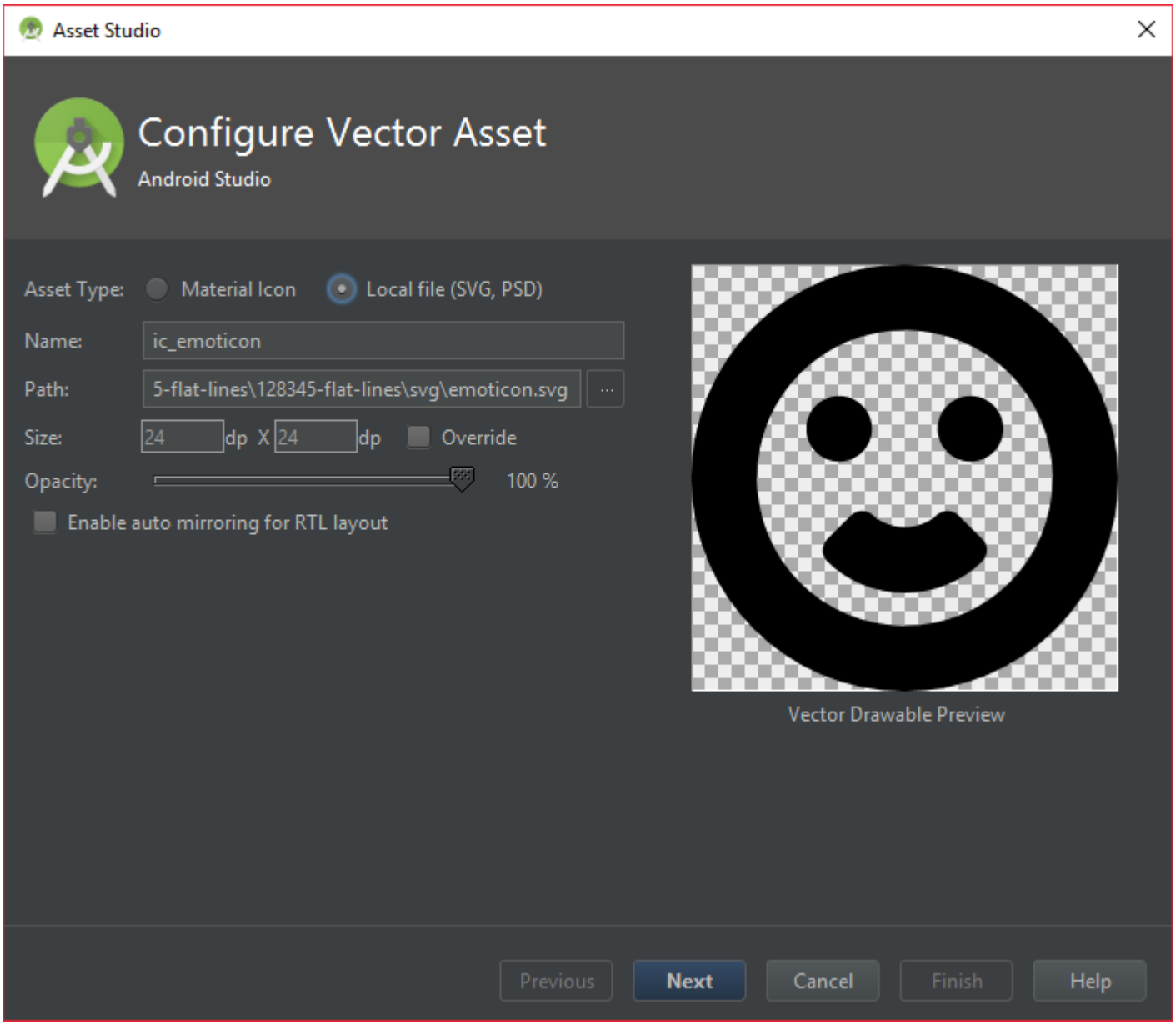
Importazione del file SVG come VectorDrawable

È possibile importare un file *SVG* come `VectorDrawable` in Android Studio, attenersi alla seguente procedura:

"Fai clic con il tasto destro del mouse" nella cartella `res` e seleziona **new** > **Vector Asset** .



Seleziona l'opzione **File locale** e cerca il tuo file .svg. Cambia le opzioni a tuo piacimento e premi avanti. Fatto.



Leggi Disegni vettoriali online: https://riptutorial.com/it/android/topic/8194/disegni-vettoriali

Capitolo 84: Disegno su tela con SurfaceView

Osservazioni

È importante comprendere il concetto di base della vista di superficie prima di utilizzare:

- In pratica è solo un buco nella finestra corrente
- L'interfaccia utente nativa può essere posizionata sopra di essa
- Il disegno viene eseguito utilizzando un thread dedicato, non dell'interfaccia utente
- Il disegno non è accelerato dall'hardware
- Utilizza due buffer: uno è attualmente mostrato, uno è utilizzato per disegnare.
- `unlockCanvasAndPost()` scambia i buffer.

I deadlock possono verificarsi facilmente se i `lockCanvas()` e `unlockCanvasAndPost()` non vengono richiamati nell'ordine corretto.

Examples

SurfaceView con il disegno del filo

Questo esempio descrive come creare un SurfaceView con un thread di disegno dedicato. Questa implementazione gestisce anche casi limite come problemi specifici di produzione e avvio / arresto del thread per risparmiare tempo 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");
    }
}
```

Questo layout contiene solo il SurfaceView personalizzato e lo ingrandisce alle dimensioni dello schermo.

```
<?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>
```

L'attività che utilizza SurfaceView è responsabile dell'avvio e dell'arresto del thread di disegno. Questo approccio consente di risparmiare batteria mentre il disegno viene interrotto non appena l'attività viene messa in background.

```
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();
    }
}
```

Leggi Disegno su tela con SurfaceView online: https://riptutorial.com/it/android/topic/3754/disegno-su-tela-con-surfaceview

Capitolo 85: Dividi schermo / Attività multischermo

Examples

Schermo diviso introdotto in Android Nougat implementato.

Imposta questo attributo nel tuo manifest o elemento per abilitare o disabilitare la visualizzazione multi-finestra:

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

Se questo attributo è impostato su true, l'attività può essere avviata in modalità a schermo diviso e in modalità libera. Se l'attributo è impostato su false, l'attività non supporta la modalità multi-finestra. Se questo valore è falso e l'utente tenta di avviare l'attività in modalità multi-finestra, l'attività viene eseguita a schermo intero.

Se la tua app ha come target il livello di API 24, ma non specifichi un valore per questo attributo, il valore dell'attributo è impostato su true.

Il codice seguente mostra come specificare le dimensioni e la posizione predefinite di un'attività e le dimensioni minime, quando l'attività viene visualizzata in modalità a mano libera:

```
<--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>
```

Funzionalità disattivate in modalità multi-finestra

Alcune funzioni sono disabilitate o ignorate quando un dispositivo è in modalità multi-finestra, perché non hanno senso per un'attività che potrebbe condividere la schermata del dispositivo con altre attività o app. Tali caratteristiche includono:

1. Alcune opzioni di personalizzazione dell'interfaccia utente di sistema sono disabilitate; ad esempio, le app non possono nascondere la barra di stato se non sono in esecuzione in modalità a schermo intero.

2. Il sistema ignora le modifiche ad **Android:** attributo **screenOrientation** .

Se la tua app ha come target il livello API 23 o inferiore

Se la tua app raggiunge il livello API 23 o inferiore e l'utente tenta di utilizzare l'app in modalità

multi-finestra, il sistema ridimensiona forzatamente l'app a meno che l'app non indichi un orientamento fisso.

Se la tua app non dichiara un orientamento fisso, devi avviare la tua app su un dispositivo con Android 7.0 o versioni successive e provare a mettere l'app in modalità schermo condiviso. Verifica che l'esperienza utente sia accettabile quando l'app viene ridimensionata forzatamente.

Se l'app dichiara un orientamento fisso, dovresti provare a mettere l'app in modalità multi-finestra. Verifica che quando lo fai, l'app rimane in modalità a schermo intero.

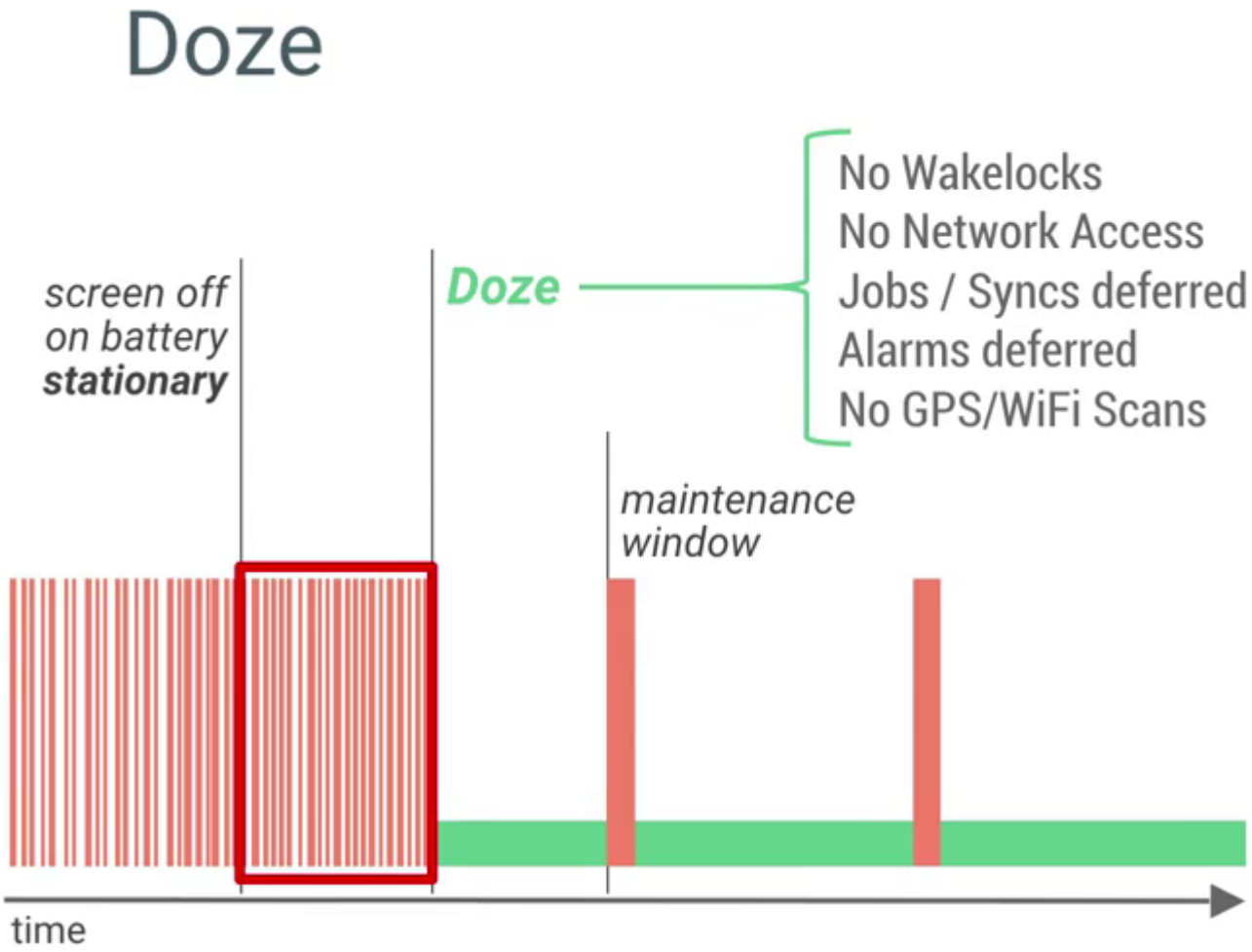
Leggi Dividi schermo / Attività multischermo online:
https://riptutorial.com/it/android/topic/7130/dividi-schermo---attivita-multischermo

Capitolo 86: Doze Mode

Osservazioni

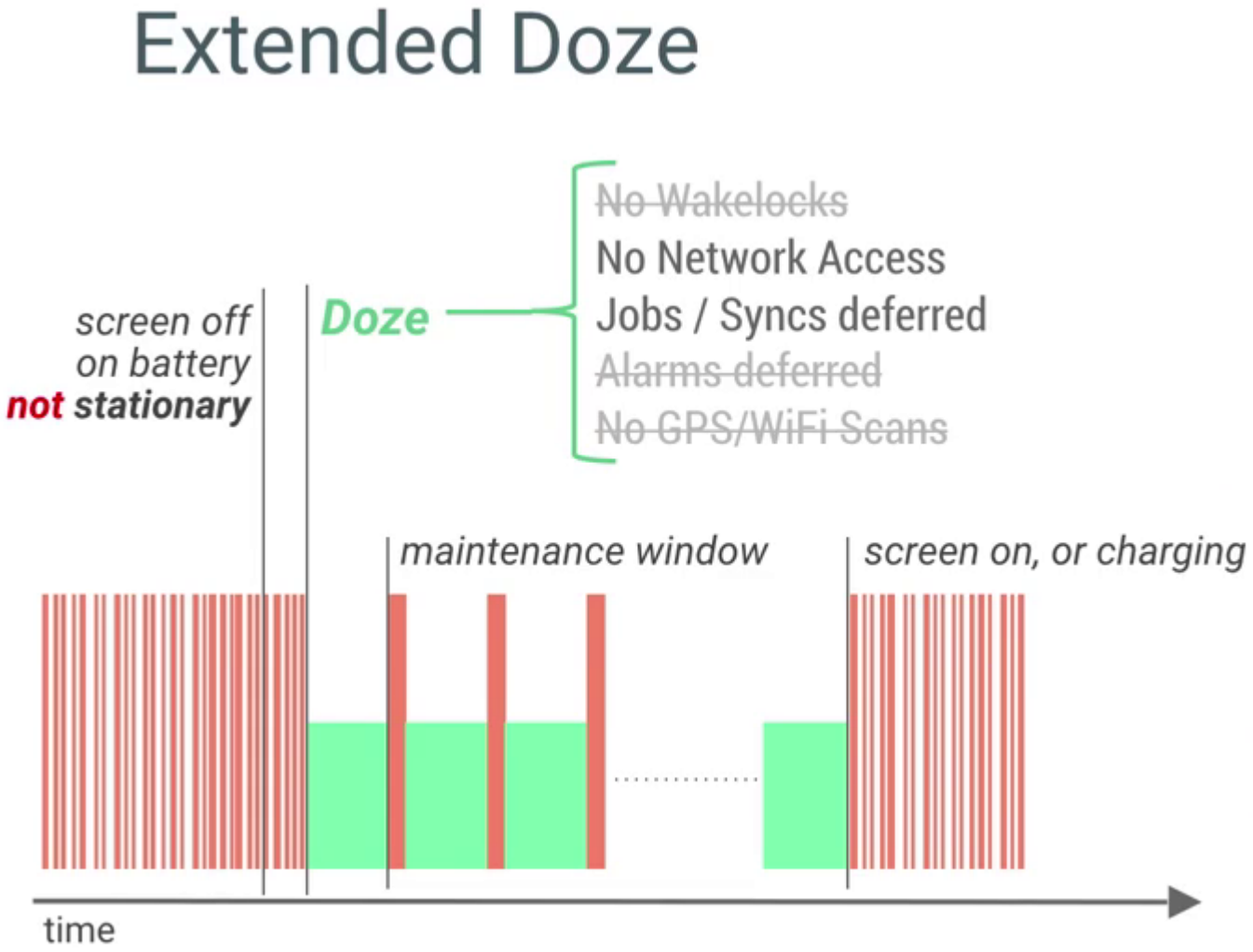
Doze Mode è un insieme di modifiche e regole che mettono il telefono in stop quando inattivo.

Su Android 6.0 Marshmallow: la modalità Doze viene attivata dopo un po 'di tempo, lo schermo è spento, il dispositivo è fermo e funziona a batteria.



Come puoi vedere nello schema sopra, quando la Modalità Doze viene attivata, il dispositivo non riceve alcun wakelock, accesso alla rete, job / sync, allarmi, scansioni GPS / Wi-fi.

Su Android 7.0 Nougat: immagina se il tuo telefono è in tasca (lo schermo è spento, funziona a batteria, ma non è fermo) potresti voler ottenere anche le funzionalità della modalità Doze, giusto? Ecco perché Google ha annunciato la modalità Doze estesa: viene eseguita quando lo schermo è spento, ma non fermo.



Come puoi vedere in questo diagramma, solo Accesso alla rete e lavori / sincronizzazioni sono disabilitati. Si noti che il Doze esteso non sostituisce la prima modalità Doze. Lavorano insieme, a seconda dello stato del telefono (fisso o non). Ecco le distinzioni:

	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

Gli sviluppatori dovrebbero essere consapevoli che:

- Doze potrebbe mantenere il wakelock temporaneo e l'accesso alla rete per i messaggi GCM ad alta priorità (Google Cloud Messaging) (per i casi in cui l'utente necessita di una notifica immediata);
- I servizi in primo piano (come una riproduzione musicale) continueranno a funzionare.

Puoi trovare maggiori informazioni qui: https://developer.android.com/training/monitoring-device-state/doze-standby.html

Examples

Escludi app dall'uso della modalità doze

1. Apri le impostazioni del telefono
2. apri la batteria
3. apri il menu e seleziona "ottimizzazione della batteria"
4. dal menu a discesa seleziona "tutte le app"
5. seleziona l'app che vuoi autorizzare
6. seleziona "non ottimizzare"

Ora questa app mostrerà le app non ottimizzate.

`isIgnoringBatteryOptimizations()` può controllare se è autorizzata chiamando `isIgnoringBatteryOptimizations()`

Whitelist di un'applicazione Android a livello di programmazione

La whitelist non disabilita la modalità doze per la tua app, ma puoi farlo usando i blocchi di rete e di attesa.

La whitelist di un'applicazione Android programmaticamente può essere eseguita come segue:

```
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);
}
```

Il risultato dell'avvio dell'attività sopra può essere verificato dal seguente codice:

```
@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
        }
    }
}
```

Leggi Doze Mode online: https://riptutorial.com/it/android/topic/4719/doze-mode

Capitolo 87: drawable

Examples

Tinta un drawable

Un drawable può essere colorato di un certo colore. Questo è utile per supportare diversi temi all'interno dell'applicazione e ridurre il numero di file di risorse estraibili.

Utilizzo delle API framework su SDK 21+:

```
Drawable d = context.getDrawable(R.drawable.ic_launcher);
d.setTint(Color.WHITE);
```

Utilizzando la libreria android.support.v4 su 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
```

Si prega di `int color` che il `int color` **non si** riferisce a una risorsa colore, tuttavia non si è limitati a quei colori definiti nella classe 'Colore'. Quando hai un colore definito nel tuo XML che vuoi usare devi prima averne il valore.

È possibile sostituire gli usi di `Color.WHITE` utilizzando i metodi seguenti

Quando si rivolgono alle API precedenti:

```
getResources().getColor(R.color.your_color);
```

O su obiettivi più recenti:

```
ContextCompat.getColor(context, R.color.your_color);
```

Crea vista con angoli arrotondati

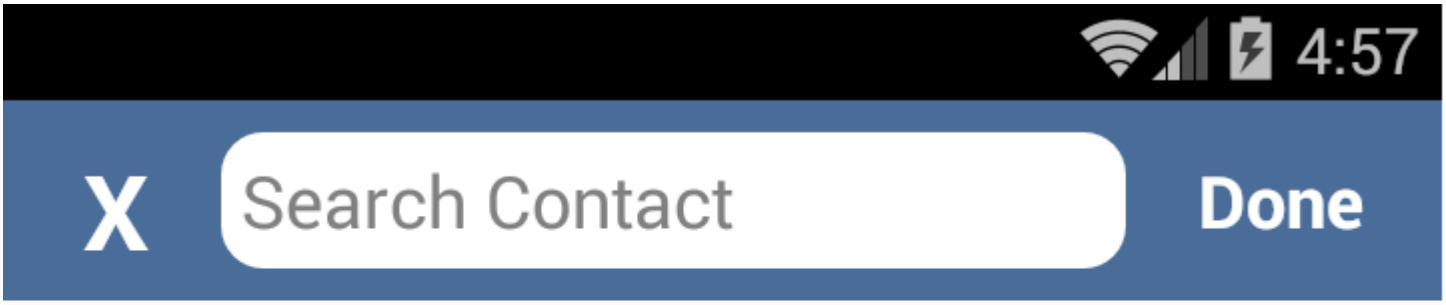
Crea un file **drawable** denominato con **custom_rectangle.xml** nella cartella **drawable** :

```
<?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>
```

Ora applica lo **sfondo del rettangolo** sulla **vista** :

```
mView.setBackGround(R.drawlable.custom_rectangle);
```

Schermata di riferimento:



Vista circolare

Per una vista circolare (in questo caso `TextView`) crea un file **draw_view.xml** nella cartella **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>
```

Assegna il drawable alla 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" />
```

Ora dovrebbe assomigliare al cerchio arancione:

Disegnato su misura

Estendi la tua classe con Drawable e sovrascrivi questi metodi

```
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;
    }
}
```

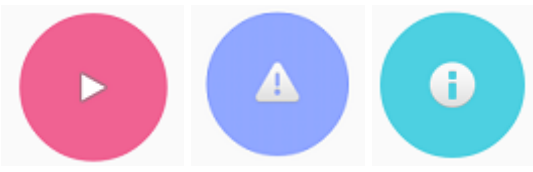
Dichiarare una ImageView nel proprio layout

```
<ImageView
   android:layout_width="80dp"
   android:id="@+id/imageView"
   android:layout_height="80dp" />
```

Imposta il tuo drawable personalizzato su ImageView

```
 IconDrawable iconDrawable=new
IconDrawable(ContextCompat.getDrawable(this,android.R.drawable.ic_media_play),ContextCompat.getColor(th

 imageView.setImageDrawable(iconDrawable);
```

Immagine dello schermo



Leggi drawable online: https://riptutorial.com/it/android/topic/4841/drawable

Capitolo 88: eccezioni

Examples

NetworkOnMainThreadException

Dalla documentazione :

> L'eccezione che viene generata quando un'applicazione tenta di eseguire un'operazione di rete sul thread principale.
>
> Questo viene lanciato solo per applicazioni che hanno come target l'SDK Honeycomb o superiore. Le applicazioni che hanno come target versioni precedenti dell'SDK sono autorizzate a fare networking sui loro thread del ciclo degli eventi principali, ma sono fortemente scoraggiate.

Ecco un esempio di un frammento di codice che può causare quell'eccezione:

```
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);
                }
            }
        }
    }
}
```

Sopra il codice verrà `NetworkOnMainThreadException` per le applicazioni di targeting Honeycomb SDK (Android v3.0) o superiore poiché l'applicazione sta tentando di eseguire un'operazione di rete sul

thread principale.

Per evitare questa eccezione, le operazioni di rete devono sempre essere eseguite in un'attività in background tramite `AsyncTask` , `Thread` , `IntentService` , ecc.

```
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

Questo è un molto comune `Exception` . Fa fermare la tua applicazione durante l'avvio o l'esecuzione della tua app. Nel `LogCat` vedi il messaggio:

```
android.content.ActivityNotFoundException : Unable to find explicit activity class;
have you declared this activity in your AndroidManifest.xml?
```

In questo caso, controlla se hai dichiarato la tua attività nel file `AndroidManifest.xml` .

Il modo più semplice per dichiarare la tua `Activity` in `AndroidManifest.xml` è:

```
<activity  android:name="com.yourdomain.YourStoppedActivity" />
```

OutOfMemoryError

Questo è un errore di runtime che si verifica quando si richiede una grande quantità di memoria

nell'heap. Questo è comune quando si carica un Bitmap in un ImageView.

Hai alcune opzioni:

1. Utilizzare un heap di applicazioni di grandi dimensioni

Aggiungi l'opzione "largeHeap" al tag dell'applicazione nel tuo AndroidManifest.xml. Ciò renderà disponibile più memoria per la tua app, ma probabilmente non risolverà il problema di root.

```
<application largeHeap="true" ... >
```

2. Ricicla i tuoi bitmap

Dopo aver caricato una bitmap, assicurati di riciclarla e liberare memoria:

```
    if (bitmap != null && !bitmap.isRecycled())
       bitmap.recycle();
```

3. Carica bitmap campionati in memoria

Evita di caricare l'intera bitmap in memoria in una sola volta campionando una dimensione ridotta, utilizzando BitmapOptions e inSampleSize.

Vedi la documentazione di Android, per esempio

DexException

```
com.android.dex.DexException: Multiple dex files define Lcom/example/lib/Class;
```

Questo errore si verifica perché l'app, in fase di pacchettizzazione, trova due file `.dex` che definiscono lo stesso insieme di metodi.

Solitamente ciò accade perché l'app ha accidentalmente acquisito 2 dipendenze separate sulla stessa libreria.

Ad esempio, supponiamo di avere un progetto e di affidarti a due librerie: `A` e `B` , ciascuna delle quali ha le sue dipendenze. Se la libreria `B` ha già una dipendenza dalla libreria `A` , questo errore verrà generato se la libreria `A` viene aggiunta al progetto da sola. Compilare la libreria `B` fornisce già il set di codice da `A` , quindi quando il compilatore va a raggruppare la libreria `A` , trova i metodi della libreria `A` già impacchettati.

Per risolvere, assicurati che nessuna delle tue dipendenze possa essere accidentalmente aggiunta due volte in questo modo

Eccezione non rilevata

Se vuoi gestire le eccezioni non rilevate prova a catturarle tutte nel metodo onCreate della tua classe Application:

```
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();
    }
}
```

Registrazione del proprio gestore per eccezioni impreviste

È così che puoi reagire alle eccezioni che non sono state rilevate, in modo simile allo standard del sistema "L'applicazione XYZ si è bloccata"

```
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
    }
}
```

Quindi registra questa classe di applicazione nel tuo AndroidManifest.xml:

```
<application android:name="de.ioxp.arkmobile.MyApplication" >
```

Leggi eccezioni online: https://riptutorial.com/it/android/topic/112/eccezioni

Capitolo 89: Email di convalida

Examples

Convalida dell'indirizzo email

Aggiungi il seguente metodo per verificare se un indirizzo email è valido o meno:

```
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();
}
```

Il metodo sopra può essere facilmente verificato convertendo il testo di un widget `EditText` in 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();
}
```

Convalida dell'indirizzo e-mail con l'utilizzo di Pattern

```
  if (Patterns.EMAIL_ADDRESS.matcher(email).matches()){
       Log.i("EmailCheck","It is valid");
  }
```

Leggi Email di convalida online: https://riptutorial.com/it/android/topic/5605/email-di-convalida

Capitolo 90: Emulatore

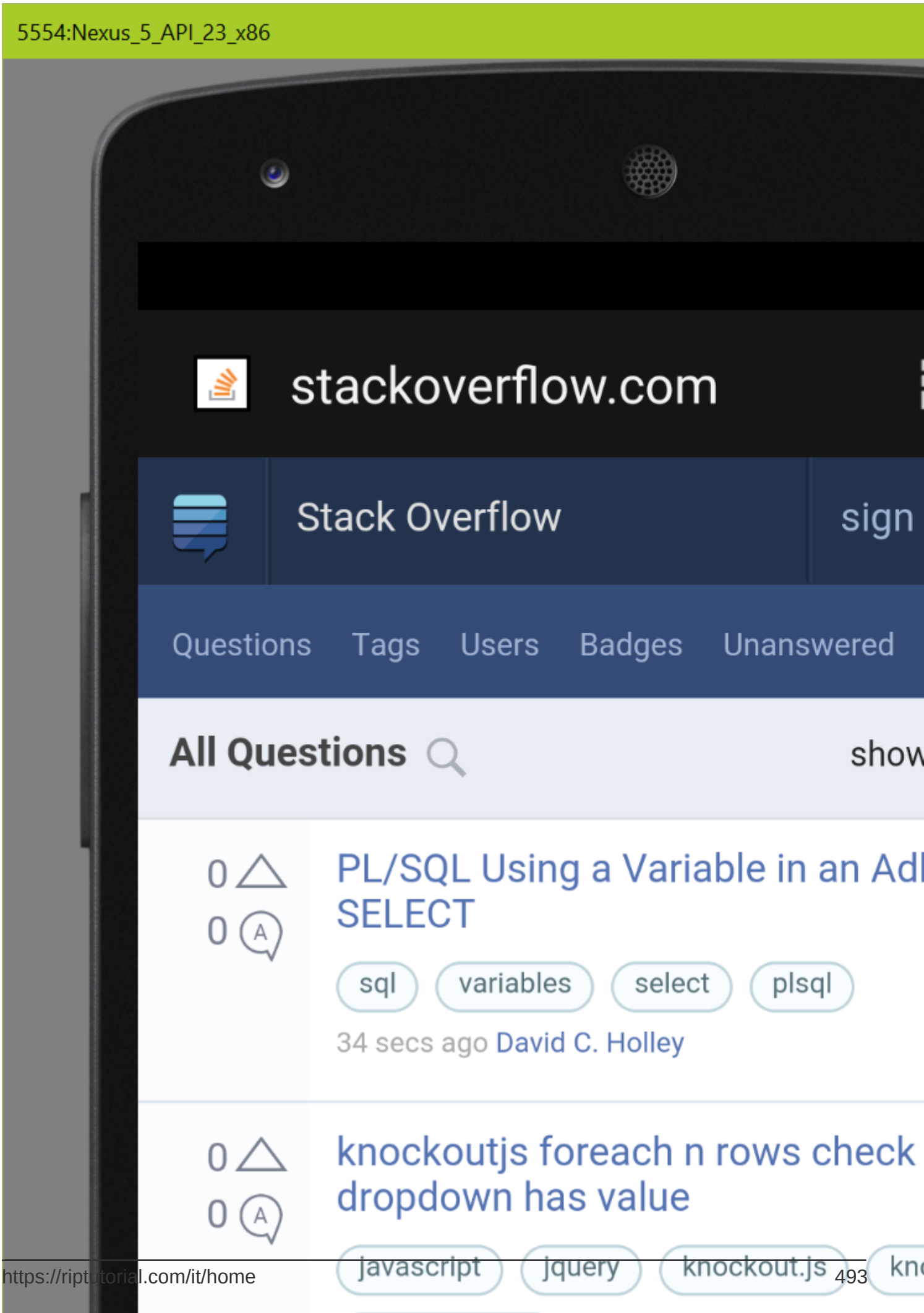
Osservazioni

AVD è l' acronimo di *Dispositivo virtuale Android*

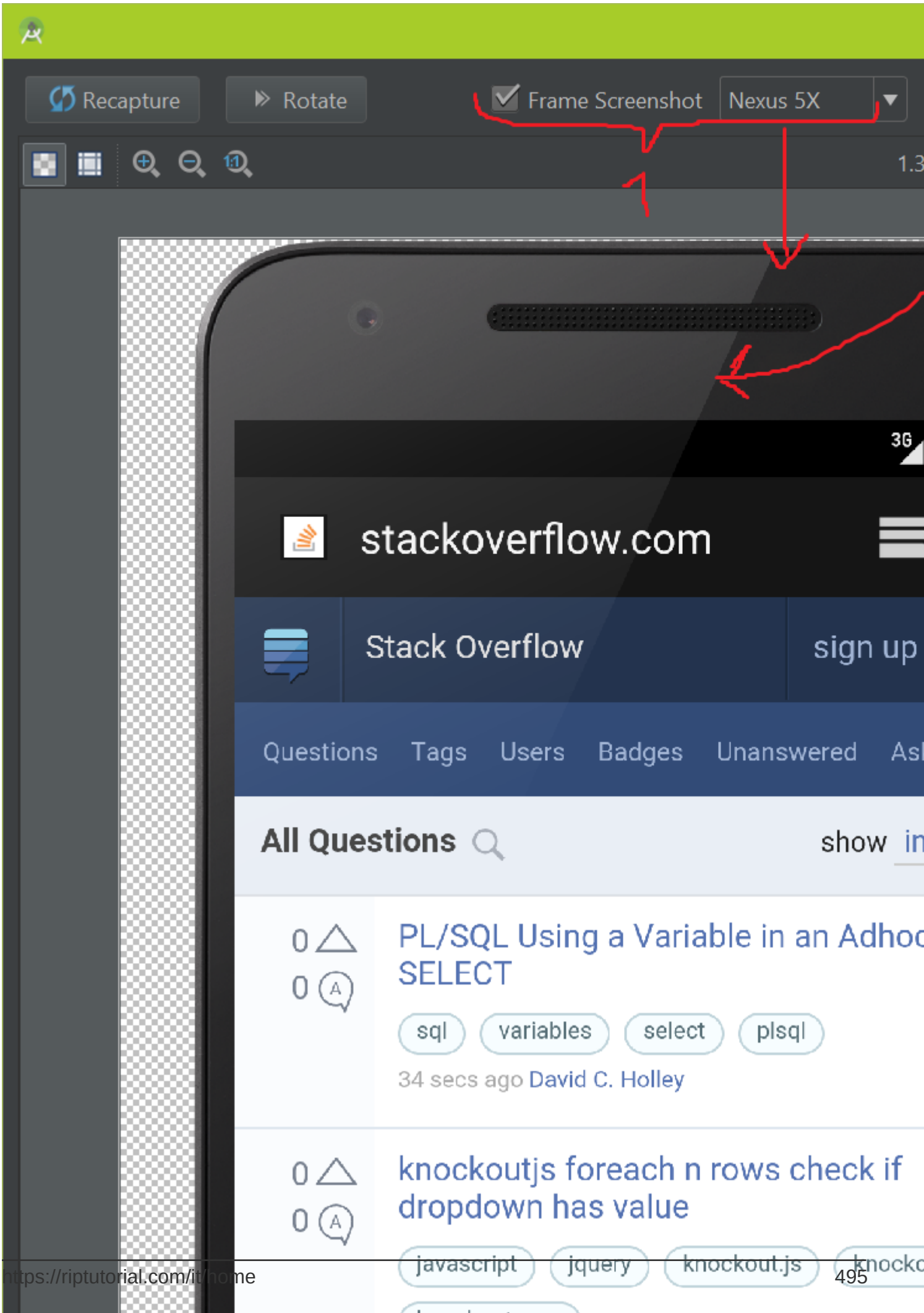
Examples

Prendendo screenshot

Se vuoi fare uno screenshot da Android Emulator (2.0), devi solo premere `Ctrl` + `S` o fare clic sull'icona della fotocamera nella barra laterale:


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. Un'ombra esterna sotto la cornice del dispositivo.
3. Un bagliore dello schermo sul telaio del dispositivo e sullo screenshot.

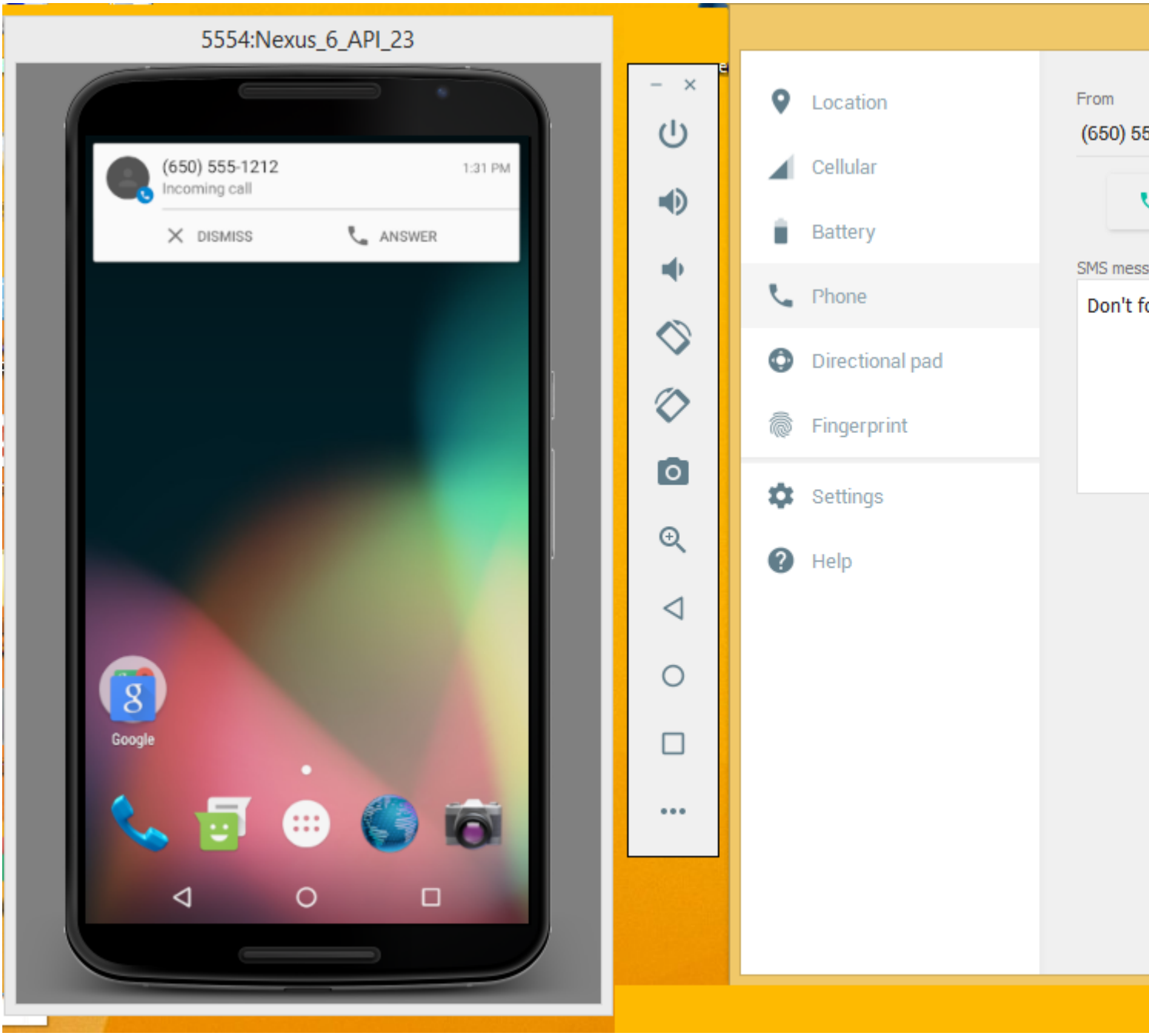

Recapture
Rotate
Frame Screenshot
Nexus 5X
3G
stackoverflow.com
Stack Overflow
sign up
Questions
Tags
Users
Badges
Unanswered
All Questions
show
0
0
PL/SQL Using a Variable in an Adhoc SELECT
sql
variables
select
plsql
34 secs ago David C. Holley
0
0
knockoutjs foreach n rows check if dropdown has value
javascript
jquery
knockout.js

`android avd` .

Puoi accedere ad AVD Manager anche da Android studio usando `Tools` > `Android` > `AVD Manager` o facendo clic sull'icona di AVD Manager nella barra degli strumenti, che è la seconda nella schermata qui sotto.

Simula chiamata

Per simulare una telefonata, premi il pulsante "Controlli estesi" indicato da tre punti, scegli "Telefono" e seleziona "Chiama". È inoltre possibile modificare il numero di telefono facoltativamente.



Risoluzione degli errori durante l'avvio dell'emulatore

Prima di tutto, assicurati di aver abilitato la ' **Virtualizzazione'** nella configurazione del BIOS.

Avviare **Android SDK Manager** , selezionare **Extra,** quindi selezionare **Intel Hardware Accelerated Execution Manager** e attendere il completamento del download. Se continua a non funzionare, apri la cartella SDK ed esegui
`/extras/intel/Hardware_Accelerated_Execution_Manager/IntelHAXM.exe` .

Seguire le istruzioni visualizzate per completare l'installazione.

O per OS X puoi farlo senza prompt su schermo come questo:
`/extras/intel/Hardware_Accelerated_Execution_Manager/HAXM\ installation`

> Se la tua CPU non supporta VT-x o SVM, non puoi usare immagini Android basate su x86. Si prega di utilizzare invece immagini basate su ARM.

Dopo aver completato l'installazione, verificare che il driver di virtualizzazione funzioni correttamente aprendo una finestra del prompt dei comandi ed eseguendo il seguente comando:
`sc query intelhaxm`

Per eseguire un emulatore basato su x86 con accelerazione VM: se si sta eseguendo l'emulatore dalla riga di comando, è sufficiente specificare un AVD basato su x86: `emulator -avd <avd_name>`

Se segui correttamente tutti i passaggi sopra menzionati, sicuramente dovresti essere in grado di vedere il tuo AVD con HAXM in arrivo normalmente.

Leggi Emulatore online: https://riptutorial.com/it/android/topic/122/emulatore

Capitolo 91: Esegui istantaneamente in Android Studio

Osservazioni

L'esecuzione istantanea è un comportamento esteso per i comandi di esecuzione e debug che consente un debugging più rapido non richiedendo una compilazione completa e la reinstallazione di eevry nel codice dell'app.

> Introdotto in Android Studio 2.0, Instant Run è un comportamento per i comandi Esegui e Debug che riduce significativamente il tempo tra gli aggiornamenti alla tua app. Anche se la prima build potrebbe richiedere più tempo per essere completata, Instant Run spinge gli aggiornamenti successivi alla tua app senza creare un nuovo APK, quindi le modifiche sono visibili molto più rapidamente.
>
> L'esecuzione istantanea è supportata solo quando si distribuisce la variante di compilazione di debug, si utilizza Android Plugin per Gradle versione 2.0.0 o successiva e si imposta minSdkVersion su 15 o versione successiva nel file build.gradle a livello di modulo dell'app. Per prestazioni ottimali, imposta minSdkVersion su 21 o versione successiva.
>
> Dopo aver distribuito un'app, viene visualizzata una piccola icona gialla fulmine all'interno del pulsante Esegui (o pulsante Debug), a indicare che Instant Run è pronto per trasmettere gli aggiornamenti la prossima volta che si fa clic sul pulsante. Invece di creare un nuovo APK, spinge solo quelle nuove modifiche e, in alcuni casi, l'app non ha nemmeno bisogno di riavviarsi, ma mostra immediatamente l'effetto di quelle modifiche al codice.
>
> Instant Run spinge codice e risorse aggiornati sul dispositivo o sull'emulatore connesso eseguendo un hot swap, hot swap o cold swap. Determina automaticamente il tipo di swap da eseguire in base al tipo di modifica effettuata. Il video qui sopra fornisce dettagli interessanti su come funziona tutto questo sotto il cofano. Per un rapido riepilogo di come si comporta Instant Run quando si inviano determinate modifiche al codice su un dispositivo di destinazione, vedere la seguente tabella.

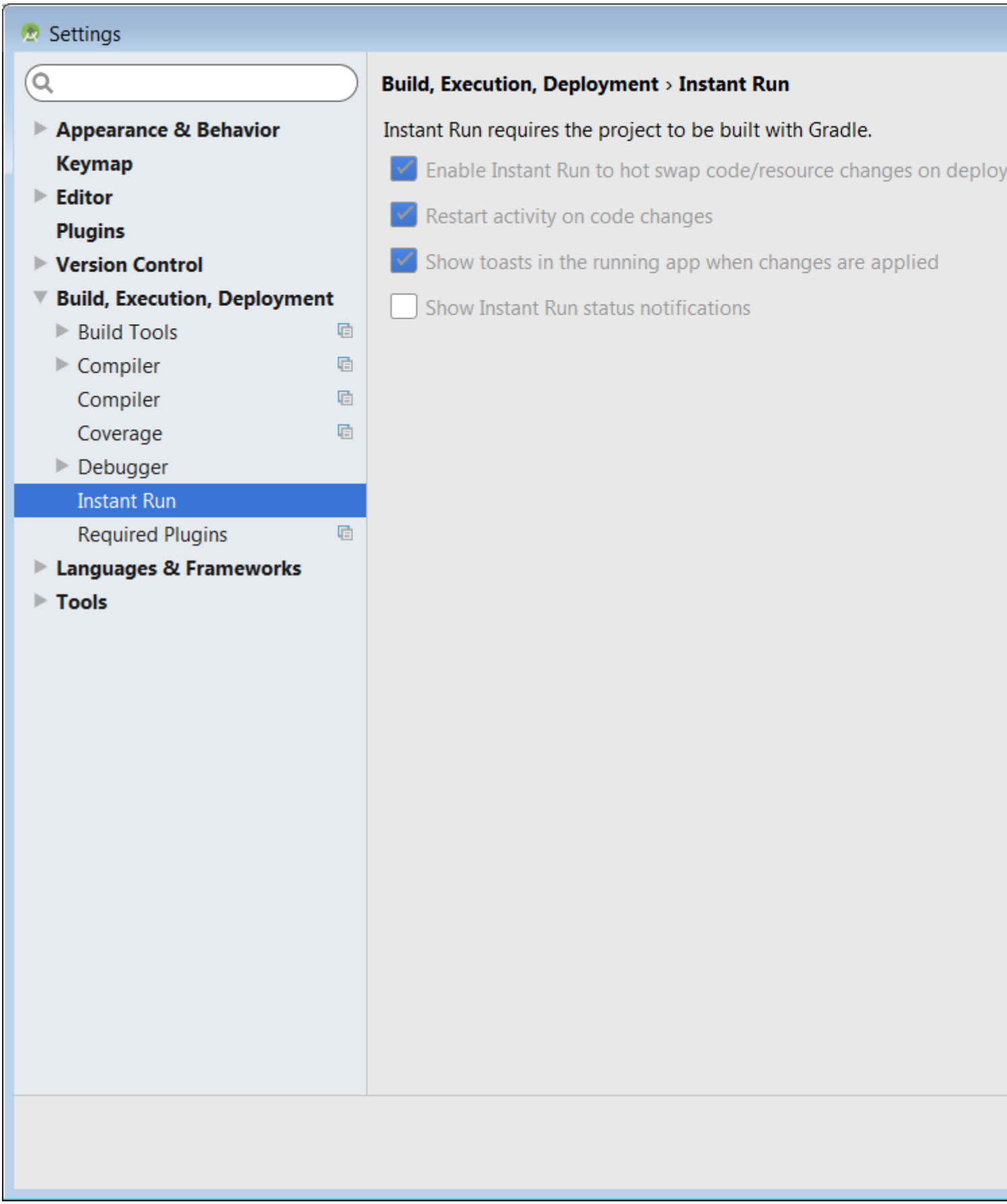
Documentazione

Examples

Abilitazione o disabilitazione di Esecuzione istantanea

1. Apri la finestra di dialogo Impostazioni o Preferenze:
   - Su Windows o Linux, seleziona `File` > `Settings` dal menu principale.
   - Su Mac OSX, seleziona `Android Studio` > `Preferences` dal menu principale.

2. Passare a `Build, Execution, Deployment` > `Compiler` .
3. Nel campo di testo accanto a Opzioni della riga di comando, inserisci le opzioni della riga di comando.
4. Fare clic su OK per salvare e uscire.



L'opzione principale è Instant run. Seleziona / deseleziona quella casella.

Documentazione

Tipi di codice Scambia in esecuzione istantanea

Esistono tre tipi di swap del codice che Instant run consente di supportare il debugging più veloce e l'esecuzione dell'app dal codice in Android Studio.

- Hot Swap
- Scambio caldo
- Cold Swap

Quando vengono attivati ciascuno di questi swap?

HOT SWAP viene attivato quando viene modificata l'implementazione di un metodo esistente.

WARM SWAP viene attivato quando una risorsa esistente viene modificata o rimossa (qualsiasi cosa nella cartella res)

COLD SWAP ogni volta che c'è un cambio di codice strutturale nel codice della tua app es

1. Aggiungi, rimuovi o modifica:

- un'annotazione
- un campo istanza
- un campo statico
- una firma del metodo statico
- una firma del metodo di istanza

2. Cambia quale classe genitrice eredita la classe corrente
3. Cambia l'elenco delle interfacce implementate
4. Cambiare l'inizializzatore statico di una classe
5. Riordina gli elementi del layout che utilizzano gli ID delle risorse dinamiche

Cosa succede quando si verifica uno scambio di codice?

Le modifiche allo **SWOT HOT** sono immediatamente visibili - non appena viene effettuata la prossima chiamata al metodo la cui implementazione è stata modificata.

WARM SWAP riavvia l'attività corrente

COLD SWAP riavvia l'intera app (senza reinstallare)

Cambiamenti di codice non supportati quando si utilizza Esecuzione istantanea

Ci sono alcuni cambiamenti in cui l'instant non farà il suo trucco e una build completa e la reinstallazione della tua app avverranno esattamente come prima che accadesse prima

dell'esecuzione di Instant Run.

1. Cambia il manifest dell'app
2. Cambia risorse a cui fa riferimento il manifest dell'app
3. Modificare un elemento dell'interfaccia utente del widget Android (richiede una pulizia e riesegui)

Documentazione

Leggi Esegui istantaneamente in Android Studio online:
https://riptutorial.com/it/android/topic/2119/esegui-istantaneamente-in-android-studio

Capitolo 92: EventBus di GreenRobot

Sintassi

- @Subscribe (threadMode = ThreadMode.POSTING) public void onEvent (EventClass event) {}

Parametri

Modalità discussione	Descrizione
`ThreadMode.POSTING`	Sarà chiamato sullo stesso thread su cui è stato pubblicato l'evento. Questa è la modalità di default.
`ThreadMode.MAIN`	Sarà chiamato sul thread principale dell'interfaccia utente.
`ThreadMode.BACKGROUND`	Sarà chiamato su un thread in background. Se il thread di pubblicazione non è il thread principale, verrà utilizzato. Se pubblicato sul thread principale, `EventBus` ha un singolo thread in background che utilizzerà.
`ThreadMode.ASYNC`	Sarà chiamato sul proprio thread.

Examples

Creare un oggetto Event

Per inviare e ricevere eventi abbiamo prima bisogno di un oggetto Event. Gli oggetti evento sono in realtà POJO semplici.

```
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;
    }
}
```

Ricevere eventi

Per ricevere eventi è necessario registrare la propria lezione su `EventBus` .

```
@Override
public void onStart() {
    super.onStart();
    EventBus.getDefault().register(this);
}

@Override
public void onStop() {
   EventBus.getDefault().unregister(this);
   super.onStop();
}
```

E poi iscriviti agli eventi.

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void handleEvent(ArbitaryEvent event) {
    Toast.makeText(getActivity(), "Event type: "+event.getEventType(),
Toast.LENGTH_SHORT).show();
}
```

Invio di eventi

L'invio di eventi è facile come creare l'oggetto Event e quindi pubblicarlo.

```
EventBus.getDefault().post(new ArbitaryEvent(ArbitaryEvent.TYPE_1));
```

Passando un semplice evento

La prima cosa che dobbiamo fare è aggiungere EventBus al file gradle del nostro modulo:

```
dependencies {
    ...
    compile 'org.greenrobot:eventbus:3.0.0'
    ...
}
```

Ora dobbiamo creare un modello per il nostro evento. Può contenere tutto ciò che vogliamo passare. Per ora faremo solo una lezione vuota.

```
public class DeviceConnectedEvent
{
}
```

Ora possiamo aggiungere il codice alla nostra `Activity` che si registrerà con EventBus e si iscriverà all'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
    }
}
```

In questa `Activity` ottiene un'istanza di `EventBus` nel metodo `onCreate()` . Registriamo / `onStart()` registrazione per gli eventi in `onStart()` / `onStop()` . È importante ricordare di annullare la registrazione quando l'ascoltatore perde l'ambito o si potrebbe perdere l' `Activity` .

Infine definiamo il metodo che vogliamo chiamare con l'evento. L'annotazione `@Subscribe` indica a EventBus quali metodi può cercare per gestire gli eventi. Devi avere almeno un metodo annotato con `@Subscribe` per registrarti con EventBus o genererà un'eccezione. Nell'annotazione definiamo la modalità thread. Questo dice a EventBus su quale thread chiamare il metodo. È un modo molto pratico per passare informazioni da un thread in background al thread dell'interfaccia utente! Questo è esattamente quello che stiamo facendo qui. `ThreadMode.MAIN` significa che questo metodo verrà chiamato sul thread principale dell'interfaccia utente di Android, quindi è sicuro fare qualsiasi manipolazione dell'interfaccia utente qui di cui hai bisogno. Il nome del metodo non ha importanza. L'unica cosa da pensare, altro che l'annotazione `@Subscribe` , che EventBus sta cercando è il tipo dell'argomento. Finché il tipo corrisponde verrà chiamato quando viene pubblicato un evento.

L'ultima cosa che dobbiamo fare per pubblicare un evento. Questo codice sarà nel nostro `Service` .

```
EventBus.getDefault().post(new DeviceConnectedEvent());
```

Questo è tutto ciò che c'è da fare! EventBus prenderà quel DeviceConnectedEvent e guarderà attraverso i suoi listener registrati, controllerà i metodi che hanno sottoscritto e troverà quelli che prendono un DeviceConnectedEvent come argomento e li chiamano sul thread su cui vogliono essere chiamati.

Leggi EventBus di GreenRobot online: https://riptutorial.com/it/android/topic/3551/eventbus-di-greenrobot

Capitolo 93: ExoPlayer

Examples

Aggiungi ExoPlayer al progetto

Via jCenter

includendo quanto segue nel file build.gradle del tuo progetto:

```
compile 'com.google.android.exoplayer:exoplayer:rX.X.X'
```

dove rX.XX è la tua versione preferita. Per l'ultima versione, vedere il progetto Uscite . Per maggiori dettagli, vedi il progetto su Bintray .

Utilizzando ExoPlayer

Crea un'istanza di ExoPlayer:

```
exoPlayer = ExoPlayer.Factory.newInstance(RENDERER_COUNT, minBufferMs, minRebufferMs);
```

Per riprodurre solo l'audio è possibile utilizzare questi valori:

```
RENDERER_COUNT = 1 //since you want to render simple audio
minBufferMs = 1000
minRebufferMs = 5000
```

Entrambi i valori del buffer possono essere ottimizzati in base alle proprie esigenze.

Ora devi creare un DataSource. Quando vuoi riprodurre lo streaming mp3 puoi usare DefaultUriDataSource. Devi passare il contesto e un utenteAgent. Per semplificare, riprodurre un file locale e passare null come userAgent:

```
DataSource dataSource = new DefaultUriDataSource(context, null);
```

Quindi crea l'origine campione:

```
ExtractorSampleSource sampleSource = new ExtractorSampleSource(
                uri, dataSource, new Mp3Extractor(), RENDERER_COUNT, requestedBufferSize);
```

uri punta al tuo file, come Extractor puoi usare un semplice Mp3Extractor predefinito se vuoi riprodurre mp3. requiredBufferSize può essere modificato di nuovo in base alle tue esigenze. Usa 5000 per esempio.

Ora puoi creare il tuo renderer di traccia audio usando la sorgente di esempio come segue:

```
MediaCodecAudioTrackRenderer audioRenderer = new MediaCodecAudioTrackRenderer(sampleSource);
```

Infine chiama la preparazione sulla tua istanza di exoPlayer:

```
exoPlayer.prepare(audioRenderer);
```

Per avviare la chiamata di riproduzione:

```
exoPlayer.setPlayWhenReady(true);
```

Passi principali per riprodurre video e audio usando le implementazioni standard di 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!
```

Leggi ExoPlayer online: https://riptutorial.com/it/android/topic/6248/exoplayer

Capitolo 94: Facebook SDK per Android

Sintassi

- **newInstance** : per creare una singola istanza della classe helper di Facebook.
- **loginUser** : per accedere all'utente.
- **signOut** : per disconnettersi.
- **getCallbackManager** : per ottenere la richiamata per Facebook.
- **getLoginCallback** : per ottenere la richiamata per l'accesso.
- **getKeyHash** : per generare l'hash della chiave di Facebook.

Parametri

Parametro	Dettagli
ETICHETTA	Una stringa utilizzata durante la registrazione
FacebookSignInHelper	Un riferimento statico all'aiuto di Facebook
CallbackManager	Una richiamata per le operazioni di Facebook
Attività	Un contesto
PERMISSION_LOGIN	Un array che contiene tutte le autorizzazioni richieste da Facebook per accedere.
loginCallback	Una richiamata per l'accesso a Facebook

Examples

Come aggiungere Facebook Login in Android

Aggiungi dipendenze sottostanti al tuo `build.gradle`

```
  // Facebook login
    compile 'com.facebook.android:facebook-android-sdk:4.21.1'
```

Aggiungi sotto la classe helper al tuo pacchetto di utilità:

```
/**
 * 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;
    }
}
```

Aggiungi sotto il codice nella tua attività:

```
FacebookSignInHelper facebookSignInHelper =
FacebookSignInHelper.newInstance(LoginActivity.this, fireBaseAuthHelper);
facebookSignInHelper.loginUser();
```

Aggiungi sotto il codice al tuo `OnActivityResult` :

```
 facebookSignInHelper.getCallbackManager().onActivityResult(requestCode, resultCode, data);
```

Impostazione delle autorizzazioni per accedere ai dati dal profilo di Facebook

Se si desidera recuperare i dettagli del profilo Facebook di un utente, è necessario impostare le autorizzazioni per lo stesso:

```
loginButton = (LoginButton)findViewById(R.id.login_button);

loginButton.setReadPermissions(Arrays.asList("email", "user_about_me"));
```

Puoi continuare ad aggiungere ulteriori autorizzazioni come amici-elenco, post, foto, ecc. Seleziona il permesso giusto e aggiungilo all'elenco precedente.

Nota: non è necessario impostare autorizzazioni esplicite per l'accesso al profilo pubblico (nome, cognome, ID, sesso ecc.).

Crea il tuo pulsante personalizzato per l'accesso a Facebook

Quando aggiungi per la prima volta l'accesso / iscrizione a Facebook, il pulsante assomiglia a qualcosa:

Il più delle volte, non corrisponde alle specifiche di progettazione della tua app. Ed ecco come puoi personalizzarlo:

```
<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>
```

Basta avvolgere l'originale `com.facebook.login.widget.LoginButton` in un `FrameLayout` e renderne visibile la visibilità.

Successivamente, aggiungi il tuo pulsante personalizzato nello stesso `FrameLayout` . Ho aggiunto alcune specifiche di esempio. Puoi sempre creare il tuo sfondo disegnabile per il pulsante facebook e impostarlo come sfondo del pulsante.

L'ultima cosa che facciamo è semplicemente convertire il clic sul mio pulsante personalizzato in un clic sul pulsante facecbook:

```
//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();
    }
}
```

Grande! Ora il pulsante ha un aspetto simile al seguente:

Una guida minimalistica all'implementazione di accesso / registrazione di Facebook

1. Devi impostare i prerequisiti .

2. Aggiungi l'attività di Facebook al file *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. Aggiungi il pulsante di accesso al tuo file XML di layout:

   ```
   <com.facebook.login.widget.LoginButton
       android:id="@+id/login_button"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content" />
   ```

4. Ora hai il pulsante Facebook. Se l'utente fa clic su di esso, la finestra di accesso di Facebook comparirà in cima allo schermo dell'app. Qui l'utente può inserire le proprie credenziali e premere il pulsante *Accedi* . Se le credenziali sono corrette, la finestra di dialogo concede le autorizzazioni corrispondenti e una richiamata viene inviata all'attività originale contenente il pulsante. Il codice seguente mostra come puoi ricevere quella richiamata:

   ```
   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.
    }
});
```

Disconnessione da Facebook

Facebook SDK 4.0 in poi, questo è il modo in cui effettuiamo il logout:

```
com.facebook.login.LoginManager.getInstance().logOut();
```

Per le versioni precedenti alla 4.0, la disconnessione è terminata cancellando esplicitamente il token di accesso:

```
Session session = Session.getActiveSession();
session.closeAndClearTokenInformation();
```

Leggi Facebook SDK per Android online: https://riptutorial.com/it/android/topic/3919/facebook-sdk-per-android

Capitolo 95: Fastjson

introduzione

Fastjson è una libreria Java che può essere utilizzata per convertire oggetti Java nella loro rappresentazione JSON. Può anche essere utilizzato per convertire una stringa JSON in un oggetto Java equivalente.

Caratteristiche di Fastjson:

Fornisci le migliori prestazioni sul lato server e sul client Android

Fornire `toJSONString()` semplici `toJSONString()` e `parseObject()` per convertire oggetti Java in JSON e viceversa

Consenti agli oggetti non modificabili preesistenti di essere convertiti da e verso JSON

Ampio supporto di Java Generics

Sintassi

- Object parse (testo stringa)
- JSONObject parseObject (Testo stringa)
- T parseObject (Testo stringa, Classe <T> clazz)
- JSONArray parseArray (testo stringa)
- <T> Elenco <T> parseArray (Testo stringa, Classe <T> clazz)
- String toJSONString (oggetto Object)
- String toJSONString (oggetto Object, booleano prettyFormat)
- Object toJSON (Object javaObject)

Examples

Parsing JSON con Fastjson

Puoi guardare l'esempio nella libreria Fastjson

Codificare

```
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);
```

Produzione

```
{"id":0,"name":"admin","users":[{"id":2,"name":"guest"},{"id":3,"name":"root"}]}
```

Decodificare

```
String jsonString = ...;
Group group = JSON.parseObject(jsonString, Group.class);
```

Group.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);
    }
}
```

User.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;
    }
}
```

Converti i dati del tipo Map in stringa JSON

Codice

```
    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);
```

Produzione

```
{1:"No.1",2:"No.2",3:[{"id":2,"name":"Liu"},{"id":3,"name":"Yue"}]}
```

Leggi Fastjson online: https://riptutorial.com/it/android/topic/10865/fastjson

Capitolo 96: Fatturazione in-app

Examples

Consumabili acquisti in-app

I prodotti gestiti consumabili sono prodotti che possono essere acquistati più volte, ad esempio valuta del gioco, vite del gioco, power-up, ecc.

In questo esempio, implementeremo 4 diversi prodotti **gestiti** consumabili `"item1", "item2", "item3", "item4"` .

Passaggi in sintesi:

1. Aggiungi la libreria di fatturazione in-app al tuo progetto (file AIDL).
2. Aggiungi l'autorizzazione richiesta nel file `AndroidManifest.xml` .
3. Distribuisci un apk firmato a Google Developers Console.
4. Definisci i tuoi prodotti.
5. Implementa il codice.
6. Prova la fatturazione in-app (opzionale).

Passo 1:

Prima di tutto, dovremo aggiungere il file AIDL al tuo progetto come spiegato chiaramente nella documentazione di Google qui .

`IInAppBillingService.aidl` è un file AIDL (Android Interface Definition Language) che definisce l'interfaccia al servizio di fatturazione in-app versione 3. Utilizzerai questa interfaccia per fare richieste di fatturazione invocando le chiamate al metodo IPC.

Passo 2:

Dopo aver aggiunto il file AIDL, aggiungi l'autorizzazione BILLING in `AndroidManifest.xml` :

```
<!-- Required permission for implementing In-app Billing -->
<uses-permission android:name="com.android.vending.BILLING" />
```

Passaggio 3:

Genera un apk firmato e caricalo su Google Developers Console. Questo è necessario per iniziare a definire i nostri prodotti in-app.

Passaggio 4:

Definisci tutti i tuoi prodotti con ID prodotto diverso e imposta un prezzo per ognuno di essi. Esistono 2 tipi di prodotti (Prodotti gestiti e Abbonamenti). Come già detto, implementeremo 4 diversi prodotti **gestibili** consumabili `"item1", "item2", "item3", "item4"` .

Passaggio 5:

Dopo aver eseguito tutti i passaggi precedenti, sei pronto per iniziare a implementare il codice stesso nella tua attività.

Attività principale:

```
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();
        }
    }
}
```

Passaggio 6:

Dopo aver implementato il codice, puoi testarlo distribuendo l'apk al canale beta / alfa e consentire ad altri utenti di testare il codice per te. Tuttavia, non è possibile effettuare veri acquisti in-app mentre ci si trova in modalità di test. Devi pubblicare la tua app / gioco prima su Play Store in

modo che tutti i prodotti siano completamente attivati.

Maggiori informazioni sulla verifica della fatturazione in-app possono essere trovate qui .

(Terze parti) Libreria in-app v3

Passaggio 1: Prima di tutto, segui questi due passaggi per aggiungere funzionalità dell'app:

1. Aggiungi la libreria usando:

```
repositories {
           mavenCentral()
       }
       dependencies {
          compile 'com.anjlab.android.iab.v3:library:1.0.+'
       }
```

2. Aggiungi permesso nel file manifest.

```
<uses-permission android:name="com.android.vending.BILLING" />
```

Passaggio 2: inizializza il tuo processore di fatturazione:

```
BillingProcessor bp = new BillingProcessor(this, "YOUR LICENSE KEY FROM GOOGLE PLAY CONSOLE
HERE", this);
```

e implementare Billing Handler: BillingProcessor.IBillingHandler che contiene 4 metodi: a. onBillingInitialized (); b. onProductPurchased (String productId, TransactionDetails details): qui è dove devi gestire le azioni da eseguire dopo l'acquisto riuscito c. onBillingError (int errorCode, Throwable error): gestisce qualsiasi errore si è verificato durante il processo di acquisto d. onPurchaseHistoryRestored (): per ripristinare gli acquisti di app

Passaggio 3: come acquistare un prodotto.

Per acquistare un prodotto gestito:

```
bp.purchase(YOUR_ACTIVITY, "YOUR PRODUCT ID FROM GOOGLE PLAY CONSOLE HERE");
```

E per acquistare un abbonamento:

```
bp.subscribe(YOUR_ACTIVITY, "YOUR SUBSCRIPTION ID FROM GOOGLE PLAY CONSOLE HERE");
```

Passaggio 4: consumo di un prodotto.

Per consumare un prodotto è sufficiente chiamare il metodo consumePurchase.

bp.consumePurchase ("IL TUO ID PRODOTTO DALLA CONSOLE DI GOOGLE PLAY QUI");

Per altri metodi relativi a in github visita app

Leggi Fatturazione in-app online: https://riptutorial.com/it/android/topic/2843/fatturazione-in-app

Capitolo 97: File zip in Android

Examples

File zip su 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();
        }

      }

    }
```

Leggi File zip in Android online: https://riptutorial.com/it/android/topic/8137/file-zip-in-android

Capitolo 98: FileIO con Android

introduzione

La lettura e la scrittura di file in Android non sono diversi dalla lettura e dalla scrittura di file in Java standard. `java.io` possibile utilizzare lo stesso pacchetto `java.io` Tuttavia, vi sono alcune specifiche relative alle cartelle in cui è consentito scrivere, le autorizzazioni in generale e MTP funzionano.

Osservazioni

Android fornisce i mezzi per condividere il file tra più applicazioni come documentato qui . Questo non è richiesto se esiste una sola app che crea e utilizza il file.

Android offre opzioni di archiviazione alternative come le preferenze condivise e private, i pacchetti salvati e il database integrato. In alcuni casi, sono una scelta migliore rispetto all'utilizzo di semplici file.

L'attività Android ha pochi metodi specifici che assomigliano a sostituzioni dei metodi di File IO standard di Java. Ad esempio, invece per `File.delete()` puoi chiamare `Context.deleteFile()` , e invece di applicare `File.listFiles()` modo ricorsivo puoi chiamare `Context.fileList()` per ottenere l'elenco di tutti i tuoi file specifici dell'app con un po 'meno codice. Tuttavia, non forniscono funzionalità aggiuntive oltre `java.io` pacchetto `java.io` standard.

Examples

Ottenere la cartella di lavoro

Puoi ottenere la tua cartella di lavoro chiamando il metodo `getFilesDir()` sulla tua attività (Activity è la classe centrale nella tua applicazione che eredita dal contesto. Vedi qui). La lettura non è diversa. Solo la tua applicazione avrà accesso a questa cartella.

La tua attività potrebbe contenere il seguente codice, ad esempio:

```
File myFolder = getFilesDir();
File myFile = new File(myFolder, "myData.bin");
```

Scrivere una matrice di byte grezza

```
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()
```

Non c'è nulla di specifico per Android con questo codice. Se si scrivono spesso molti piccoli valori, utilizzare BufferedOutputStream per ridurre l'usura dell'unità SSD interna del dispositivo.

Serializzare l'oggetto

La vecchia seria serializzazione di oggetti Java è disponibile per te su Android. puoi definire classi serializzabili come:

```
class Cirle implements Serializable {
  final int radius;
  final String name;

  Circle(int radius, int name) {
    this.radius = radius;
    this.name = name;
  }
}
```

e quindi scrivere su 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 serializzazione degli oggetti Java può essere una scelta perfetta o davvero pessima, a seconda di cosa si vuole fare con esso - al di fuori dello scopo di questo tutorial ea volte basato sull'opinione. Leggi prima il versioning se decidi di usarlo.

Scrittura su memoria esterna (scheda SD)

È inoltre possibile leggere e scrivere da / su una scheda di memoria (scheda SD) presente in molti dispositivi Android. I file in questa posizione sono accessibili da altri programmi, anche direttamente dall'utente dopo aver collegato il dispositivo al PC tramite cavo USB e abilitato il protocollo MTP.

Trovare la posizione della scheda SD è un po 'più problematico. La classe Environment contiene metodi statici per ottenere "directory esterne" che dovrebbero normalmente trovarsi all'interno della scheda SD, anche le informazioni se la scheda SD esiste del tutto ed è scrivibile. Questa domanda contiene risposte preziose su come accertarsi di trovare la giusta posizione.

L'accesso alla memoria esterna richiede permessi nel tuo manifest Android:

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

Per le versioni precedenti di Android che inseriscono i permessi è sufficiente mettere queste

autorizzazioni in manifest (l'utente deve approvare durante l'installazione). Tuttavia, a partire da Android 6.0 Android chiede l'approvazione dell'utente al momento del primo accesso e devi supportare questo nuovo approccio. In caso contrario, l'accesso è negato indipendentemente dal tuo manifest.

In Android 6.0, per prima cosa è necessario verificare l'autorizzazione, quindi, se non concesso, richiederlo. Gli esempi di codice possono essere trovati all'interno di questa domanda SO .

Risolvere il problema "File invisibili MTP".

Se si creano file per l'esportazione tramite cavo USB sul desktop usando il protocollo MTP, potrebbe essere un problema che i file appena creati non siano immediatamente visibili nel file explorer in esecuzione sul PC desktop connesso. Per rendere visibili i nuovi file, è necessario chiamare 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)));
```

Questo codice di chiamata MediaScannerConnection funziona solo per i file, non per le directory. Il problema è descritto in questo bug report di Android . Questo potrebbe essere risolto per alcune versioni in futuro o su alcuni dispositivi.

Lavorare con file di grandi dimensioni

I file piccoli vengono elaborati in una frazione di secondo e puoi leggerli / scriverli al posto del codice in cui ti serve. Tuttavia, se il file è più grande o più lento da elaborare, potrebbe essere necessario utilizzare AsyncTask in Android per lavorare con il file in background:

```
    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
        }
    }
}
```

e poi

```
new FileOperation().execute("Some parameters");
```

Questa domanda SO contiene l'esempio completo su come creare e chiamare AsyncTask. Vedi anche la domanda sulla gestione degli errori su come gestire IOExceptions e altri errori.

Leggi FileIO con Android online: https://riptutorial.com/it/android/topic/8689/fileio-con-android

Capitolo 99: FileProvider

Examples

Condivisione di un file

In questo esempio imparerai come condividere un file con altre app. In questo esempio useremo un file pdf anche se il codice funziona anche con ogni altro formato.

La tabella di marcia:

Specificare le directory in cui sono posizionati i file che si desidera condividere

Per condividere file utilizzeremo un FileProvider, una classe che consente la condivisione sicura dei file tra le app. Un FileProvider può solo condividere file in directory predefinite, quindi definiamoli.

1. Creare un nuovo file XML che conterrà i percorsi, ad esempio *res / xml / filepaths.xml*
2. Aggiungi i percorsi

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <files-path name="pdf_folder" path="documents/"/>
</paths>
```

Definire un FileProvider e collegarlo con i percorsi dei file

Questo è fatto nel manifest:

```
<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>
```

Genera l'URI per il file

Per condividere il file dobbiamo fornire un identificatore per il file. Questo viene fatto usando un URI (Uniform Resource Identifier).

```
// 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);
```

Come puoi vedere nel codice, prima creiamo una nuova classe File che rappresenta il file. Per ottenere un URI chiediamo a FileProvider di ottenerne uno. Il secondo argomento è importante: passa l'autorità di un FileProvider. Deve essere uguale all'autorità del FileProvider definita nel manifest.

Condividi il file con altre app

Usiamo ShareCompat per condividere il file con altre app:

```
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 selettore è un menu da cui l'utente può scegliere con quale app desidera condividere il file. Il flag Intent.FLAG_GRANT_READ_URI_PERMISSION è necessario per concedere l'autorizzazione di accesso in lettura temporanea all'URI.

Leggi FileProvider online: https://riptutorial.com/it/android/topic/6266/fileprovider

Capitolo 100: Filo

Examples

Esempio di discussione con la sua descrizione

Durante l'avvio dell'applicazione, viene eseguito innanzitutto il thread principale. Questo thread principale gestisce tutto il concetto di applicazione dell'interfaccia utente. Se vogliamo eseguire a lungo l'attività in cui non abbiamo bisogno dell'interfaccia utente, usiamo thread per eseguire quell'attività in background.

Ecco l'esempio di Thread che descrive blow:

```
new Thread(new Runnable() {
    public void run() {
        for(int i = 1; i < 5;i++) {
            System.out.println(i);
        }
    }
}).start();
```

Possiamo creare thread creando l'oggetto di Thread che ha il metodo `Thread.run()` per eseguire il thread. Il metodo `run()` viene chiamato dal metodo `start()` .

Possiamo anche eseguire indipendentemente più thread, che è noto come MultiThreading. Questo thread ha anche la funzionalità di sleep con cui il thread attualmente in esecuzione viene sospeso (interruzione temporanea dell'esecuzione) per il numero di volte specificato. Ma il sonno lancia l'InterruptedException Quindi, dobbiamo gestirlo usando try / catch come questo.

```
try{Thread.sleep(500);}catch(InterruptedException e){System.out.println(e);}
```

Aggiornamento dell'interfaccia utente da un thread in background

È comune utilizzare un thread in background per eseguire operazioni di rete o attività a esecuzione prolungata, quindi aggiornare l'interfaccia utente con i risultati quando necessario.

Questo pone un problema, in quanto solo il thread principale può aggiornare l'interfaccia utente.

La soluzione è utilizzare il metodo `runOnUiThread()` , poiché consente di avviare l'esecuzione del codice sul thread dell'interfaccia utente da un thread in background.

In questo semplice esempio, una discussione viene avviata quando l'attività viene creata, viene eseguita fino a quando il numero magico `42` viene generato casualmente e quindi utilizza il metodo `runOnUiThread()` per aggiornare l'interfaccia utente una volta soddisfatta questa condizione.

```
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();
    }
}
```

Leggi Filo online: https://riptutorial.com/it/android/topic/7131/filo

Capitolo 101: Firebase

introduzione

Firebase è una piattaforma per applicazioni mobili e web con strumenti e infrastrutture progettati per aiutare gli sviluppatori a creare app di alta qualità.

Caratteristiche

Firebase Cloud Messaging, Firebase Auth, Realtime Database, Firebase Storage, Firebase Hosting, Firebase Test Lab per Android, Firebase Crash Reporting.

Osservazioni

Firebase - Documentazione estesa:

C'è un altro tag dove puoi trovare più argomenti ed esempi sull'uso di Firebase.

Altri argomenti correlati:

- Firebase Realtime DataBase
- Indicizzazione dell'app Firebase
- Firebase Crash Reporting
- Firebase Cloud Messaging

Examples

Crea un utente 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;
    }
}
```

Accedi utente Firebase con e-mail e password

```
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));
    }
}
```

Invia un'email di reimpostazione della password di 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();
                        }
                    }
                });
    }
}
```

Aggiornamento dell'email di un utente 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 password

```
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());
    }
}
```

Re-autentica utente 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();
    }
}
```

Firebase Storage Operations

Con questo esempio, sarai in grado di eseguire le seguenti operazioni:

1. Connetti a Firebase Storage
2. Crea una directory chiamata "images"
3. Carica un file nella directory images
4. Scarica un file dalla directory delle immagini
5. Elimina un file dalla directory delle immagini

```
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();
    }
}
```

Per impostazione predefinita, le regole di Storage di Firebase applicano la restrizione di autenticazione. Se l'utente è autenticato, solo allora, può eseguire operazioni su Firebase Storage, altrimenti non può farlo. Ho disabilitato la parte di autenticazione in questa demo aggiornando le regole di archiviazione. In precedenza, le regole erano simili a:

```
 service firebase.storage {
   match /b/**something**.appspot.com/o {
     match /{allPaths=**} {
       allow read, write: if request.auth != null;
     }
   }
 }
```

Ma ho cambiato per saltare l'autenticazione:

```
service firebase.storage {
```

```
  match /b/**something**.appspot.com/o {
    match /{allPaths=**} {
      allow read, write;
    }
  }
}
```

Firebase Cloud Messaging

Prima di tutto devi configurare il tuo progetto aggiungendo Firebase al tuo progetto Android seguendo i passaggi descritti in questo argomento .

Imposta Firebase e SDK FCM

Aggiungi la dipendenza FCM al file `build.gradle` livello di `build.gradle`

```
dependencies {
 compile 'com.google.firebase:firebase-messaging:11.0.4'
}
```

E in fondo (questo è importante) aggiungi:

```
// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

Modifica il manifest dell'app

Aggiungi il seguente al manifest dell'app:

- Un servizio che estende `FirebaseMessagingService` . Questo è necessario se si desidera eseguire la gestione dei messaggi oltre a ricevere notifiche su app in background.
- Un servizio che estende `FirebaseInstanceIdService` per gestire la creazione, la rotazione e l'aggiornamento dei token di registrazione.

Per esempio:

```
    <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>
```

Ecco le semplici implementazioni dei 2 servizi.

Per recuperare il token di registrazione corrente estendere la classe `FirebaseInstanceIdService` e sovrascrivere il metodo `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
    }
}
```

Per ricevere messaggi, utilizzare un servizio che estenda `FirebaseMessagingService` e sovrascrivi il metodo `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
    }
```

in **Firebase** puoi raggruppare l'utente in base al loro comportamento come "AppVersion, utente gratuito, utente di acquisto o regole specifiche" e quindi inviare notifiche a gruppi specifici inviando la funzione **argomento** in fireBase.
per registrare l'utente nell'uso dell'argomento

```
FirebaseMessaging.getInstance().subscribeToTopic("Free");
```

quindi nella console FireBase, invia la notifica per nome dell'argomento

Maggiori informazioni nell'argomento dedicato Firebase Cloud Messaging .

Aggiungi Firebase al tuo progetto Android

Ecco alcuni passaggi semplificati (basati sulla documentazione ufficiale) necessari per creare un progetto Firebase e collegarlo con un'app Android.

Aggiungi Firebase alla tua app

1. Crea un progetto Firebase nella console Firebase e fai clic su **Crea nuovo progetto** .
2. Fai clic su **Aggiungi Firebase all'app per Android** e segui i passaggi di installazione.
3. Quando richiesto, inserisci il **nome del pacchetto dell'app** .
   È importante inserire il nome del pacchetto completo che la tua app sta utilizzando; questo può essere impostato solo quando aggiungi un'app al tuo progetto Firebase.
4. Alla fine, scaricherai un file `google-services.json` . Puoi scaricare nuovamente questo file in qualsiasi momento.
5. Se non lo hai già fatto, copia il file `google-services.json` nella cartella del modulo del progetto, in genere `app/` .

Il prossimo passo è aggiungere l'SDK per integrare le librerie Firebase nel progetto.

Aggiungi l'SDK

Per integrare le librerie Firebase in uno dei tuoi progetti, devi eseguire alcune attività di base per preparare il tuo progetto Android Studio. Potresti averlo già fatto come parte dell'aggiunta di Firebase alla tua app.

1. Aggiungi le regole al tuo file `build.gradle` livello `build.gradle` , per includere il **plugin di google-services** :

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.1.0'
    }
}
```

Quindi, nel tuo file Gradle del modulo (in genere l' `app/build.gradle`), aggiungi la riga del plug-in in fondo al file per abilitare il plugin 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'
```

Il passaggio finale consiste nell'aggiungere le dipendenze per l'SDK di Firebase utilizzando una o più **librerie disponibili** per le diverse funzionalità di Firebase.

Gradle Dependency Line	Servizio
com.google.firebase: Firebase-core: 11.0.4	analitica
com.google.firebase: Firebase-database: 11.0.4	Database in tempo reale
com.google.firebase: Firebase-storage: 11.0.4	Conservazione
com.google.firebase: Firebase-incidente: 11.0.4	Segnalazione di crash
com.google.firebase: Firebase-auth: 11.0.4	Autenticazione
com.google.firebase: Firebase-messaging: 11.0.4	Cloud Messaging / Notifiche
com.google.firebase: Firebase-config: 11.0.4	Configurazione remota
com.google.firebase: Firebase-invita: 11.0.4	Invita / collegamenti dinamici
com.google.firebase: Firebase-annunci: 11.0.4	AdMob
com.google.android.gms: play-servizi-appindexing: 11.0.4	Indicizzazione delle app

Firebase Realtime Database: come impostare / ottenere dati

Nota: impostiamo alcune autenticazioni anonime per l'esempio

```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

Una volta terminato, crea un bambino modificando l'indirizzo del tuo database. Per esempio:

https://your-project.firebaseio.com/ a **https://your-project.firebaseio.com/chat**

Inseriremo i dati in questa posizione dal nostro dispositivo Android. Non **è necessario** creare la struttura del database (schede, campi ... ecc.), Verrà automaticamente creata quando si invierà l'oggetto Java a Firebase!

Creare un oggetto Java che contenga tutti gli attributi che si desidera inviare al database:

```
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;
    }
}
```

Quindi nella tua attività:

```
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
                }
            }
        });
}
```

Per inviare un valore:

```
ChatMessage msg = new ChatMessage("user1", "Hello World!");
reference.push().setValue(msg);
```

Per ricevere le modifiche che si verificano nel database:

```
reference.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        ChatMessage msg = dataSnapshot.getValue(ChatMessage.class);
```

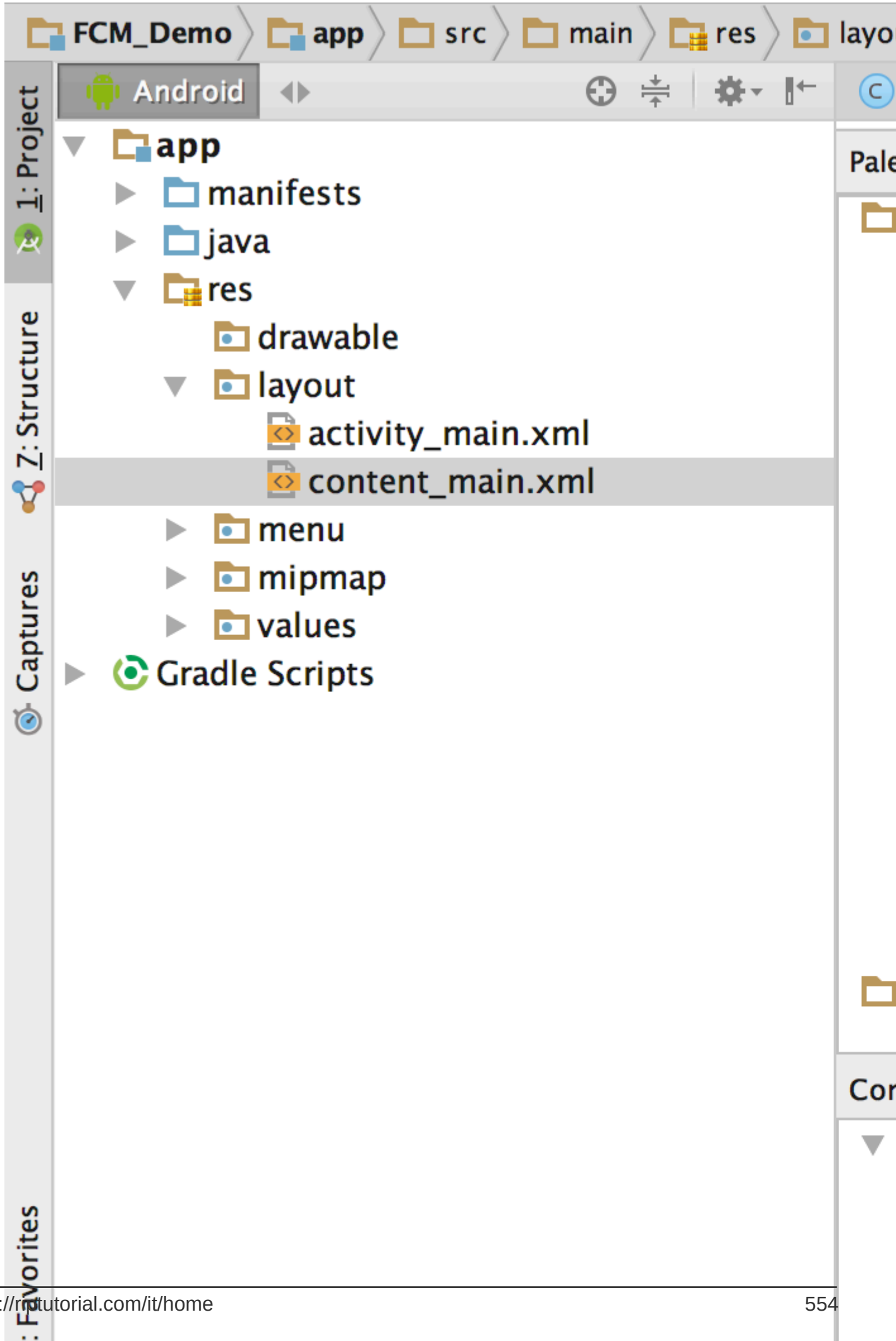
```
        Log.d(TAG, msg.getUsername()+" "+msg.getMessage());
    }

    public void onChildChanged(DataSnapshot dataSnapshot, String s) {}
    public void onChildRemoved(DataSnapshot dataSnapshot) {}
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {}
    public void onCancelled(DatabaseError databaseError) {}
});
```

Demo di notifiche basate su FCM

Questo esempio mostra come utilizzare la piattaforma Firebase Cloud Messaging (FCM). FCM è un successore di Google Cloud Messaging (GCM). Non richiede autorizzazioni C2D_MESSAGE dagli utenti dell'app.

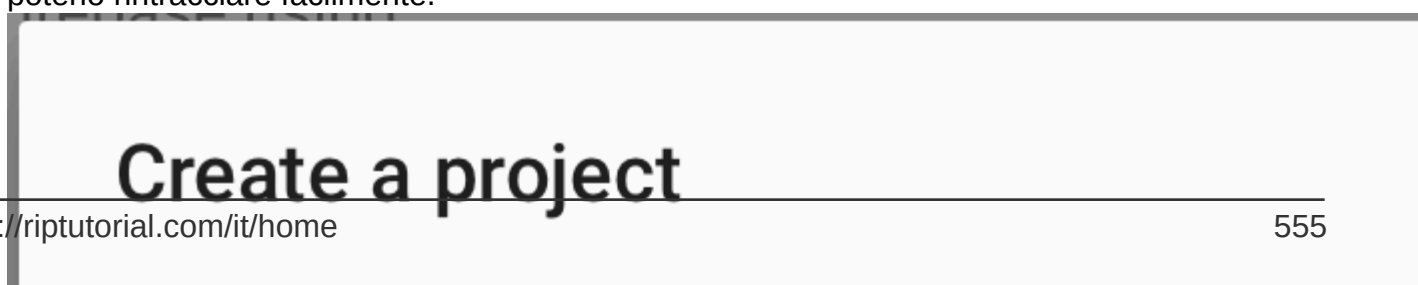
I passaggi per integrare FCM sono i seguenti.

1. Crea un esempio di progetto Hello World in Android Studio La tua schermata di Android Studio sarà simile alla seguente immagine.


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. Il prossimo passo è impostare il progetto Firebase. Visita https://console.firebase.google.com e crea un progetto con un nome identico, in modo da poterlo rintracciare facilmente.

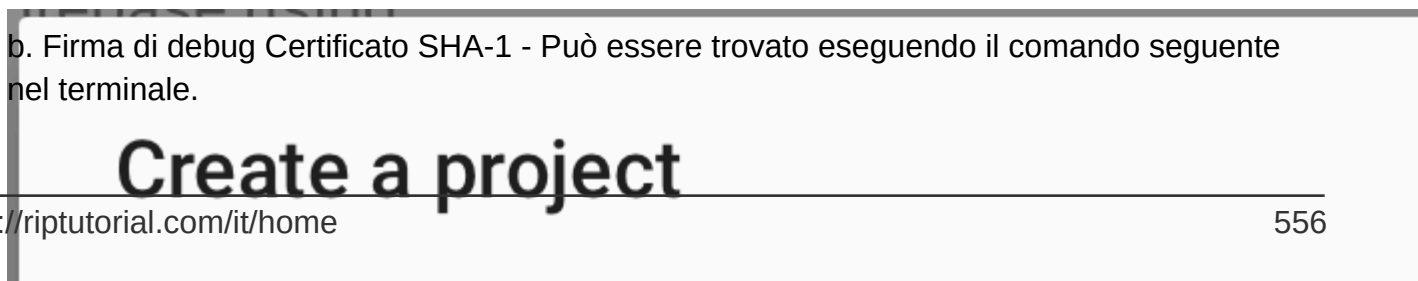
https://console.firebase.google.com e crea un progetto con un nome identico, in modo da poterlo rintracciare facilmente.



3. Ora è il momento di aggiungere firebase al tuo progetto Android di esempio che hai appena creato. Avrai bisogno del nome del pacchetto del tuo progetto e del certificato di firma debug SHA-1 (opzionale).

   un. Nome del pacchetto - Può essere trovato dal file XML manifest Android.

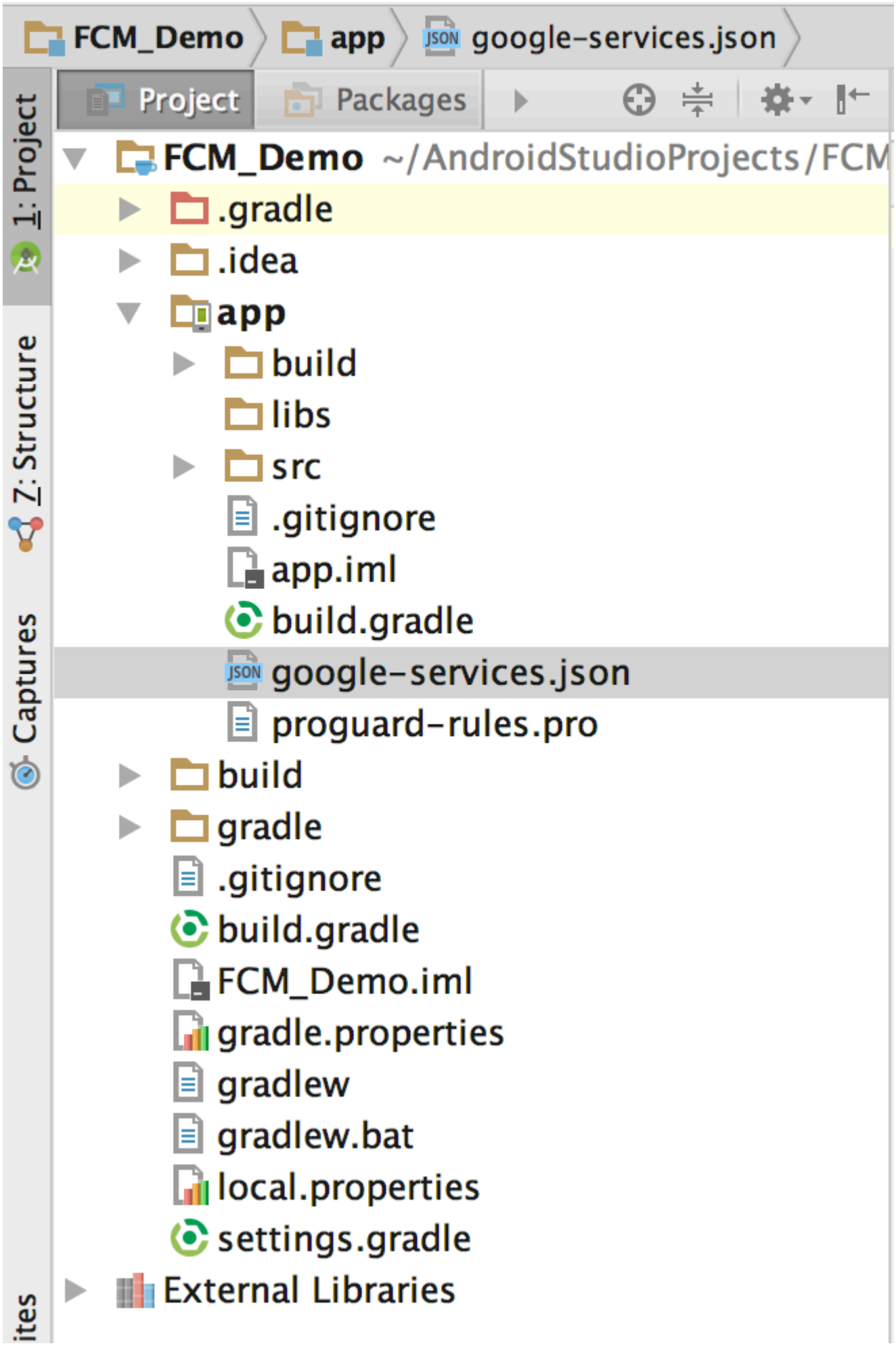
   b. Firma di debug Certificato SHA-1 - Può essere trovato eseguendo il comando seguente nel terminale.



```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
```

Inserisci queste informazioni nella console di Firebase e aggiungi l'app al progetto firebase. Dopo aver fatto clic sul pulsante aggiungi app, il browser scarica automaticamente un file JSON denominato "google-services.json".

4. Ora copia il file google-services.json che hai appena scaricato nella directory principale del modulo dell'app 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. Segui le istruzioni fornite sulla console Firebase mentre procedi. un. Aggiungi la seguente riga di codice al tuo livello di progetto build.gradle

```
dependencies{ classpath 'com.google.gms:google-services:3.1.0' .....
```

b. Aggiungi la seguente riga di codice alla fine del tuo livello di applicazione 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'
```

c. Android Studio ti chiederà di sincronizzare il progetto. Clicca su Sincronizza ora.

6. Il prossimo compito è aggiungere due servizi. un. Un estendere FirebaseMessagingService con intent-filter come segue

```
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
```

b. Un servizio FirebaseInstanceIDS esteso.

```
<intent-filter>
    <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
</intent-filter>
```

7. Il codice FirebaseMessagingService dovrebbe essere simile a questo.

```
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() {
    }
}
```

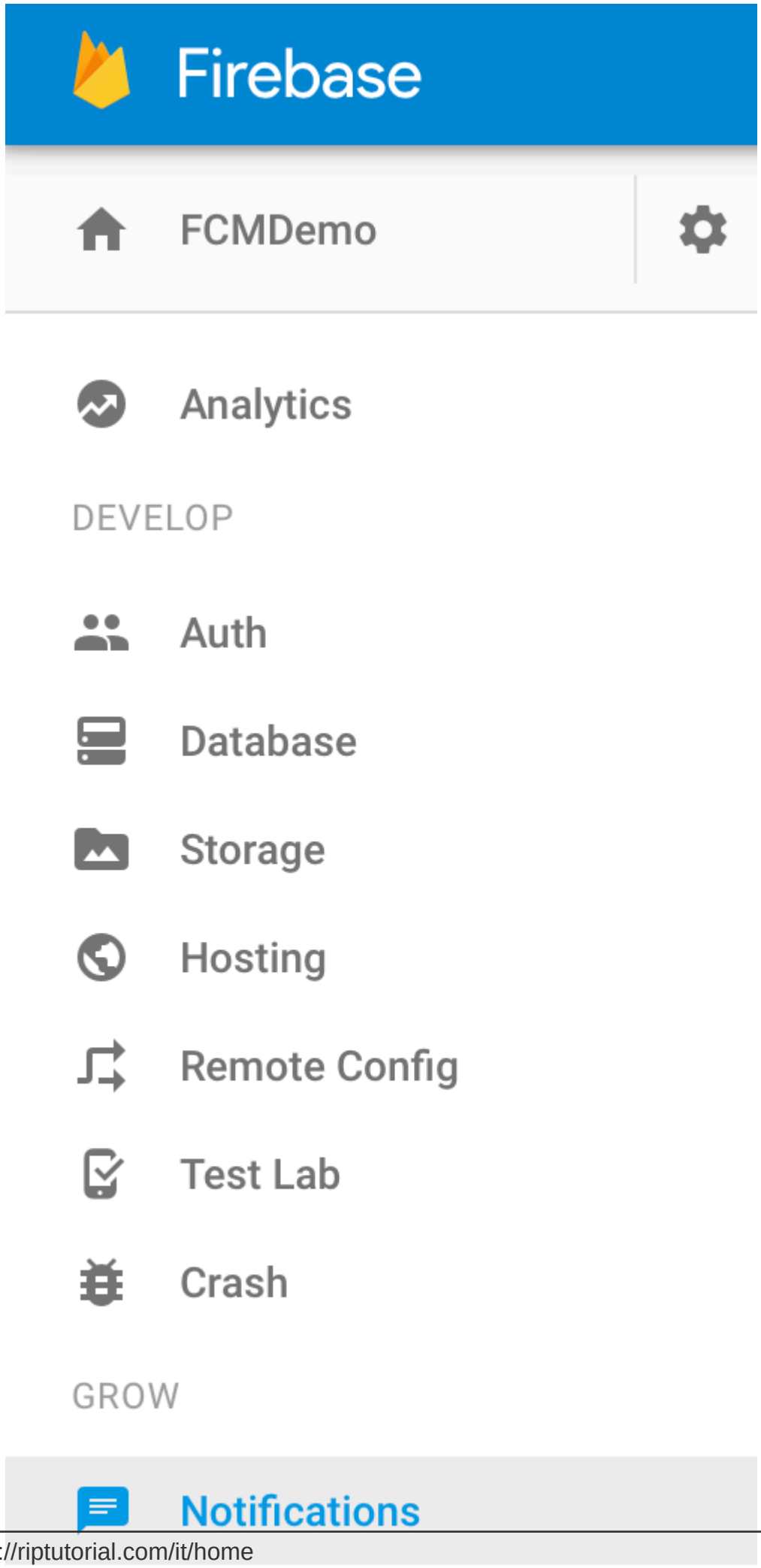
8. FirebaseInstanceIdService dovrebbe assomigliare a questo.

```
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. Ora è il momento di acquisire il token di registrazione del dispositivo. Aggiungi la seguente riga di codice al metodo onCreate di MainActivity.

```
String token = FirebaseInstanceId.getInstance().getToken();
Log.d("FCMAPP", "Token is "+token);
```

10. Una volta ottenuto il token di accesso, possiamo utilizzare la console di Firebase per inviare la notifica. Esegui l'app sul tuo telefono Android.


Firebase
FCMDemo
Analytics
DEVELOP
Auth
Database
Storage
Hosting
Remote Config
Test Lab
Crash
GROW
Notifications

Capitolo 102: Firebase Cloud Messaging

introduzione

Firebase Cloud Messaging (FCM) è una soluzione di messaggistica multipiattaforma che ti consente di consegnare i messaggi in modo affidabile senza alcun costo.

Utilizzando FCM, è possibile notificare a un'app client che sono disponibili nuove e-mail o altri dati da sincronizzare. È possibile inviare messaggi di notifica per guidare il riutilizzo e la fidelizzazione degli utenti. Per casi d'uso come la messaggistica istantanea, un messaggio può trasferire un carico utile fino a `4KB` in un'app client.

Examples

Configurare un'app client di messaggistica cloud Firebase su Android

1. Completa la parte Installazione e configurazione per collegare la tua app a Firebase. Questo creerà il progetto in Firebase.
2. Aggiungi la dipendenza per Firebase Cloud Messaging al tuo file `build.gradle` livello di `build.gradle` :

```
dependencies {
     compile 'com.google.firebase:firebase-messaging:10.2.1'
}
```

Ora sei pronto per lavorare con FCM in Android.

I client FCM richiedono dispositivi con `Android 2.3` o versioni successive con installata l'app Google Play Store o un emulatore con Android 2.3 con API Google.

Modifica il tuo file `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 di registrazione

All'avvio iniziale della tua app, l'SDK di FCM genera un token di registrazione per l'istanza dell'app client.
Se si desidera scegliere come target singoli dispositivi o creare gruppi di dispositivi, è necessario accedere a questo token estendendo `FirebaseInstanceIdService` .

Il callback `onTokenRefresh` ogni volta che viene generato un nuovo token ed è possibile utilizzare il metodo `FirebaseInstanceID.getToken()` per recuperare il token corrente.

Esempio:

```
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);

    }

}
```

Questo codice che ho implementato nella mia app per spingere immagini, messaggi e link per l'apertura nella tua webView

Questo è il mio 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;

    }
}}
```

E questo è MainActivity per aprire il link nella mia WebView o altro depanditore del browser in base alle tue esigenze tramite intenti.

```
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();
    }}
```

Ricevi messaggi

Per ricevere messaggi, utilizzare un servizio che estenda `FirebaseMessagingService` e sovrascrivi il metodo `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());
        }

        //.....
    }
```

Quando l'app è in background, Android indirizza i messaggi di notifica alla barra delle applicazioni. Un utente che tocca la notifica apre l'avvio dell'app per impostazione predefinita.

Ciò include i messaggi che contengono sia la notifica che il carico utile dei dati (e tutti i messaggi inviati dalla console delle notifiche). In questi casi, la notifica viene inviata alla barra delle applicazioni del dispositivo e il carico di dati viene consegnato negli extra dell'intento dell'attività di avvio.

Ecco un breve riassunto:

Stato dell'app	Notifica	Dati	Tutti e due
Primo piano	`onMessageReceived`	`onMessageReceived`	`onMessageReceived`
sfondo	Area di notifica	`onMessageReceived`	Notifica: vassoio di sistema
			Dati: in extra dell'intento.

Iscriviti a un argomento

Le app client possono iscriversi a qualsiasi argomento esistente oppure possono creare un nuovo argomento. Quando un'app client si iscrive a un nuovo nome di argomento, in FCM viene creato un nuovo argomento di tale nome e qualsiasi client può successivamente sottoscriverlo.

Per iscriverti a un argomento usa il metodo `subscribeToTopic()` specificando il nome dell'argomento:

```
FirebaseMessaging.getInstance().subscribeToTopic("myTopic");
```

Leggi Firebase Cloud Messaging online: https://riptutorial.com/it/android/topic/8826/firebase-cloud-messaging

Capitolo 103: Firebase Crash Reporting

Examples

Come aggiungere Firebase Crash Reporting alla tua app

Per aggiungere *Firebase Crash Reporting* alla tua app, procedi nel seguente modo:

- Crea un'app sulla *Firebase Console* qui .
- Copia il file `google-services.json` dal tuo progetto nella tua `app/` directory.
- Aggiungi le seguenti regole al file *build.gradle di* livello *root* per includere il plug `google-services` :

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.0.0'
    }
}
```

- Nel tuo file Gradle del modulo, aggiungi la linea del `apply plugin` nella parte inferiore del file per abilitare il plugin Gradle:

```
apply plugin: 'com.google.gms.google-services'
```

- Aggiungi la dipendenza per la *segnalazione di crash* al tuo file *build.gradle a* livello di *app* :

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

- È quindi possibile generare un'eccezione personalizzata dall'applicazione utilizzando la seguente riga:

```
FirebaseCrash.report(new Exception("Non Fatal Error logging"));
```

  Tutte le eccezioni fatali verranno segnalate alla tua *console Firebase* .

- Se si desidera aggiungere registri personalizzati a una console, è possibile utilizzare il seguente codice:

```
FirebaseCrash.log("Level 2 completed.");
```

Per maggiori informazioni per favore visita:

- Documentazione ufficiale

- Argomento dedicato Stack Overflow

Come segnalare un errore

Firebase Crash Reporting genera automaticamente report per errori irreversibili (o eccezioni non rilevate).

Puoi creare il tuo rapporto personalizzato usando:

```
FirebaseCrash.report(new Exception("My first Android non-fatal error"));
```

È possibile archiviare il registro quando FirebaseCrash ha inizializzato il modulo:

> 07-20 08: 57: 24.442 D / FirebaseCrashApiImpl: **API di reporting FirebaseCrash inizializzata** 07-20 08: 57: 24.442 I / FirebaseCrash: **report FirebaseCrash inizializzare** d com.google.firebase.crash.internal.zzg@3333d325 07-20 08: 57: 24.442 D / FirebaseApp: **classe inizializzata com.google.firebase.crash.FirebaseCrash.**

E poi quando ha inviato l'eccezione:

> 07-20 08: 57: 47.052 D / FirebaseCrashApiImpl: **lanciabile java.lang.Exception: il mio primo errore Android non fatale** 07-20 08: 58: 18.822 D / FirebaseCrashSenderServiceImpl: **codice risposta: 200** 07-20 08: 58: 18.822 D / FirebaseCrashSenderServiceImpl: **rapporto inviato**

Puoi aggiungere registri personalizzati al tuo rapporto con

```
FirebaseCrash.log("Activity created");
```

Leggi Firebase Crash Reporting online: https://riptutorial.com/it/android/topic/5965/firebase-crash-reporting

Capitolo 104: Firebase Realtime DataBase

Osservazioni

Altri argomenti correlati:

- Firebase

Examples

Gestore di eventi DataBase in realtime di Firebase

Prima inizializza FirebaseDatabase:

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

Scrivi nel tuo database:

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Leggi dal tuo database:

```
// 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());
    }
});
```

Recupera dati su eventi 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);
```

Configurazione rapida

1. Completa la parte Installazione e configurazione per collegare la tua app a Firebase. Questo creerà il progetto in Firebase.

2. Aggiungi la dipendenza per Firebase Realtime Database al tuo file `build.gradle` livello di `build.gradle` :

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

3. Configura le regole del database Firebase

Ora sei pronto per lavorare con il database Realtime in Android.

Ad esempio, si scrive un messaggio `Hello World` nel database sotto la chiave del `message` .

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Progettare e comprendere come recuperare i dati in tempo reale dal database Firebase

Questo esempio presuppone che tu abbia già impostato un database in tempo reale di Firebase.

Se sei un principiante, ti preghiamo di informarti qui su come aggiungere Firebase al tuo progetto Android.

Innanzitutto, aggiungi la dipendenza del database Firebase al file *build.gradle a* livello di app:

```
compile 'com.google.firebase:firebase-database:9.4.0'
```

Ora, creiamo un'app di chat che memorizza i dati nel database di Firebase.

Passaggio 1: creare una classe denominata Chat

Basta creare una classe con alcune variabili di base richieste per la chat:

```
public class Chat{
    public String name, message;
}
```

Passaggio 2: crea alcuni dati JSON

Per inviare / recuperare dati da / verso il database Firebase, è necessario utilizzare JSON. Supponiamo che alcune chat siano già memorizzate al livello root nel database. I dati di queste chat potrebbero apparire come segue:

```
[
    {
        "name":"John Doe",
        "message":"My first Message"
    },
    {
        "name":"John Doe",
        "message":"Second Message"
    },
    {
        "name":"John Doe",
        "message":"Third Message"
    }
]
```

Passaggio 3: aggiungere gli ascoltatori

Ci sono tre tipi di ascoltatori. Nell'esempio seguente useremo `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();
    }
});
```

Passaggio 4: aggiungere dati al database

Basta creare un oggetto di classe Chat e aggiungere i valori come segue:

```
Chat chat=new Chat();
chat.name="John Doe";
```

```
chat.message="First message from android";
```

Ora ottieni un riferimento al nodo chat come fatto nella sessione di recupero:

```
DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference().child("chats");
```

Prima di iniziare ad aggiungere dati, tieni presente che hai bisogno di un riferimento più approfondito poiché un nodo chat ha molti più nodi e aggiungere una nuova chat significa aggiungere un nuovo nodo contenente i dettagli della chat. Possiamo generare un nuovo nome univoco del nodo utilizzando la funzione `push()` sull'oggetto `DatabaseReference` , che restituirà un altro `DatabaseReference` , che a sua volta punta a un nodo appena formato per inserire i dati della chat.

Esempio

```
// The parameter is the chat object that was newly created a few lines above.
chatDb.push().setValue(chat);
```

La funzione `setValue()` farà in modo che tutte le funzioni `onDataChanged` dell'applicazione vengano chiamate (incluso lo stesso dispositivo), che risulta essere il listener collegato del nodo "chat".

Denormalizzazione: struttura piatta del database

La denormalizzazione e una struttura piatta del database sono necessarie per scaricare in modo efficiente chiamate separate. Con la seguente struttura, è anche possibile mantenere relazioni bidirezionali. Lo svantaggio di questo approccio è che devi sempre aggiornare i dati in più punti.

Ad esempio, immagina un'app che consenta all'utente di memorizzare i messaggi su se stesso (promemoria).

Struttura del database piatta desiderata:

```
|--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 classe memo utilizzata

```
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;
    }
}
```

Recupero dei memo di un utente

```
//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) { }
    }
```

Creare 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);
```

Dopo che il push, o il database appare così:

```
|--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"
```

Comprensione del database JSON di Firebase

Prima che ci sporchiamo le mani con il codice, sento che è necessario capire come i dati vengono archiviati in Firebase. A differenza dei database relazionali, Firebase memorizza i dati in formato JSON. Pensa a ogni riga di un database relazionale come un oggetto JSON (che è in pratica una coppia chiave-valore non ordinata). Quindi il nome della colonna diventa chiave e il valore memorizzato in quella colonna per una riga specifica è il valore. In questo modo l'intera riga viene rappresentata come un oggetto JSON e un elenco di questi rappresenta un'intera tabella di database. Il vantaggio immediato che vedo per questo è la modifica dello schema diventa un'operazione molto più economica rispetto al vecchio RDBMS. È più semplice aggiungere un paio di ulteriori attributi a un JSON piuttosto che alterare una struttura di tabella.

ecco un esempio JSON per mostrare come i dati sono memorizzati in 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
}
```

Questo mostra chiaramente come i dati che abbiamo usato per memorizzare nei database

relazionali possono essere archiviati in formato JSON. Ora vediamo come leggere questi dati nei dispositivi Android.

Recupero di dati da Firebase

Immagino che tu sappia già aggiungere il campo base delle dipendenze di gradle in Android Studio. Se non segui semplicemente la guida da qui . Aggiungi la tua app nella console di Firebase, gradle sync android studio dopo aver aggiunto le dipendenze. Tutte le dipendenze non sono necessarie solo il database di Firebase e l'autenticazione di Firebase.

Ora che sappiamo come vengono archiviati i dati e come aggiungere le dipendenze gradle, vediamo come utilizzare l'SDK Android firebase importato per recuperare i dati.

creare un riferimento al database Firebase

```
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)
```

da qui puoi concatenare più chiamate di metodo child () per puntare ai dati a cui sei interessato. Ad esempio, se i dati sono memorizzati come illustrato nella sezione precedente e vuoi puntare all'utente di Bruce Wayne puoi usare:

```
DatabaseReference bruceWayneRef = userDBRef.child("371298");
// 371298 is key of bruce wayne user in JSON structure (previous section)
```

O semplicemente passa l'intero riferimento all'oggetto 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..."
```

Ora che abbiamo il riferimento dei dati che vogliamo recuperare, possiamo usare gli ascoltatori per recuperare i dati nelle app Android. A differenza delle chiamate tradizionali in cui si attivano le chiamate API REST utilizzando retrofit o volley, qui è necessario un semplice listener di callback per ottenere i dati. Firebase sdk chiama i metodi di callback e il gioco è fatto.

Esistono fondamentalmente due tipi di listener che è possibile collegare, uno è ValueEventListener e l'altro è ChildEventListener (descritto nella sezione successiva). Per qualsiasi modifica dei dati nel nodo abbiamo riferimenti e aggiunti listener, i listener di eventi value restituiscono l'intera struttura JSON e il listener di eventi figlio restituisce child specifico in cui è avvenuta la modifica. Entrambi sono utili a modo loro. Per recuperare i dati da Firebase possiamo aggiungere uno o più listener a un database di Firebase (elenco userDBRef creato in precedenza).

Ecco alcuni esempi di codice (spiegazione del codice dopo il codice):

```
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
    });
```

Hai notato il tipo di classe passato. DataSnapshot può convertire i dati JSON nei POJO definiti, passare semplicemente il tipo di classe corretto.

Se il tuo caso d'uso non richiede l'intero dato (nel nostro caso la tabella user_base) ogni volta che si verifica qualche piccolo cambiamento o dici di voler ***recuperare i dati solo una volta*** , puoi usare il metodo ***addListenerForSingleValueEvent ()*** del riferimento Database. Questo spara il callback solo una volta.

```
userDBRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // Do something
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Do something about the error
    });
```

Sopra i campioni verrà fornito il valore del nodo JSON. Per ottenere la chiave basta chiamare:

```
String myKey = dataSnapshot.getKey();
```

Ascolto di aggiornamenti secondari

Prendi un caso d'uso, come un'app di chat o un'app per la lista della spesa collaborativa (in pratica richiede un elenco di oggetti da sincronizzare tra gli utenti). Se si utilizza il database Firebase e si aggiunge un listener di eventi di valore al nodo padre della chat o al nodo padre dell'elenco generi alimentari, si terminerà con l'intera struttura della chat dall'inizio (intendevo l'inizio della chat) ogni volta che viene aggiunto un nodo di chat (cioè qualcuno dice ciao). Che non vogliamo fare, ciò a cui siamo interessati è solo il nuovo nodo o solo il vecchio nodo che è stato cancellato o modificato, quelli non modificati non dovrebbero essere restituiti.

In questo caso possiamo usare ChildEvenListener . Senza ulteriori adieu, ecco un esempio di codice (vedere le sezioni precedenti per dati JSON di esempio):

```
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) {
    });
```

I nomi dei metodi sono auto esplicativi. Come puoi vedere ogni volta che viene aggiunto un nuovo utente o se alcune proprietà dell'utente esistente vengono modificate o l'utente viene eliminato o rimosso, viene richiamato il metodo di callback appropriato del listener di eventi figlio con i dati rilevanti. Quindi se stai mantenendo l'interfaccia utente aggiornata per dire l'app di chat, prendi il JSON da onChildAdded () analizza in POJO e inseriscilo nell'interfaccia utente. Ricordati di rimuovere l'ascoltatore quando l'utente lascia lo schermo.

onChildChanged () fornisce l'intero valore figlio con proprietà modificate (nuove).

onChiledRemoved () restituisce il nodo figlio rimosso.

Recupero dei dati con impaginazione

Quando hai un enorme database JSON, aggiungere un listener di eventi di valore non ha senso. Restituirà l'enorme JSON e la sua analisi richiederebbe molto tempo. In questi casi possiamo usare l'impaginazione e recuperare parte dei dati e visualizzarli o elaborarli. Un po 'come caricare pigro o come recuperare vecchie chat quando l'utente fa clic su mostra chat più vecchia. In questo caso, è possibile utilizzare Query .

Prendiamo il nostro vecchio esempio nelle sezioni precedenti. La base utenti contiene 3 utenti, se cresce fino a dire trecentomila utenti e si desidera recuperare l'elenco utenti in gruppi di 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
    });
```

Qui è possibile aggiungere e ascoltare eventi value o child. Chiama nuovamente la query per recuperare il prossimo 50. **Assicurati di aggiungere il metodo orderByChild ()** , questo non funzionerà senza. Firebase deve conoscere l'ordine con cui stai impaginando.

Leggi Firebase Realtime DataBase online: https://riptutorial.com/it/android/topic/5511/firebase-realtime-database

Capitolo 105: Firma la tua app per Android per la versione

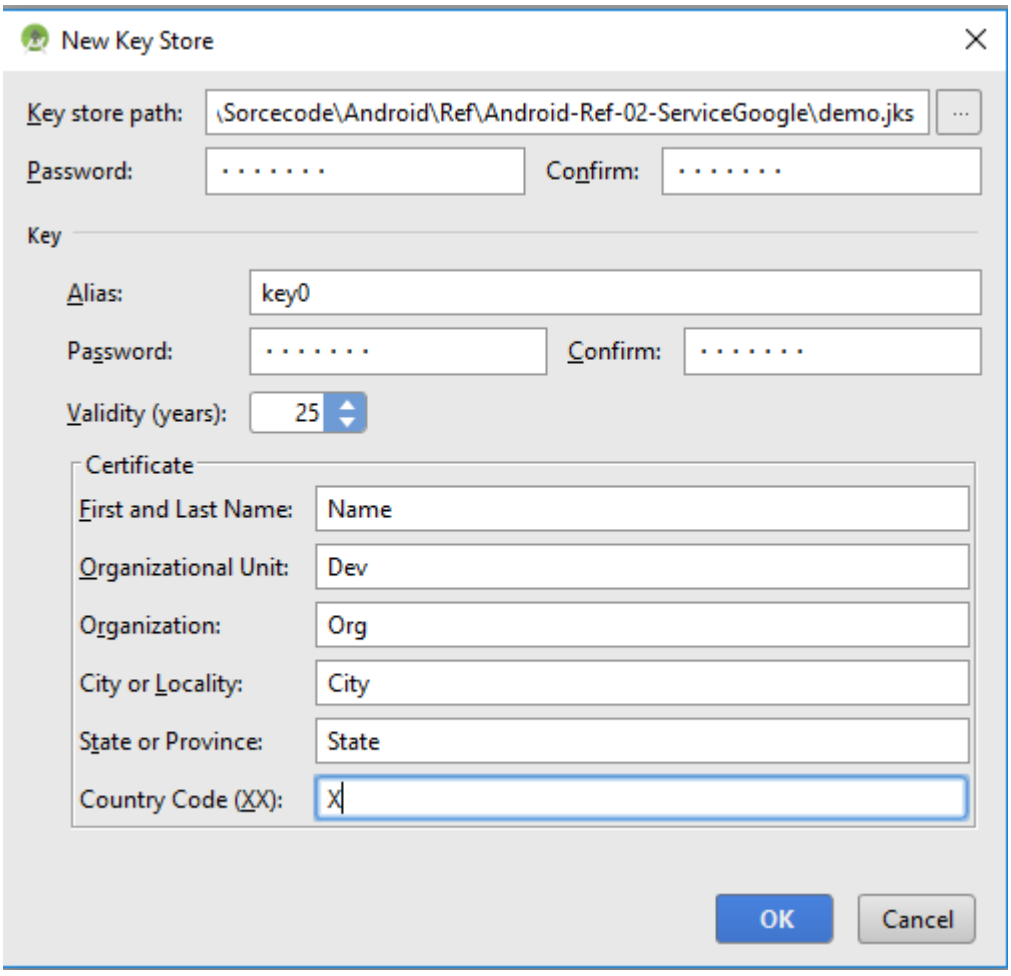
introduzione

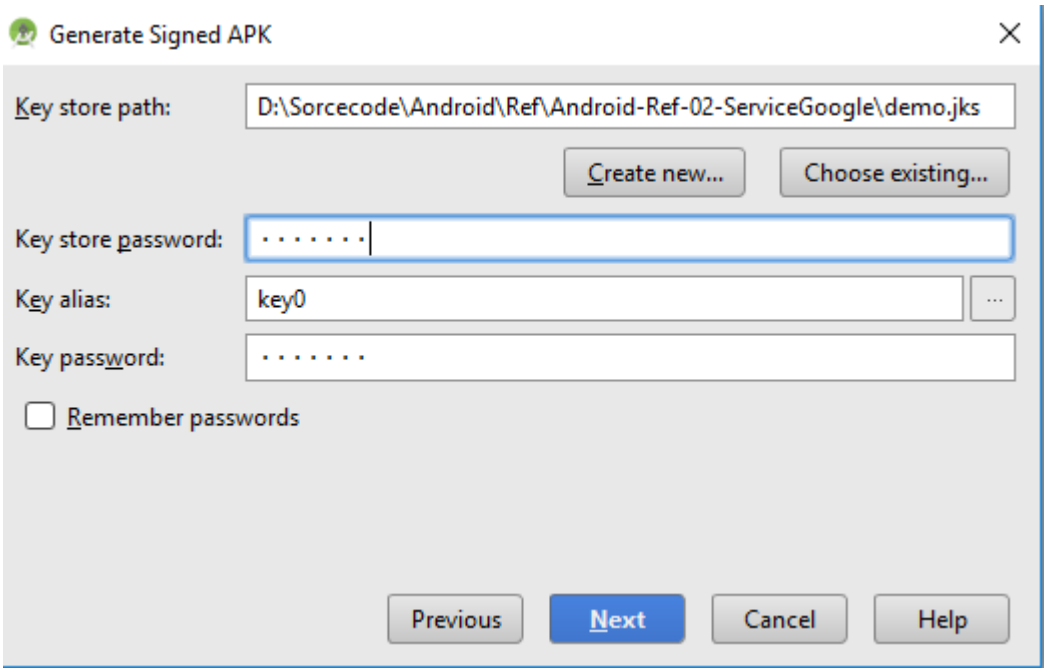
Android richiede che tutti gli APK siano firmati per il rilascio.

Examples

Firma la tua app

1. Nella barra dei menu, fai clic su Crea> Genera APK firmato.
2. Seleziona il modulo che desideri rilasciare dal menu a discesa e fai clic su Avanti.
3. Per creare un nuovo keystore, fare clic su Crea nuovo. Ora inserisci le informazioni richieste e premi ok in New Key Store.





4. Nella Creazione guidata APK firmato i campi sono già compilati per te se hai appena creato un nuovo archivio chiavi, altrimenti compila e fai clic su Avanti.

5. Nella finestra successiva, seleziona una destinazione per l'APK firmato, seleziona il tipo di costruzione e fai clic su Fine.

Configura build.gradle con la configurazione della firma

È possibile definire la configurazione della firma per firmare l'apk nel file `build.gradle` .

Puoi definire:

- `storeFile` : il file keystore
- `storePassword` : la password del keystore
- `keyAlias` : un nome alias chiave
- `keyPassword` : una password alias chiave

Devi **definire** il blocco `signingConfigs` per creare una configurazione di firma:

```
android {
    signingConfigs {

        myConfig {
            storeFile file("myFile.keystore")
            storePassword "xxxx"
            keyAlias "xxxx"
            keyPassword "xxxx"
        }
    }
    //....
}
```

Quindi puoi **assegnarlo** a uno o più tipi di build.

```
android {

    buildTypes {
        release {
            signingConfig signingConfigs.myConfig
        }
    }
}
```

Leggi Firma la tua app per Android per la versione online:
https://riptutorial.com/it/android/topic/9721/firma-la-tua-app-per-android-per-la-versione

Capitolo 106: FloatingActionButton

introduzione

Il pulsante di azione mobile viene utilizzato per un tipo speciale di azione promossa, che si anima sullo schermo come un pezzo di materiale in espansione, per impostazione predefinita. L'icona al suo interno può essere animata, inoltre FAB può spostarsi in modo diverso rispetto ad altri elementi dell'interfaccia utente a causa della loro importanza relativa. Un pulsante di azione mobile rappresenta l'azione principale in un'applicazione che può semplicemente attivare un'azione o navigare da qualche parte.

Parametri

Parametro	Dettaglio
`android.support.design:elevation`	Valore di elevazione per il FAB. Può essere un riferimento a un'altra risorsa, nella forma "@ [+] [pacchetto:] tipo / nome" o un attributo tema nel modulo "? [Pacchetto:] tipo / nome".
`android.support.design:fabSize`	Dimensioni per il FAB.
`android.support.design:rippleColor`	Ripple color per il FAB.
`android.support.design:useCompatPadding`	Abilita riempimento compatto.

Osservazioni

I pulsanti di azione mobili vengono utilizzati per un tipo speciale di azione promossa. Sono caratterizzati da un'icona circolare che galleggia sopra l'interfaccia utente e presentano comportamenti di movimento speciali correlati al morphing, al lancio e al punto di ancoraggio di trasferimento.

È consigliato un solo pulsante di azione mobile per schermata per rappresentare l'azione più comune.

Prima di utilizzare `FloatingActionButton` è necessario aggiungere la dipendenza della libreria del supporto di progettazione nel file `build.gradle` :

```
dependencies {
    compile 'com.android.support:design:25.1.0'
}
```

Documentazione ufficiale:

https://developer.android.com/reference/android/support/design/widget/FloatingActionButton.html

Specifiche di progettazione materiale:

https://material.google.com/components/buttons-floating-action-button.html

Examples

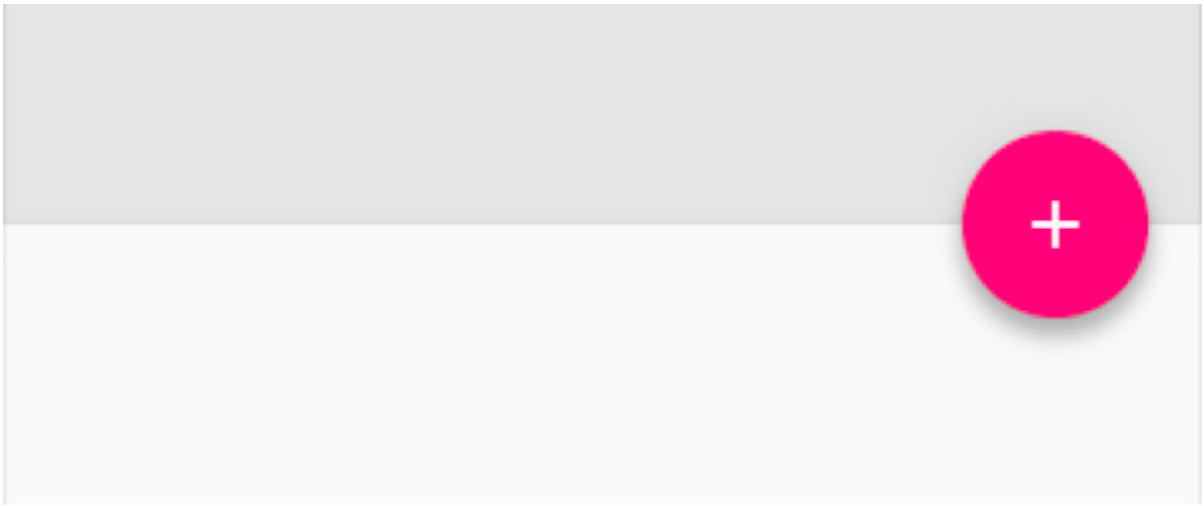
Come aggiungere il FAB al layout

Per usare un FloatingActionButton basta aggiungere la dipendenza nel file `build.gradle` come descritto nella sezione commenti.

Quindi aggiungi al layout:

```
<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 esempio:



Colore

Il colore di sfondo di questa vista è impostato su colorAccent del tema.

Nell'immagine sopra se l' `src` punta solo a + icona (di default 24x24 dp), per ottenere il *colore* di *sfondo* della circonferenza completa puoi usare `app:backgroundTint="@color/your_colour"`

Se desideri cambiare il colore nel codice puoi usare,

```
myFab.setBackgroundTintList(ColorStateList.valueOf(your color in int));
```

Se si desidera modificare il colore di FAB in uso stato premuto

```
mFab.setRippleColor(your color in int);
```

Posizionamento

Si consiglia di posizionare minimo di 16 dp dal bordo sul cellulare e minimo di 24 dp su tablet / desktop.

Nota : una volta impostato un src eccetto per coprire l'intera area di `FloatingActionButton` assicurati di avere la giusta dimensione di quell'immagine per ottenere il miglior risultato.

La dimensione del cerchio predefinita è 56 x 56 dp

Dimensioni minime cerchio: 40 x 40 dpi

Se si desidera modificare solo l'icona Interno, utilizzare l'icona 24 x 24 dpi per la dimensione predefinita

Mostra e Nascondi FloatingActionButton su Swipe

Per mostrare e nascondere un oggetto `FloatingActionButton` con l'animazione predefinita, basta chiamare i metodi `show()` e `hide()` . È buona prassi mantenere un oggetto `FloatingActionButton` nel layout di attività invece di inserirlo in un frammento, questo consente alle animazioni predefinite di funzionare quando vengono mostrate e nascoste.

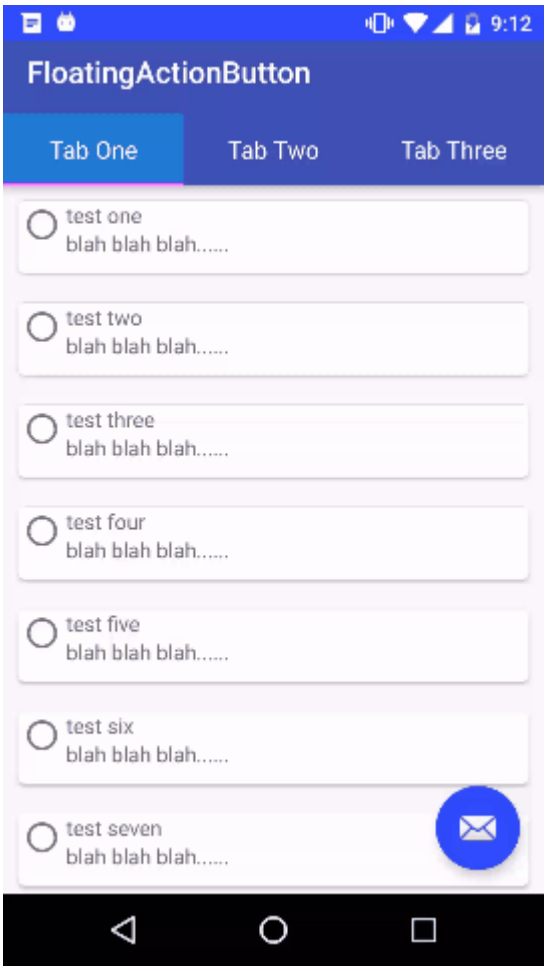
Ecco un esempio con `ViewPager` :

- Tre schede
- Mostra `FloatingActionButton` per la prima e terza scheda
- Nascondi il `FloatingActionButton` nella scheda centrale

```
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();
                }
            }
        });

    }
}
```

Risultato:



Mostra e nasconde FloatingActionButton su Scroll

A partire dalla Libreria di supporto versione 22.2.1, è possibile mostrare e nascondere un FloatingActionButton dal comportamento di scorrimento usando una sottoclasse `FloatingActionButton.Behavior` che sfrutta i metodi `show()` e `hide()` .

Si noti che questo funziona solo con un CoordinatorLayout in combinazione con le viste interne che supportano lo scorrimento annidato, come RecyclerView e NestedScrollView .

Questa classe `ScrollAwareFABBehavior` proviene dalle guide Android su Codepath (cc-wiki con attribuzione richiesta)

```
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();
        }
    }
}
```

Nel file di layout FloatingActionButton xml, specifica l' `app:layout_behavior` con il nome di classe `ScrollAwareFABBehavior` di `ScrollAwareFABBehavior` :

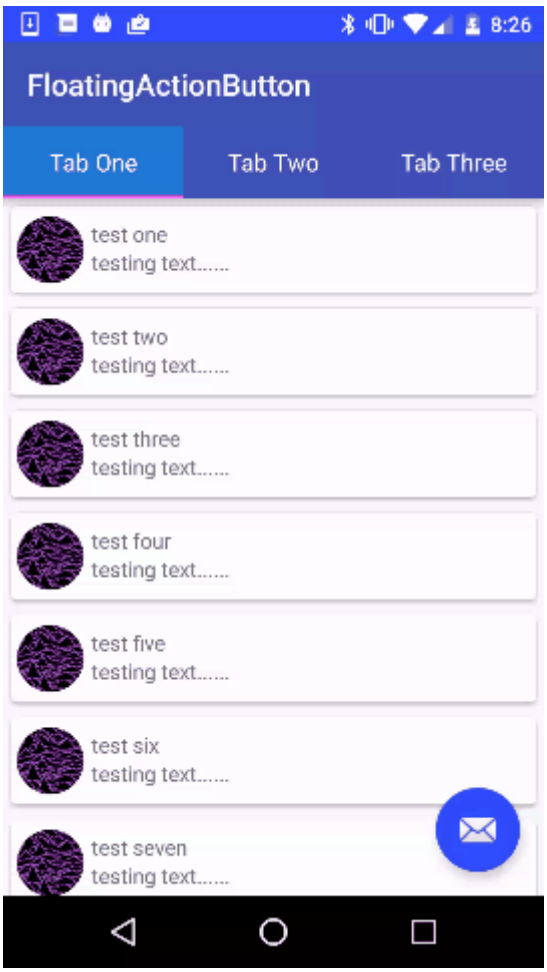
```
app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
```

Ad esempio con questo layout:

```
<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>
```

Ecco il risultato:



Impostazione del comportamento di FloatingActionButton

È possibile impostare il comportamento del FAB in XML.

Per esempio:

```
<android.support.design.widget.FloatingActionButton
   app:layout_behavior=".MyBehavior" />
```

Oppure puoi impostare a livello di codice usando:

```
CoordinatorLayout.LayoutParams p = (CoordinatorLayout.LayoutParams) fab.getLayoutParams();
p.setBehavior(xxxx);
fab.setLayoutParams(p);
```

Leggi FloatingActionButton online: https://riptutorial.com/it/android/topic/2979/floatingactionbutton

Capitolo 107: Fogli inferiori

introduzione

Un foglio inferiore è un foglio che scorre dal bordo inferiore dello schermo.

Osservazioni

I fogli in basso scorrono verso l'alto dalla parte inferiore dello schermo per rivelare più contenuti. Sono stati aggiunti alla libreria di supporto Android nella versione v23.2.0.

Examples

BottomSheetBehavior come le mappe di Google

2.1.x

Questo esempio dipende dalla libreria di supporto 23.4.0. +.

BottomSheetBehavior è caratterizzato da:

1. Due barre degli strumenti con animazioni che rispondono ai movimenti del foglio inferiore.
2. Un FAB che si nasconde quando è vicino alla "barra degli strumenti modale" (quella che appare quando si fa scorrere verso l'alto).
3. Un'immagine sullo sfondo dietro il foglio inferiore con una sorta di effetto di parallasse.
4. Un titolo (TextView) nella barra degli strumenti che appare quando il foglio di base lo raggiunge.
5. La barra di notifica satus può trasformare il suo sfondo in trasparente o a colori.
6. Un comportamento del foglio di fondo personalizzato con uno stato di "ancora".

Ora controlliamoli uno per uno:

ToolBars
Quando apri la visualizzazione in Google Maps, puoi vedere una barra degli strumenti in cui puoi effettuare ricerche, è l'unica che non sto facendo esattamente come Google Maps, perché volevo farlo più generico. In ogni caso che `ToolBar` è all'interno di un `AppBarLayout` ed ottenuto nascosto quando si inizia a trascinare il BottomSheet e appare di nuovo quando il BottomSheet raggiungere il `COLLAPSED` Stato.
Per realizzarlo è necessario:

- creare un `Behavior` ed estenderlo da `AppBarLayout.ScrollingViewBehavior`
- sovrascrivere i metodi `layoutDependsOn` e `onDependentViewChanged` . Facendolo ascolterai i movimenti del bottomSheet.

- creare alcuni metodi per nascondere e mostrare AppBarLayout / ToolBar con animazioni.

Ecco come l'ho fatto per la prima barra degli strumenti 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();
}
```

Ecco il file completo se ne hai bisogno

La seconda barra degli strumenti o la barra degli strumenti "Modale":
Devi sovrascrivere gli stessi metodi, ma in questo devi occuparti di più comportamenti:

- mostra / nascondi la barra degli strumenti con animazioni
- cambia colore / sfondo della barra di stato
- mostra / nasconde il titolo BottomSheet nella barra degli strumenti
- chiudi il bottomSheet o invialo allo stato compresso

Il codice per questo è un po 'esteso, quindi lascerò il link

Il FAB

Anche questo è un comportamento personalizzato, ma si estende da `FloatingActionButton.Behavior` . In `onDependentViewChanged` devi guardare quando raggiunge "offSet" o puntare dove vuoi nasconderlo. Nel mio caso, voglio nasconderlo quando è vicino alla seconda barra degli strumenti, quindi scaverò nel genitore FAB (un CoordinatorLayout) cercando l'AppBarLayout che contiene la barra degli strumenti, quindi uso la posizione di ToolBar come `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 il collegamento Comportamento FAB personalizzato

L'immagine dietro il BottomSheet con effetto di parallasse :
Come gli altri, è un comportamento personalizzato, l'unica cosa "complicata" in questo è il piccolo algoritmo che mantiene l'immagine ancorata al BottomSheet ed evita il collasso dell'immagine come l'effetto di parallasse predefinito:

```
@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;
}
```

Il file completo per l'immagine di sfondo con effetto di parallasse

Ora per la fine: **il comportamento del documento di base personalizzato**
Per raggiungere i 3 passaggi, devi prima capire che il predefinito BottomSheetBehavior ha 5 stati: `STATE_DRAGGING`, `STATE_SETTLING`, `STATE_EXPANDED`, `STATE_COLLAPSED`, `STATE_HIDDEN` e per il comportamento di Google Maps devi aggiungere uno stato intermedio tra collassato ed espanso: `STATE_ANCHOR_POINT` .
Ho provato ad estendere il bottomSheetBehavior predefinito senza successo, quindi ho appena copiato tutto il codice e ho modificato ciò di cui ho bisogno.
Per ottenere ciò di cui sto parlando, segui i seguenti passi:

1. Creare una classe Java ed estenderla da `CoordinatorLayout.Behavior<V>`
2. Copia il codice incolla dal file `BottomSheetBehavior` predefinito a quello nuovo.
3. Modificare il metodo `clampViewPositionVertical` con il seguente codice:

```
@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. Aggiungi un nuovo stato

   public static final int STATE_ANCHOR_POINT = X;

5. Modificare i seguenti metodi: `onLayoutChild` , `onStopNestedScroll` , `BottomSheetBehavior<V> from(V view)` e `setState` (opzionale)

```
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;
    }
```

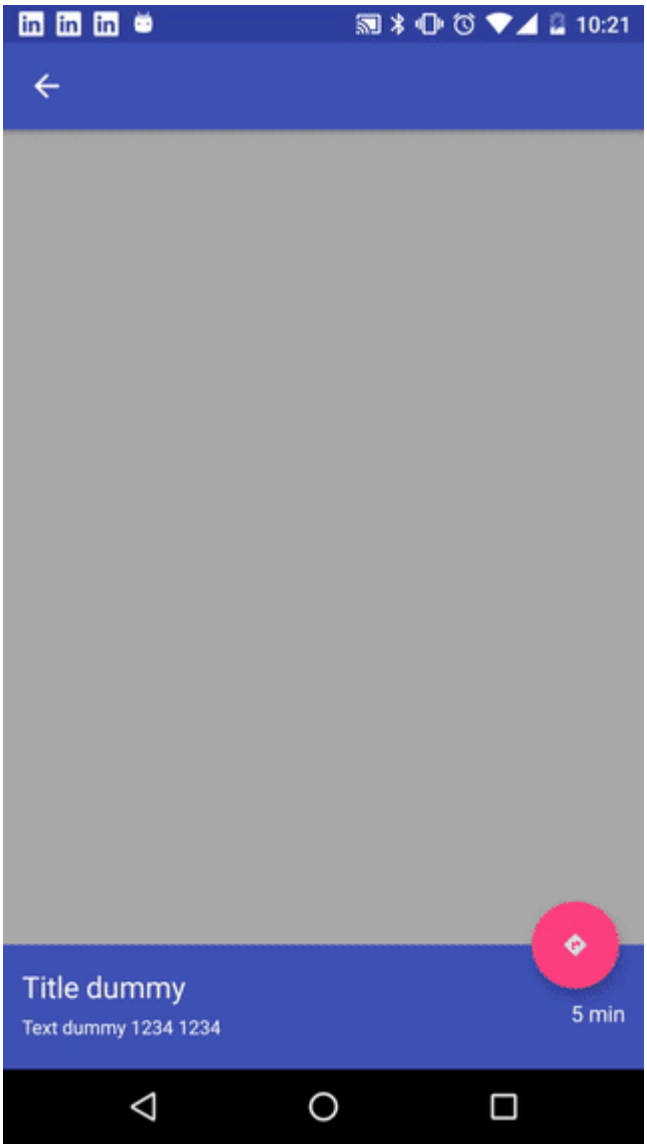
```
    if (target != mNestedScrollingChildRef.get() || !mNestedScrolled) {
        return;
    }
    int top;
    int targetState;
    if (mLastNestedScrollDy > 0) {
        //top = mMinOffset;
        //targetState = STATE_EXPANDED;
        int currentTop = child.getTop();
        if (currentTop > 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;
}
```

Collegamento all'intero progetto in cui è possibile visualizzare tutti i comportamenti personalizzati

Ed ecco come appare:

[



]

Configurazione rapida

Assicurati che la seguente dipendenza venga aggiunta al file build.gradle dell'app in dipendenze:

```
compile 'com.android.support:design:25.3.1'
```

Quindi puoi utilizzare il foglio in basso usando queste opzioni:

- `BottomSheetBehavior` da utilizzare con `CoordinatorLayout`
- `BottomSheetDialog` che è una finestra di dialogo con un comportamento del foglio di fondo
- `BottomSheetDialogFragment` che è un'estensione di `DialogFragment` , che crea un `BottomSheetDialog` anziché una finestra di dialogo standard.

Fogli inferiori persistenti

È possibile ottenere un foglio inferiore persistente allegando un `BottomSheetBehavior` foglio inferiore a un bambino Vista di un `CoordinatorLayout` `BottomSheetBehavior` :

```
<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>
```

Quindi nel tuo codice puoi creare un riferimento usando:

```
 // The View with the BottomSheetBehavior
 View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
 BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

Puoi impostare lo stato di BottomSheetBehavior usando il metodo setState () :

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

Puoi utilizzare uno di questi stati:

- `STATE_COLLAPSED` : questo stato compresso è l'impostazione predefinita e mostra solo una parte del layout lungo il fondo. L'altezza può essere controllata con l' `app:behavior_peekHeight` attributo `app:behavior_peekHeight` (predefinito su 0)
- `STATE_EXPANDED` : lo stato completamente espanso del foglio inferiore, dove è visibile l'intero foglio inferiore (se la sua altezza è inferiore al `CoordinatorLayout` contenente) o l'intero `CoordinatorLayout` è pieno
- `STATE_HIDDEN` : disabilitato per impostazione predefinita (e abilitato con l' `app:behavior_hideable` attributo `app:behavior_hideable`), abilitando questo consente agli utenti di scorrere verso il basso sul foglio inferiore per nascondere completamente il foglio inferiore

Se desideri ricevere i callback delle modifiche di stato, puoi aggiungere un `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
   }
 });
```

Fogli di fondo modali con BottomSheetDialogFragment

È possibile realizzare un foglio inferiore modale utilizzando un oggetto `BottomSheetDialogFragment` .

`BottomSheetDialogFragment` è un foglio inferiore modale.
Questa è una versione di `DialogFragment` che mostra un foglio inferiore utilizzando `BottomSheetDialog` anziché una finestra di dialogo mobile.

Basta definire il frammento:

```
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);
    }
}
```

Quindi utilizzare questo codice per mostrare il frammento:

```
MyBottomSheetDialogFragment mySheetDialog = new MyBottomSheetDialogFragment();
FragmentManager fm = getSupportFragmentManager();
mySheetDialog.show(fm, "modalSheetDialog");
```

Questo frammento creerà un `BottomSheetDialog` .

Fogli di fondo modali con BottomSheetDialog

`BottomSheetDialog` è una finestra di dialogo disegnata come un foglio inferiore

Basta usare:

```
//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();
```

In questo caso non è necessario allegare un comportamento BottomSheet.

Apri il parametro BottomFragment BottomSheet in modalità estesa per impostazione predefinita.

BottomSheet DialogFragment si apre in `STATE_COLLAPSED` per impostazione predefinita. Quale può essere forzato per aprirsi a `STATE_EXPANDED` e occupare lo schermo di dispositivo completo con l'aiuto del seguente modello di codice.

@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;
}
```

Sebbene l'animazione delle finestre di dialogo sia leggermente visibile, l'operazione di apertura di DialogFragment a schermo intero è molto buona.

Leggi Fogli inferiori online: https://riptutorial.com/it/android/topic/5702/fogli-inferiori

Capitolo 108: Formattare stringhe

Examples

Formatta una risorsa stringa

È possibile aggiungere caratteri jolly nelle risorse stringa e popolarli in fase di runtime:

1. Modifica strings.xml

```
<string name="my_string">This is %1$s</string>
```

2. Formattare la stringa secondo necessità

```
String fun = "fun";
context.getString(R.string.my_string, fun);
```

Formatta un timestamp in stringa

Per la descrizione completa dei modelli, vedere Riferimento SimpleDateFormat

```
Date now = new Date();
long timestamp = now.getTime();
SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy", Locale.US);
String dateStr = sdf.format(timestamp);
```

Formattazione dei tipi di dati in String e viceversa

Tipi di dati per la formattazione di stringhe

I tipi di dati come int, float, double, long, boolean possono essere formattati in string 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 versa di questo, formattare una stringa su un altro tipo di dati

```
Integer.parseInt("1"); //Output -> 1
Float.parseFloat("1.2"); //Output -> 1.2
Boolean.parseBoolean("true"); //Output -> true
```

Leggi Formattare stringhe online: https://riptutorial.com/it/android/topic/1346/formattare-stringhe

Capitolo 109: Formattazione dei numeri di telefono con pattern.

introduzione

Questo esempio mostra come formattare i numeri di telefono con un patter

Avrai bisogno della seguente libreria nel tuo gradle.

compila "com.googlecode.libphonenumber: libphonenumber: 7.2.2"

Examples

Patterns + 1 (786) 1234 5678

Dato un numero di telefono normalizzato come +178612345678 otterremo un numero formattato con il modello fornito.

```
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;
}
```

Leggi Formattazione dei numeri di telefono con pattern. online: https://riptutorial.com/it/android/topic/9083/formattazione-dei-numeri-di-telefono-con-pattern-

Capitolo 110: Fornitore di contenuti

Osservazioni

I fornitori di contenuti gestiscono l'accesso a un insieme strutturato di dati. Incapsulano i dati e forniscono meccanismi per la definizione della sicurezza dei dati. I fornitori di contenuti sono l'interfaccia standard che collega i dati in un processo con codice in esecuzione in un altro processo.

Quando si desidera accedere ai dati in un provider di contenuti, utilizzare l'oggetto `ContentResolver` nel `Context` dell'applicazione per comunicare con il provider come client. L'oggetto `ContentResolver` comunica con l'oggetto provider, un'istanza di una classe che implementa `ContentProvider` . L'oggetto provider riceve richieste di dati dai client, esegue l'azione richiesta e restituisce i risultati.

Non è necessario sviluppare il proprio provider se non si intende condividere i dati con altre applicazioni. Tuttavia, è necessario il proprio provider per fornire suggerimenti di ricerca personalizzati nella propria applicazione. È inoltre necessario il proprio provider se si desidera copiare e incollare dati o file complessi dall'applicazione ad altre applicazioni.

Lo stesso Android include fornitori di contenuti che gestiscono dati come audio, video, immagini e informazioni di contatto personali. Puoi vedere alcuni di essi elencati nella documentazione di riferimento per il pacchetto `android.provider` . Con alcune restrizioni, questi provider sono accessibili a qualsiasi applicazione Android.

Examples

Implementazione di una classe di provider di contenuti di base

1) Creare una classe di contratto

Una classe del contratto definisce le costanti che aiutano le applicazioni a lavorare con gli URI del contenuto, i nomi delle colonne, le azioni intent e altre caratteristiche di un fornitore di contenuti. Le classi di contratto non sono incluse automaticamente con un fornitore; lo sviluppatore del provider deve definirli e renderli disponibili ad altri sviluppatori.

Un provider di solito ha una singola autorità, che funge da nome interno Android. Per evitare conflitti con altri fornitori, utilizzare un'autorità di contenuto univoca. Poiché questo consiglio vale anche per i nomi dei pacchetti Android, è possibile definire l'autorizzazione del provider come un'estensione del nome del pacchetto contenente il provider. Ad esempio, se il nome del pacchetto Android è `com.example.appname` , è necessario fornire all'autorità di `com.example.appname.provider` l'autorità `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 di contenuto è un URI che identifica i dati in un provider. Gli URI di contenuto includono il nome simbolico dell'intero provider (la sua autorità) e un nome che punta a una tabella o file (un percorso). La parte id opzionale indica una singola riga in una tabella. Ogni metodo di accesso ai dati di ContentProvider ha un URI di contenuto come argomento; questo ti permette di determinare la tabella, riga o file per accedere. Definire questi nella classe di contratto.

```
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) Crea la classe di supporto

Una classe helper gestisce la creazione del database e la gestione delle versioni.

```
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) Creare una classe che estende la classe 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 associa un'autorità e un percorso a un valore intero. Il metodo `match()` restituisce un valore intero univoco per un URI (può essere qualsiasi numero arbitrario, purché sia univoco).

Un'istruzione switch sceglie tra l'interrogazione dell'intera tabella e l'interrogazione per un singolo record. Il nostro UriMatcher restituisce 100 se l'URI è l'URI contenuto di Tabella e 101 se l'URI punta a una riga specifica all'interno di quella tabella. Puoi usare il `#` carattere jolly per abbinarlo con qualsiasi numero e `*` per abbinarlo a qualsiasi stringa.

```
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: l'ordine delle chiamate ad `addURI()` importante! UriMatcher guarderà in ordine sequenziale dal primo aggiunto all'ultimo. Poiché i caratteri jolly come `#` e `*` sono golosi, dovrai assicurarti di aver ordinato correttamente gli URI. Per esempio:

```
uriMatcher.addURI(CONTENT_AUTHORITY, "/example", 1);
uriMatcher.addURI(CONTENT_AUTHORITY, "/*",       2);
```

è l'ordine corretto, dal momento che il matcher cercherà `/example` prima di ricorrere alla corrispondenza `/*` . Se queste chiamate al metodo sono state annullate e hai chiamato `uriMatcher.match("/example")` , UriMatcher smetterà di cercare le corrispondenze una volta che incontra il percorso `/*` e restituisce il risultato errato!

Dovrai quindi eseguire l'override di queste funzioni:

onCreate () : inizializza il tuo provider. Il sistema Android chiama questo metodo subito dopo aver creato il tuo provider. Si noti che il proprio provider non viene creato fino a quando un oggetto ContentResolver tenta di accedervi.

```
@Override
public boolean onCreate() {
    dbhelper = new DatabaseHelper(getContext());
    return true;
}
```

getType () : restituisce il tipo MIME corrispondente a un URI di contenuto

```
@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);

    }
}
```

query () : recupera i dati dal tuo provider. Utilizzare gli argomenti per selezionare la tabella da interrogare, le righe e le colonne da restituire e l'ordinamento del risultato. Restituisce i dati come oggetto Cursore.

```
@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;
}
```

Inserisci una nuova riga nel tuo provider. Utilizzare gli argomenti per selezionare la tabella di destinazione e ottenere i valori della colonna da utilizzare. Restituisce un URI di contenuto per la riga appena inserita.

```
@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 () : elimina le righe dal tuo provider. Utilizzare gli argomenti per selezionare la tabella e le righe da eliminare. Restituisce il numero di righe cancellate.

```
@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 () : aggiorna le righe esistenti nel tuo provider. Utilizzare gli argomenti per selezionare la tabella e le righe da aggiornare e ottenere i nuovi valori di colonna. Restituisce il numero di righe aggiornate.

```
@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) Aggiorna file manifest

```
<provider
        android:authorities="com.example.myApp"
        android:name=".DatabaseProvider"/>
```

Leggi Fornitore di contenuti online: https://riptutorial.com/it/android/topic/3075/fornitore-di-contenuti

Capitolo 111: frammenti

introduzione

Introduzione sui frammenti e il loro meccanismo di intercomunicazione

Sintassi

- void onActivityCreated (Bundle savedInstanceState) // Chiamato quando l'attività del frammento è stata creata e la gerarchia della vista di questo frammento è stata istanziata.
- void onActivityResult (int requestCode, int resultCode, Intent data) // Ricevi il risultato da una chiamata precedente a startActivityForResult (Intent, int).
- void onAttach (Activity activity) // Questo metodo è stato deprecato nel livello API 23. Utilizzare invece onAttach (Context).
- void onAttach (Contesto contesto) // Chiamato quando un frammento viene prima collegato al suo contesto.
- void onAttachFragment (Fragment childFragment) // Chiamato quando un frammento è collegato come figlio di questo frammento.
- void onConfigurationChanged (Configuration newConfig) // Chiamato dal sistema quando la configurazione del dispositivo cambia mentre il componente è in esecuzione.
- void onCreate (Bundle savedInstanceState) // Chiamato per fare la creazione iniziale di un frammento.
- Visualizza onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) // Chiamato per fare in modo che il frammento istanzia la sua vista dell'interfaccia utente.
- void onDestroy () // Chiamato quando il frammento non è più in uso.
- void onDestroyView () // Chiamato quando la vista precedentemente creata da onCreateView (LayoutInflater, ViewGroup, Bundle) è stata scollegata dal frammento.
- void onDetach () // Chiamato quando il frammento non è più associato alla sua attività.
- void onInflate (Activity activity, AttributeSet attrs, Bundle savedInstanceState) // Questo metodo è stato deprecato nel livello API 23. Utilizza invece onInflate (Context, AttributeSet, Bundle).
- void onInflate (Contesto contesto, AttributeSet attrs, Bundle savedInstanceState) // Chiamato quando un frammento viene creato come parte di un'inflazione del layout della vista, in genere dall'impostazione della visualizzazione del contenuto di un'attività.

- void onPause () // Chiamato quando il frammento non viene più ripreso.
- void onResume () // Chiamato quando il frammento è visibile all'utente e in esecuzione attiva.
- void onSaveInstanceState (Bundle outState) // Chiamato per chiedere al frammento di salvare il suo stato dinamico corrente, in modo che possa essere ricostruito in un secondo momento in modo che il nuovo processo venga riavviato.
- void onStart () // Chiamato quando Fragment è visibile all'utente.
- void onStop () // Chiamato quando Fragment non è più avviato.
- void onViewStateRestored (Bundle savedInstanceState) // Chiamato quando tutti gli stati salvati sono stati ripristinati nella gerarchia della vista del frammento.

Osservazioni

Un frammento rappresenta un comportamento o una parte dell'interfaccia utente in un'attività. È possibile combinare più frammenti in un'unica attività per creare un'interfaccia utente a più riquadri e riutilizzare un frammento in più attività. Puoi pensare a un frammento come a una sezione modulare di un'attività, che ha il suo ciclo di vita, riceve i propri eventi di input e che puoi aggiungere o rimuovere mentre l'attività è in esecuzione (una specie di "attività secondaria" che puoi riutilizzo in diverse attività).

Costruttore

Ogni frammento deve avere un **costruttore vuoto** , quindi può essere istanziato quando si ripristina lo stato della propria attività. Si raccomanda vivamente che le sottoclassi non abbiano altri costruttori con parametri, poiché questi costruttori non saranno chiamati quando il frammento viene riattivato; invece, gli argomenti possono essere forniti dal chiamante con setArguments (Bundle) e successivamente recuperati da Fragment con getArguments ().

Examples

Il modello newInstance ()

Sebbene sia possibile creare un costruttore di frammenti con parametri, Android richiama internamente il costruttore di argomenti zero durante la ricreazione di frammenti (ad esempio, se vengono ripristinati dopo essere stati uccisi per motivi personali di Android). Per questo motivo, non è consigliabile fare affidamento su un costruttore che ha parametri.

Per garantire che gli argomenti dei frammenti previsti siano sempre presenti, è possibile utilizzare un metodo `newInstance()` statico per creare il frammento e inserire i parametri desiderati in un pacchetto che sarà disponibile quando si crea una nuova istanza.

```
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);
    }
  }
}
```

Ora, nell'attività:

```
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();
```

Questo modello è una best practice per garantire che tutti gli argomenti necessari vengano passati ai frammenti della creazione. Si noti che quando il sistema distrugge il frammento e lo ricrea in seguito, ripristinerà automaticamente il suo stato, ma è necessario fornirlo con `onSaveInstanceState(Bundle)` .

Navigazione tra i frammenti usando il backstack e il modello di tessuto statico

Prima di tutto, dobbiamo aggiungere il nostro primo `Fragment` all'inizio, dovremmo farlo nel metodo `onCreate()` della nostra attività:

```
if (null == savedInstanceState) {
    getSupportFragmentManager().beginTransaction()
      .addToBackStack("fragmentA")
      .replace(R.id.container, FragmentA.newInstance(), "fragmentA")
      .commit();
}
```

Successivamente, dobbiamo gestire il nostro backstack. Il modo più semplice è utilizzare una funzione aggiunta nella nostra attività che viene utilizzata per tutte le FragmentTransactions.

```
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();
}
```

Infine, dovremmo sovrascrivere `onBackPressed()` per uscire dall'applicazione quando torno dall'ultimo frammento disponibile nel 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();
    }
}
```

Esecuzione in attività:

```
replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Esecuzione all'esterno dell'attività (presupponendo che `MainActivity` sia la nostra attività):

```
((MainActivity) getActivity()).replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Passa i dati da Attività a frammento usando Bundle

Tutti i frammenti dovrebbero avere un costruttore vuoto (cioè un metodo di costruzione che non ha argomenti di input). Pertanto, per passare i tuoi dati al Frammento in fase di creazione, devi utilizzare il metodo `setArguments()` . Questo metodo ottiene un pacchetto in cui vengono archiviati i dati e archivia il pacchetto negli argomenti. Successivamente, questo pacchetto può quindi essere richiamato nelle `onCreate()` e `onCreateView()` di Fragment.

Attività:

```
 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();
```

Frammento:

```
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    String myValue = this.getArguments().getString("message");
    ...
 }
```

Invio di eventi a un'attività con interfaccia di callback

Se è necessario inviare eventi dal frammento all'attività, una delle possibili soluzioni è definire l'interfaccia di callback e richiedere che l'attività dell'host lo implementa.

Esempio

Invia callback a un'attività, quando si fa clic sul pulsante di frammento

Prima di tutto, definisci l'interfaccia di callback:

```
public interface SampleCallback {
    void onButtonClicked();
```

```
}
```

Il prossimo passo è assegnare questo callback in frammento:

```
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;
    }
}
```

Infine, implementa la funzione di callback nell'attività:

```
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
    }
}
```

Animare la transizione tra i frammenti

Per animare la transizione tra i frammenti, o per animare il processo di mostrare o nascondere un frammento, si usa `FragmentManager` per creare un `FragmentTransaction` .

Per una singola `FragmentTransaction` , ci sono due modi diversi per eseguire le animazioni: puoi usare un'animazione standard o puoi fornire le tue animazioni personalizzate.

Le animazioni standard vengono specificate chiamando `FragmentTransaction.setTransition(int transit)` e utilizzando una delle costanti predefinite disponibili nella classe `FragmentTransaction` . Al momento della scrittura, queste costanti sono:

```
FragmentTransaction.TRANSIT_NONE
FragmentTransaction.TRANSIT_FRAGMENT_OPEN
FragmentTransaction.TRANSIT_FRAGMENT_CLOSE
FragmentTransaction.TRANSIT_FRAGMENT_FADE
```

La transazione completa potrebbe essere simile a questa:

```
getSupportFragmentManager()
    .beginTransaction()
    .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

Le animazioni personalizzate vengono specificate chiamando `FragmentTransaction.setCustomAnimations(int enter, int exit)` o `FragmentTransaction.setCustomAnimations(int enter, int exit, int popEnter, int popExit)` .

Le animazioni di `enter` e `exit` verranno riprodotte per `FragmentTransaction` s che non implicano frammenti fuori dallo stack posteriore. Le animazioni `popEnter` e `popExit` verranno riprodotte quando si inserisce un frammento fuori dallo stack posteriore.

Il codice seguente mostra come sostituire un frammento facendo scorrere un frammento e facendo scorrere l'altro al suo posto.

```
getSupportFragmentManager()
    .beginTransaction()
    .setCustomAnimations(R.anim.slide_in_left, R.anim.slide_out_right)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

Le definizioni dell'animazione XML utilizzano il tag `objectAnimator` . Un esempio di **slide_in_left.xml** potrebbe essere qualcosa del genere:

```
<?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>
```

Comunicazione tra frammenti

Tutte le comunicazioni tra i frammenti devono passare per un'attività. I frammenti **NON possono** comunicare tra loro senza un'attività.

Risorse addizionali

- Come implementare OnFragmentInteractionListener
- Android | Comunicare con altri frammenti

In questo esempio, abbiamo un `MainActivity` che ospita due frammenti, `SenderFragment` e `ReceiverFragment` , per l'invio e la ricezione di un `message` (una stringa semplice in questo caso) rispettivamente.

Un pulsante in `SenderFragment` avvia il processo di invio del messaggio. Una TextView in `ReceiverFragment` viene aggiornata quando il messaggio viene ricevuto da esso.

Di seguito è riportato lo snippet per MainActivity con i commenti che spiegano le importanti righe di codice:

```
// 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);
    }
}
}
```

Il file di layout per `MainActivity` ospita due frammenti all'interno di 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>
```

`SenderFragment` espone un'interfaccia `SendMessageListener` che aiuta il `MainActivity` sapere quando è stato fatto clic su Button in `SenderFragment` .

Di seguito è riportato lo snippet di codice per `SenderFragment` spiega le importanti righe di codice:

```
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;
}
}
```

Il file di layout per `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` è semplice ed espone un semplice metodo pubblico per l'aggiornamento del TextView. Quando `MainActivity` riceve il messaggio da `SenderFragment` , chiama questo metodo pubblico di `ReceiverFragment`

Di seguito è riportato lo snippet di codice per `ReceiverFragment` con i commenti che spiegano le linee importanti di codice:

```
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);
}
}
```

Il file di layout per `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>
```

Leggi frammenti online: https://riptutorial.com/it/android/topic/1396/frammenti

Capitolo 112: Genymotion per Android

introduzione

Genymotion è un veloce emulatore di terze parti che può essere utilizzato al posto dell'emulatore Android predefinito. In alcuni casi è buono o migliore rispetto allo sviluppo su dispositivi reali!

Examples

Installazione di Genymotion, la versione gratuita

Passaggio 1: installazione di `VirtualBox`

Scarica e installa VirtualBox in base al tuo sistema operativo. , è necessario per eseguire `Genymotion` .

Passaggio 2: download di `Genymotion`

Vai alla pagina di download di `Genymotion` e scarica `Genymotion` base al tuo sistema operativo.

Nota: sarà necessario creare un nuovo account OPPURE accedere con il proprio account.

Passaggio 3: installazione di `Genymotion`

se su `Linux` fai riferimento a questa risposta , installa ed esegui un file `.bin` .

Passaggio 4 - Installazione degli emulatori di `Genymotion`

- eseguire `Genymotion`
- Premi il pulsante Aggiungi (nella barra in alto).
- Accedi con il tuo account e sarai in grado di navigare tra gli emulatori disponibili.
- seleziona e installa quello che ti serve.

Passaggio 5: integrazione della `genymotion` con `Android Studio`

`Genymotion` , può essere integrato con `Android Studio` tramite un plugin, qui i passaggi per installarlo su `Android Studio`

- vai su File / Impostazioni (per Windows e Linux) o su Android Studio / Preferenze (per Mac

OS X)
- Seleziona Plugin e fai clic su Sfoglia repository.
- Fai clic con il tasto destro del mouse su Genymotion e fai clic su Scarica e installa.

Ora dovresti essere in grado di vedere l'icona del plugin, vedere questa immagine

Nota, potresti voler visualizzare la barra degli strumenti facendo clic su Visualizza> Barra degli strumenti.

Passaggio 6: esecuzione di `Genymotion` da `Android Studio`

- vai su File / Impostazioni (per Windows e Linux) o su Android Studio / Preferenze (per Mac OS X)
- vai su Other Settings / Genymotion e aggiungi il percorso della cartella di `Genymotion's` e applica le tue modifiche.

Ora dovresti essere in grado di eseguire `Genymotion's` emulatore `Genymotion's` premendo l'icona del plugin e selezionando un emulatore installato e poi premi il pulsante di avvio!

Quadro Google su Genymotion

Se gli sviluppatori vogliono testare Google Maps o qualsiasi altro servizio Google come Gmail, Youtube, Google Drive, ecc., Devono prima installare Google framework su Genymotion. Ecco i passaggi: -

4.4 Kitkat
5.0 Lollipop
5.1 Lecca lecca
6.0 Marshmallow
7.0 Nougat
7.1 Nougat (patch webview)

1. Scarica dal link qui sopra
2. Basta trascinare e rilasciare il file zip scaricato su genymotion e riavviare
3. Aggiungi l'account Google e scarica "Google Play Music" ed esegui.

Riferimento:-
Domanda di overflow dello stack su questo argomento

Leggi Genymotion per Android online: https://riptutorial.com/it/android/topic/9245/genymotion-per-android

Capitolo 113: Gestione degli eventi di tocco e movimento

introduzione

Un riepilogo di alcuni dei sistemi touch / di gestione del movimento di base nell'API Android.

Parametri

Ascoltatore	Dettagli
onTouchListener	Gestisce i singoli tocchi per pulsanti, superfici e altro
onTouchEvent	Un ascoltatore che può essere trovato nelle superfici (ad es. SurfaceView). Non ha bisogno di essere impostato come altri listener (e, su onTouchListener)
onLongTouch	Simile a onTouch, ma ascolta le lunghe pressioni su pulsanti, superfici e altro.

Examples

pulsanti

Gli eventi di tocco relativi a un `Button` possono essere controllati come segue:

```
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

Toccare il gestore di eventi per le superfici (ad es. SurfaceView, GLSurfaceView e altri):

```
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 in alternativa (in 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;
    }
}
```

Manipolazione multitouch in 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.
    }
}
```

Leggi Gestione degli eventi di tocco e movimento online:
https://riptutorial.com/it/android/topic/9315/gestione-degli-eventi-di-tocco-e-movimento

Capitolo 114: Gestire i collegamenti profondi

introduzione

I link diretti sono URL che portano gli utenti direttamente a contenuti specifici nella tua app. Puoi impostare i link diretti aggiungendo filtri di intent ed estraendo i dati dagli intenti in entrata per indirizzare gli utenti alla schermata giusta nella tua app.

Parametri

`<data>` Attributo	Dettagli
schema	La parte di *schema* di un URI (sensibile al maiuscolo / minuscolo). Esempi: `http` , `https` , `ftp`
ospite	La parte *host* di un URI (sensibile al maiuscolo / minuscolo). Esempi: `google.com` , `example.org`
porta	La parte di *porta* di un URI. Esempi: `80` , `443`
sentiero	La parte del *percorso* di un URI. Deve iniziare con `/` . Esempi: `/` , `/about`
Pathprefix	Un prefisso per la parte del *percorso* di un URI. Esempi: `/item` , `/article`
pathPattern	Un modello da abbinare per la parte del *percorso* di un URI. Esempi: `/item/.*` , `/article/[0-9]*`
mimeType	Un tipo mime da abbinare. Esempi: `image/jpeg` , `audio/*`

Osservazioni

`<intent-filter>`

Questa combinazione di elementi `<action>` e `<category>` è ciò che dice al sistema Android che deve essere lanciata un'attività specifica quando l'utente fa clic su un link in un'altra applicazione.

```
<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>
```

Più tag `<data>`

La serie di link diretti supportati da `<intent-filter>` è il prodotto incrociato di tutti gli elementi `<data>` definiti in tale filtro intent. Il dominio multiplo, il percorso multiplo e più esempi di schemi lo dimostrano.

risorse

- Abilitazione di collegamenti profondi per il contenuto dell'app (developer.android.com)
- `<intent-filter>` (developer.android.com

Examples

Semplice link diretto

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>
```

Ciò accetterà qualsiasi collegamento che inizia con `http://www.example.com` come link `MainActivity` per avviare `MainActivity` .

Percorsi multipli su un singolo 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>
```

Questo avvierà `MainActivity` quando l'utente fa clic su uno di questi collegamenti:

- `http://www.example.com/`
- `http://www.example.com/about`
- `http://www.example.com/map`

Domini multipli e percorsi multipli

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>
```

Questo avvierà MainActivity quando l'utente fa clic su uno di questi collegamenti:

- `http://www.example.com/`
- `http://www.example2.com/`
- `http://www.example.com/map`
- `http://www.example2.com/map`

Sia http e https per lo stesso 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>
```

Questo avvierà MainActivity quando l'utente fa clic su uno di questi collegamenti:

- `http://www.example.com/`
- `https://www.example.com/`
- `http://www.example.com/map`
- `https://www.example.com/map`

Recupero dei parametri di query

```
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");
        }
    }

}
```

Se l'utente fa clic su un link a `http://www.example.com/map?param1=FOO&param2=BAR` , quindi `param1` qui avrà un valore di `"FOO"` e `param2` avrà un valore di `"BAR"` .

Utilizzando 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>
```

Questo avvierà la tua MainActivity quando l'utente fa clic su qualsiasi link a partire da `http://www.example.com/item` , ad esempio:

- `https://www.example.com/item`
- `http://www.example.com/item/1234`
- `https://www.example.com/item/xyz/details`

Leggi Gestire i collegamenti profondi online: https://riptutorial.com/it/android/topic/3716/gestire-i-collegamenti-profondi

Capitolo 115: Google Play Store

Examples

Apri l'elenco di Google Play Store per la tua app

Il seguente frammento di codice mostra come aprire il listato Google Play Store della tua app in modo sicuro. Di solito lo si vuole usare quando si chiede all'utente di lasciare una recensione per la propria app.

```
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 : il codice apre Google Play Store se l'app è installata. Altrimenti aprirà semplicemente il browser web.

Apri Google Play Store con l'elenco di tutte le applicazioni dal tuo account publisher

Puoi aggiungere un pulsante "Sfoglia le altre nostre app" nella tua app, elencando tutte le tue applicazioni (dell'editore) nell'app 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);
    }
}
```

Leggi Google Play Store online: https://riptutorial.com/it/android/topic/10900/google-play-store

Capitolo 116: Gradle per Android

introduzione

Gradle è un sistema di build basato su JVM che consente agli sviluppatori di scrivere script di alto livello che possono essere utilizzati per automatizzare il processo di compilazione e produzione di applicazioni. È un sistema flessibile basato su plugin, che consente di automatizzare vari aspetti del processo di creazione; inclusi la compilazione e la firma di un file `.jar` , il download e la gestione di dipendenze esterne, l'inserimento di campi `AndroidManifest` o l'utilizzo di versioni specifiche dell'SDK.

Sintassi

- `apply plugin` : i plugin che dovrebbero essere usati normalmente solo `'com.android.application'` o `'com.android.library'` .
- `android` : la configurazione principale della tua app
    - `compileSdkVersion` : la versione di compilazione dell'SDK
    - `buildToolsVersion` : la versione degli strumenti di compilazione
    - `defaultConfig` : le impostazioni predefinite che possono essere sovrascritte da sapori e tipi di build
        - `applicationId` : l'ID dell'applicazione che usi, ad esempio, nel Play Store, quasi uguale al nome del tuo pacchetto
        - `minSdkVersion` : la versione minima richiesta dell'SDK
        - `targetSdkVersion` : la versione dell'SDK con cui si compila (dovrebbe essere sempre la nuova)
        - `versionCode` : il numero di versione interno che deve essere più grande su ciascun aggiornamento
        - `versionName` : il numero di versione che l'utente può vedere nella pagina dei dettagli dell'app
    - `buildTypes` : guarda da qualche altra parte (TODO)
- `dependencies` : le `dependencies` Maven o locali della tua app
    - `compile` una singola dipendenza
    - `testCompile` : una dipendenza per l'unità o test di integrazione

Osservazioni

Guarda anche

- La homepage ufficiale di gradle
- Come configurare build gradle
- Il plugin Android per gradle

- Android Gradle DSL

Gradle per Android - Documentazione estesa:

C'è un altro tag dove puoi trovare più argomenti ed esempi sull'uso di gradle in Android.
http://www.riptutorial.com/topic/2092

Examples

Un file build.gradle di base

Questo è un esempio di un file `build.gradle` predefinito in un modulo.

```
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 (linguaggio specifico del dominio)

Ogni blocco nel file sopra è chiamato `DSL` (linguaggio specifico del dominio).

---

plugin

La prima riga, `apply plugin: 'com.android.application'` , applica il plug-in Android per Gradle alla build e rende disponibile il blocco `android {}` per dichiarare le opzioni di generazione specifiche per Android.

Per un'applicazione **Android** :

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

Per una **libreria Android** :

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

Comprendere i DSL nell'esempio sopra

La seconda parte, il blocco `android {...}` , è il `DSL` Android che contiene informazioni sul tuo progetto.

Ad esempio, puoi impostare `compileSdkVersion` che specifica il livello dell'API Android, che dovrebbe essere usato da Gradle per compilare la tua app.
Il sub-blocco `defaultConfig` contiene i valori predefiniti per il manifest. Puoi `override` con gli aromi del prodotto .

Puoi trovare maggiori informazioni in questi esempi:

- DSL per il modulo dell'app
- Tipi di costruzione
- Aromi del prodotto
- Impostazioni di firma

---

dipendenze

Il blocco delle `dependencies` è definito al di fuori del blocco `android {...}` : ciò significa che non è definito dal plug-in Android ma è Gradle standard.
Il blocco delle `dependencies` specifica quali librerie esterne (tipicamente le librerie Android, ma anche le librerie Java sono valide) che desideri includere nella tua app. Gradle scaricherà automaticamente queste dipendenze per te (se non è disponibile una copia locale), devi semplicemente aggiungere linee di `compile` simili quando desideri aggiungere un'altra libreria.

Diamo un'occhiata a una delle righe qui presenti:

```
compile 'com.android.support:design:25.3.1'
```

Questa linea dice fondamentalmente

> aggiungi una dipendenza dalla libreria di progettazione del supporto Android al mio progetto.

Gradle farà in modo che la libreria sia scaricata e presente in modo che tu possa usarla nella tua app e il suo codice sarà incluso anche nella tua app.

Se hai familiarità con Maven, questa sintassi è *GroupId* , due punti, *ArtifactId* , un altro punto, quindi la versione della dipendenza che desideri includere, dandoti il pieno controllo sul controllo delle versioni.

Mentre è possibile specificare le versioni di artefatto usando il segno più (+), la migliore pratica è evitare di farlo; può portare a problemi se la libreria viene aggiornata con interruzioni di modifica a tua insaputa, il che probabilmente causerebbe arresti anomali nella tua app.

Puoi aggiungere diversi tipi di dipendenze:

- dipendenze binarie locali
- dipendenze del modulo
- dipendenze remote

Un'attenzione particolare dovrebbe essere dedicata alle dipendenze piatte .

Puoi trovare maggiori dettagli in questo argomento.

Nota su **-v7 in *appcompat-v7***

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

Ciò significa semplicemente che questa **libreria** (`appcompat`) è compatibile con il livello API Android 7 e versioni successive.

Nota sulla **junit: junit: 4.12**

Questa è la dipendenza del test per il test dell'unità.

Specifica delle dipendenze specifiche per diverse configurazioni di build

È possibile specificare che una dipendenza deve essere utilizzato solo per un determinato configurazione di generazione oppure è possibile definire diverse dipendenze per i tipi di compilazione o dei sapori di prodotti (ad esempio, eseguire il debug, test o di rilascio) utilizzando `debugCompile` , `testCompile` o `releaseCompile` al posto della solita `compile` .

Questo è utile per mantenere le dipendenze test-debug-correlate fuori dalla build di rilascio, che manterrà il tuo `APK` release il più sottile possibile e ti aiuterà a garantire che qualsiasi informazione di debug non possa essere utilizzata per ottenere informazioni interne sulla tua app.

---

signingConfig

`signingConfig` consente di configurare Gradle in modo da includere le informazioni sul `keystore` e garantire che l'APK creato utilizzando queste configurazioni sia firmato e pronto per la versione Play Store.

Qui puoi trovare un argomento dedicato .

Nota : non è tuttavia consigliabile mantenere le credenziali di firma all'interno del file Gradle. Per rimuovere le configurazioni di firma, basta omettere la porzione `signingConfigs` .
Puoi specificarli in diversi modi:

- memorizzazione in un file esterno
- memorizzandoli nell'impostazione delle variabili d'ambiente .

Vedi questo argomento per maggiori dettagli: Firma APK senza esporre la password del keystore .

Puoi trovare ulteriori informazioni su Gradle per Android nell'argomento dedicato di Gradle .

Definire i sapori del prodotto

I sapori del prodotto sono definiti nel file `build.gradle` all'interno del blocco di `android { ... }` come mostrato di seguito.

```
...
android {
    ...
    productFlavors {
        free {
            applicationId "com.example.app.free"
            versionName "1.0-free"
        }
        paid {
            applicationId "com.example.app.paid"
            versionName "1.0-paid"
        }
    }
}
```

In questo modo, ora disponiamo di due ulteriori prodotti: `free` e a `paid` . Ciascuno può avere la propria configurazione e attributi specifici. Ad esempio, entrambi i nostri nuovi aromi hanno un `applicationId` e versioneName `versionName` rispetto al nostro aroma `main` esistente (disponibile per impostazione predefinita, quindi non mostrato qui).

Aggiunta di dipendenze specifiche per il gusto del prodotto

Le dipendenze possono essere aggiunte per un determinato gusto del prodotto , in modo simile a come possono essere aggiunte per configurazioni di build specifiche.

Per questo esempio, supponiamo di aver già definito due aromi di prodotto chiamati `free` e a `paid` (più sulla definizione degli aromi qui).
Possiamo quindi aggiungere la dipendenza di AdMob per l'aroma `free` e la libreria di Picasso per quella a `paid` modo:

```
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'
}
...
```

Aggiunta di risorse specifiche per il gusto del prodotto

Le risorse possono essere aggiunte per un determinato gusto del prodotto .

Per questo esempio, supponiamo di aver già definito due sapori del prodotto chiamati `free` e a `paid` . Per aggiungere risorse specifiche per il gusto del prodotto, creiamo ulteriori cartelle di risorse accanto alla cartella `main/res` , che possiamo aggiungere come al solito. Per questo esempio, definiremo una stringa, lo `status` , per ogni gusto del prodotto:

*/ 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>
```

Le stringhe di `status` specifiche per l'aroma del prodotto sostituiranno il valore per lo `status` nel gusto `main` .

Definire e utilizzare i Campi configurazione build

BuildConfigField

Gradle consente `buildConfigField` linee `buildConfigField` di definire le costanti. Queste costanti saranno accessibili in runtime come campi statici della classe `BuildConfig` . Questo può essere usato per creare aromi definendo tutti i campi all'interno del blocco `defaultConfig` , quindi sovrascrivendolo per i singoli sapori di build, se necessario.

Questo esempio definisce la data di costruzione e contrassegna la build per la produzione anziché il test:

```
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'
        }
    }
}
```

Il <nome_pacchetto> generato automaticamente. `BuildConfig` .java nella cartella gen contiene i seguenti campi basati sulla direttiva sopra:

```
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";
}
```

I campi definiti possono ora essere utilizzati all'interno dell'app in fase di runtime accedendo alla classe `BuildConfig` generata:

```
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();
    }
}
```

ResValue

Il `resValue` nei `productFlavors` crea un valore di risorsa. Può essere qualsiasi tipo di risorsa (`string` , `dimen` , `color` , ecc.). Questo è simile alla definizione di una risorsa nel file appropriato: ad es. `strings.xml` una stringa in un file `strings.xml` . Il vantaggio è che quello definito in gradle può essere modificato in base al prodotto Flavor / buildVariant. Per accedere al valore, scrivi lo stesso codice come se tu stessi accedendo a una res dal file delle risorse:

```
getResources().getString(R.string.app_name)
```

L'importante è che le risorse definite in questo modo non possano modificare le risorse esistenti definite nei file. Possono solo creare nuovi valori di risorse.

Alcune librerie (come l'API Android di Google Maps) richiedono una chiave API fornita nel Manifesto come tag dei `meta-data` . Se sono necessarie chiavi diverse per il debug e le build di produzione, specificare un segnaposto manifest compilato da Gradle.

Nel tuo file `AndroidManifest.xml` :

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="${MAPS_API_KEY}"/>
```

Quindi imposta il campo di conseguenza nel file `build.gradle` :

```
android {
```

```
    defaultConfig {
        ...
        // Your development key
        manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
    }

    productFlavors {
        prod {
            // Your production key
            manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
        }
    }
}
```

Il sistema di build Android genera automaticamente un numero di campi e li inserisce in `BuildConfig.java` . Questi campi sono:

Campo	Descrizione
`DEBUG`	un `Boolean` indica se l'app è in modalità di debug o di rilascio
`APPLICATION_ID`	una `String` contenente l'ID dell'applicazione (es. `com.example.app`)
`BUILD_TYPE`	una `String` contenente il tipo di build dell'applicazione (di solito sia il `debug` che il `release`)
`FLAVOR`	una `String` contiene il particolare sapore della build
`VERSION_CODE`	un `int` contenente il numero della versione (build). È lo stesso di `versionCode` in `build.gradle` o `versionCode` in `AndroidManifest.xml`
`VERSION_NAME`	una `String` contenente il nome della versione (build). È lo stesso di `versionName` in `build.gradle` o `versionName` in `AndroidManifest.xml`

In aggiunta a quanto sopra, se hai definito più dimensioni di sapore, ciascuna dimensione avrà il suo valore. Ad esempio, se hai due dimensioni di aroma per `color` e `size` avrai anche le seguenti variabili:

Campo	Descrizione
`FLAVOR_color`	una `String` contenente il valore per il sapore "colore".
`FLAVOR_size`	una `String` contenente il valore per il sapore "taglia".

Centralizzazione delle dipendenze tramite il file "dependencies.gradle"

Quando si lavora con progetti multi-modulo, è utile centralizzare le dipendenze in una singola posizione piuttosto che distribuirle su molti file di build, specialmente per le librerie comuni come le librerie di supporto Android e le librerie Firebase .

Un modo consigliato è quello di separare i file di build di Gradle, con un `build.gradle` per modulo, nonché uno nella root del progetto e un altro per le dipendenze, ad esempio:

```
root
  +- gradleScript/
  |     dependencies.gradle
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
```

Quindi, tutte le tue dipendenze possono trovarsi in `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}",
    ];
}
```

Quale può quindi essere applicato da quel file nel file di livello superiore `build.gradle` modo:

```
// Load dependencies
apply from: 'gradleScript/dependencies.gradle'
```

e nel `module1/build.gradle` modo:

```
// Module build file
dependencies {
    // ...
    compile supportDependencies.appCompat
    compile supportDependencies.design
```

```
    compile firebaseDependencies.crash
}
```

Un altro approccio

Un approccio meno dettagliato per la centralizzazione delle versioni delle dipendenze della libreria può essere ottenuto dichiarando il numero di versione come variabile una volta e utilizzandolo ovunque.

Nell'area di lavoro root `build.gradle` aggiungere questo:

```
ext.v = [
    supportVersion:'24.1.1',
]
```

E in ogni modulo che usa la stessa libreria aggiungi le librerie necessarie

```
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}"
```

Struttura della directory per risorse specifiche per il gusto

Diversi tipi di build dell'applicazione possono contenere risorse diverse. Per creare una risorsa specifica per il gusto, crea una directory con il nome in minuscolo del tuo gusto nella directory `src` e aggiungi le tue risorse nello stesso modo in cui faresti normalmente.

Ad esempio, se si dispone di uno `Development` sapore e si desidera fornire un'icona di avvio distinta per esso, si creerà una directory `src/development/res/drawable-mdpi` e all'interno di tale directory creare un file `ic_launcher.png` con l'icona specifica per lo sviluppo.

La struttura della directory sarà simile a questa:

```
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'
```

(Naturalmente, in questo caso si creerebbero anche icone per disegnabili-hdpi, disegnabili-xhdpi, *ecc* .).

Perché ci sono due file build.gradle in un progetto Android Studio?

`<PROJECT_ROOT>\app\build.gradle` è specifico per il **modulo app** .

`<PROJECT_ROOT>\build.gradle` è un **"file di build di livello superiore" in** cui è possibile aggiungere opzioni di configurazione comuni a tutti i sottoprogetti / moduli.

Se utilizzi un altro modulo nel tuo progetto, come libreria locale avresti un altro file `build.gradle` : `<PROJECT_ROOT>\module\build.gradle`

Nel file di livello superiore è possibile specificare proprietà comuni come blocco di script o alcune proprietà comuni.

```
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"
}
```

Nell'app `app\build.gradle` si definiscono solo le proprietà per il modulo:

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


android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion
}

dependencies {
    //.....
}
```

Esecuzione di uno script di shell da gradle

Uno script di shell è un modo molto versatile per estendere la tua build a praticamente qualsiasi cosa tu possa pensare.

Come un exmaple, ecco un semplice script per compilare i file di protobuf e aggiungere i file java risultanti alla directory di origine per un'ulteriore compilazione:

```
def compilePb() {
    exec {
        // NOTICE: gradle will fail if there's an error in the protoc file...
        executable "../pbScript.sh"
    }
}

project.afterEvaluate {
```

```
    compilePb()
}
```

Lo script di shell 'pbScript.sh' per questo esempio, che si trova nella cartella principale del progetto:

```
#!/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
```

Eseguire il debug degli errori di Gradle

Quello che segue è un estratto di Gradle - Cos'è un valore di uscita diverso da zero e come lo risolvo? , guardalo per la discussione completa.

Diciamo che stai sviluppando un'applicazione e ottieni un errore di Gradle che sembra che in genere sarà simile a quello.

```
: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
```

Cerca qui StackOverflow per il tuo problema e le persone dicono di pulire e ricostruire il tuo progetto, o abilitare MultiDex , e quando lo provi, non risolve il problema.

Ci sono modi per ottenere maggiori informazioni , ma l'output Gradle stesso dovrebbe puntare all'errore effettivo nelle poche righe sopra quel messaggio tra: `module:someTask FAILED` e l'ultimo `:module:someOtherTask` che è passato. Pertanto, se fai una domanda sul tuo errore, modifica le tue domande per includere più contesto nell'errore.

Quindi, ottieni un "valore di uscita diverso da zero". Bene, quel numero è un buon indicatore di ciò che dovresti provare a sistemare. Qui ci sono alcuni che si verificano più frequentemente.

- `1` è solo un codice di errore generale e l'errore è probabile nell'uscita Gradle
- `2` sembra essere correlato a dipendenze sovrapposte o errata configurazione del progetto.
- `3` sembra che includa troppe dipendenze o un problema di memoria.

Le soluzioni generali per quanto sopra (dopo aver tentato di pulire e ricostruire il progetto) sono:

- `1` - Risolvi l'errore menzionato. Generalmente, questo è un errore in fase di compilazione, nel senso che parte del codice nel tuo progetto non è valida. Questo include sia XML che Java per un progetto Android.
- `2` e `3` - Molte risposte qui ti dicono di abilitare il multidex . Mentre può risolvere il problema, è molto probabile una soluzione alternativa. Se non capisci perché lo stai usando (vedi il link), probabilmente non ne hai bisogno. Le soluzioni generali comportano un taglio eccessivo delle dipendenze della libreria (come tutti i servizi di Google Play, quando è necessario utilizzare solo una libreria, ad esempio Maps o Accesso, ad esempio).

Specifica di diversi ID di applicazione per tipi di build e aromi di prodotto

È possibile specificare gli ID di applicazione diversi o nomi dei pacchetti per ogni `buildType` o `productFlavor` utilizzando l'attributo di configurazione **applicationIdSuffix:**

Esempio di `buildType applicationId` per ogni `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"
    }
}
```

I nostri `applicationIds` risultanti ora sarebbero:

- com.package.android per il `release`
- com.package.android. **dev** per lo `development`
- com.package.android. **qa** per il `testing`

Questo può essere fatto anche per `productFlavors` :

```
productFlavors {
    free {
        applicationIdSuffix ".free"
    }
    paid {
        applicationIdSuffix ".paid"
```

```
    }
}
```

Le `applicationIds` risultanti sarebbero:

- com.package.android. **gratis** per il gusto `free`
- com.package.android. **pagato** per il sapore `paid`

Firma APK senza esporre la password del keystore

È possibile definire la configurazione della firma per firmare l'apk nel file `build.gradle` utilizzando queste proprietà:

- `storeFile` : il file keystore
- `storePassword` : la password del keystore
- `keyAlias` : un nome alias chiave
- `keyPassword` : una password alias chiave

In molti casi potrebbe essere necessario evitare questo tipo di informazioni nel file `build.gradle` .

Metodo A: Configurare la firma di rilascio utilizzando un file keystore.properties

È possibile configurare `build.gradle` modo che legga le informazioni sulla configurazione della firma da un file delle proprietà come `keystore.properties` .

Impostare la firma in questo modo è utile perché:

- Le informazioni sulla configurazione della firma sono separate dal file `build.gradle`
- Non è necessario intervenire durante la procedura di firma per fornire password per il file del keystore
- È possibile escludere facilmente il file `keystore.properties` dal controllo di versione

Per prima cosa, crea un file chiamato `keystore.properties` nella radice del tuo progetto con contenuto come questo (sostituendo i valori con il tuo):

```
storeFile=keystore.jks
storePassword=storePassword
keyAlias=keyAlias
keyPassword=keyPassword
```

Ora, nel file `build.gradle` della tua app, imposta il blocco `signingConfigs` come segue:

```
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']
            }
        }
    }
}
```

Questo è davvero tutto quello che c'è da fare, **ma non dimenticare di escludere sia il file `keystore.properties` file `keystore.properties` dal controllo di versione** .

Un paio di cose da notare:

- Il percorso del file `storeFile` specificato nel file `keystore.properties` deve essere relativo al file `build.gradle` dell'app. In questo esempio si presuppone che il file keystore si trovi nella stessa directory del file `build.gradle` dell'app.
- Questo esempio ha il file `keystore.properties` nella radice del progetto. Se lo metti da qualche altra parte, assicurati di cambiare il valore in `rootProject.file('keystore.properties')` nella posizione del tuo, rispetto alla radice del tuo progetto.

Metodo B: utilizzando una variabile di ambiente

Lo stesso può essere ottenuto anche senza un file di proprietà, rendendo la password più difficile da trovare:

```
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 variabile di ambiente `"ps"` può essere globale, ma un approccio più sicuro può essere aggiungendolo alla shell di Android Studio.
In Linux questo può essere fatto modificando la `Desktop Entry` Android Studio

```
Exec=sh -c "export ps=myPassword123 ; /path/to/studio.sh"
```

Puoi trovare maggiori dettagli in questo argomento .

Eseguendo il controllo delle versioni tramite il file "version.properties"

Puoi utilizzare Gradle per incrementare automaticamente la versione del pacchetto ogni volta che lo compili. Per fare in modo di creare un `version.properties` file nella stessa directory del `build.gradle` con il seguente contenuto:

```
VERSION_MAJOR=0
VERSION_MINOR=1
VERSION_BUILD=1
```

(Cambiando i valori per maggiore e minore come meglio credi). Quindi nel tuo `build.gradle` aggiungi il seguente codice alla sezione `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)
  }
}
```

È possibile accedere alle informazioni in Java come una stringa `BuildConfig.VERSION_NAME` per il numero {major}. {Minor}. { `BuildConfig.VERSION_NAME` } completo e come un intero `BuildConfig.VERSION_CODE` per il solo numero di build.

Modifica del nome apk di output e aggiunta del nome della versione:

Questo è il codice per cambiare il nome del file dell'applicazione di output (.apk). Il nome può essere configurato assegnando un valore diverso 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 + "]");
        }
    }
}
```

Disattiva la compressione dell'immagine per una dimensione file APK più piccola

Se si stanno ottimizzando manualmente tutte le immagini, disabilitare Cruncher APT per una dimensione file APK più piccola.

```
android {

    aaptOptions {
        cruncherEnabled = false
    }
}
```

Abilita Proguard usando Gradle

Per abilitare le configurazioni Proguard per la tua applicazione devi abilitarlo nel tuo file gradle a livello di modulo. È necessario impostare il valore di `minifyEnabled` su `true` .

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

Il codice sopra riportato applicherà le configurazioni Proguard contenute nell'SDK Android predefinito combinato con il file "proguard-rules.pro" sul tuo modulo per l'apk rilasciato.

Abilita il supporto del plug-in NDK sperimentale per Gradle e AndroidStudio

Abilita e configura il plugin Gradle sperimentale per migliorare il supporto NDK di AndroidStudio. Verifica di soddisfare i seguenti requisiti:

- Gradle 2.10 (per questo esempio)

- Android NDK r10 o successivo
- Android SDK con strumenti di compilazione v19.0.0 o versioni successive

Configura il file MyApp / build.gradle

Modifica la linea dependencies.classpath in build.gradle da es

```
classpath 'com.android.tools.build:gradle:2.1.2'
```

a

```
classpath 'com.android.tools.build:gradle-experimental:0.7.2'
```

(V0.7.2 era l'ultima versione al momento della stesura. Controlla la versione più recente e adatta la tua linea di conseguenza)

Il file build.gradle dovrebbe essere simile a questo:

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle-experimental:0.7.2'
    }
}

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Configura il file MyApp / app / build.gradle

Modifica il file build.gradle in modo che assomigli al seguente esempio. I numeri di versione potrebbero essere diversi.

```
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'])
}
```

Sincronizzare e controllare che non ci siano errori nei file Gradle prima di procedere.

Verifica se il plugin è abilitato

Innanzitutto assicurati di aver scaricato il modulo NDK di Android. Quindi crea una nuova app in AndroidStudio e aggiungi quanto segue al file ActivityMain:

```
public class MainActivity implements Activity {
    onCreate() {
        // Pregenerated code. Not important here
    }
    static {
        System.loadLibrary("myLib");
    }
    public static native String getString();
}
```

La parte `getString()` dovrebbe essere evidenziata in rosso per indicare che non è stato possibile trovare la funzione JNI corrispondente. Passa il mouse sopra la chiamata di funzione fino a quando non viene visualizzata una lampadina rossa. Fare clic sulla lampadina e selezionare `create function JNI_...` Questo dovrebbe generare un file myLib.c nella directory myApp / app / src / main / jni con la chiamata della funzione JNI corretta. Dovrebbe assomigliare a questo:

```
#include <jni.h>

JNIEXPORT jstring JNICALL
Java_com_example_mydomain_myapp_MainActivity_getString(JNIEnv *env, jobject instance)
{
    // TODO

    return (*env)->NewStringUTF(env, returnValue);
}
```

Se non sembra così, il plugin non è stato configurato correttamente o l'NDK non è stato scaricato

Mostra tutte le attività del progetto 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
```

Elimina automaticamente l'apk "non allineato"

Se non hai bisogno di file apk generati automaticamente con suffisso `unaligned` (cosa che probabilmente non fai), puoi aggiungere il seguente codice al file `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()
        }
    }
  }
}
```

Da qui

Ignorando la variante di costruzione

Per alcuni motivi potresti voler ignorare le varianti di costruzione. Ad esempio: hai "schernito" l'aroma del prodotto e lo utilizzi solo a scopo di debug, come i test di unità / strumentazione.

Ignoriamo la variante di **mockRelease** dal nostro progetto. Aprire il file **build.gradle** e scrivere:

```
    // 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);
        }
    }
```

Vedere l'albero delle dipendenze

Usa le dipendenze delle attività. A seconda di come sono configurati i tuoi moduli, possono essere le `./gradlew dependencies` o vedere le dipendenze dell'app modulo utilizzare `./gradlew :app:dependencies`

L'esempio seguente il file 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'
}
```

produrrà il seguente grafico:

```
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 (*)

. . .
```

Qui puoi vedere che il progetto include direttamente `com.android.support:design` versione 23.2.1, che a sua volta sta portando `com.android.support:support-v4` con la versione 23.2.1. Tuttavia, `com.google.android.gms:play-services` stessa hanno una dipendenza dallo stesso `support-v4` ma con una versione precedente 21.0.0, che è un conflitto rilevato da gradle.

`(*)` vengono utilizzati quando gradle salta la sottostruttura perché quelle dipendenze erano già elencate in precedenza.

Usa gradle.properties per versionnumber centrale / buildconfigurations

È possibile definire le informazioni di configurazione centrale in

- un gradle separato include il file Centralizzare le dipendenze tramite il file "dependencies.gradle"
- un file di proprietà autonomo Versionare i tuoi build tramite il file "version.properties"

oppure farlo con il file root `gradle.properties`

la struttura del progetto

```
root
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
  +- gradle.properties
```

impostazione globale per tutti i sottomoduli in 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
```

utilizzo in un sottomodulo

```
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: se si desidera pubblicare la propria app nell'app store F-Droid, è necessario utilizzare numeri magici nel file gradle perché altrimenti il robot f-droid non può leggere il versionnumner corrente per rilevare / verificare le modifiche della versione.

Visualizza le informazioni di firma

In alcune circostanze (ad esempio, ottenere una chiave API di Google) è necessario trovare l'impronta digitale del keystore. Gradle ha un compito utile che visualizza tutte le informazioni sulla firma, incluse le impronte digitali del keystore:

```
./gradlew signingReport
```

Questo è un esempio di output:

```
: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
----------
```

Definizione dei tipi di build

È possibile creare e configurare i tipi di build nel file `build.gradle` livello di `build.gradle` all'interno del blocco di `android {}` .

```
    android {
        ...
        defaultConfig {...}

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

            debug {
                applicationIdSuffix ".debug"
            }
        }
    }
```

Leggi Gradle per Android online: https://riptutorial.com/it/android/topic/95/gradle-per-android

Capitolo 117: GreenDAO

introduzione

GreenDAO è una libreria di mappatura degli oggetti relazionali per aiutare gli sviluppatori a utilizzare i database SQLite per l'archiviazione locale persistente.

Examples

Metodi di supporto per le query SELECT, INSERT, DELETE, UPDATE

Questo esempio mostra una classe helper che contiene metodi utili, quando si eseguono le query per i dati. Ogni metodo qui utilizza Java Generic per essere molto flessibile.

```
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();
}
```

Creazione di un'entità con GreenDAO 3.X con una chiave primaria composta

Quando si crea un modello per una tabella con una chiave primaria composta, è necessario un lavoro aggiuntivo sull'oggetto per consentire al modello Entity di rispettare tali vincoli.

Il seguente esempio di tabella SQL ed entità mostra la struttura per memorizzare una recensione lasciata da un cliente per un articolo in un negozio online. In questo esempio, vogliamo che le colonne `customer_id` e `item_id` siano una chiave primaria composta, consentendo l'esistenza di una sola recensione tra un cliente e un articolo specifici.

Tabella 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)
);
```

Solitamente avremmo utilizzare i `@Id` e `@Unique` annotazioni sopra i rispettivi campi nella classe entità, tuttavia per una chiave primaria composta facciamo la seguente:

1. Aggiungi l'annotazione `@Index` all'interno dell'annotazione `@Index` a livello di `@Entity` . La proprietà value contiene un elenco delimitato da virgole dei campi che compongono la chiave. Utilizzare la proprietà `unique` come mostrato per far rispettare l'univocità sulla chiave.

2. GreenDAO richiede che ogni Entità abbia un oggetto `long` o `Long` come chiave primaria. Abbiamo ancora bisogno di aggiungere questo alla classe Entity, tuttavia non abbiamo bisogno di usarlo o preoccuparci di ciò che riguarda la nostra implementazione. Nell'esempio seguente è chiamato `localID`

Entità

```
@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() {}
}
```

Iniziare con GreenDao v3.X

Dopo aver aggiunto la dipendenza della libreria GreenDao e il plugin Gradle, dobbiamo prima creare un oggetto entità.

Entità

Un'entità è un oggetto Plain Old Java *(POJO)* che modella alcuni dati nel database. GreenDao utilizzerà questa classe per creare una tabella nel database SQLite e generare automaticamente classi di supporto che è possibile utilizzare per accedere e archiviare i dati senza dover scrivere istruzioni 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...

}
```

Configurazione OneWay GreenDao

Ogni volta che viene lanciata un'applicazione, è necessario inizializzare la GreenDao. GreenDao suggerisce di mantenere questo codice in una classe Application o in qualche luogo verrà eseguito una sola volta.

```
DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this, "mydatabase", null);
db = helper.getWritableDatabase();
DaoMaster daoMaster = new DaoMaster(db);
DaoSession daoSession = daoMaster.newSession();
```

Classi Helper di GreenDao

Dopo aver creato l'oggetto entità, GreenDao crea automaticamente le classi helper utilizzate per interagire con il database. Questi sono chiamati in modo simile al nome dell'oggetto entità che è stato creato, seguito da `Dao` e recuperati dall'oggetto `daoSession` .

```
UsersDao usersDao = daoSession.getUsersDao();
```

Molte azioni tipiche del database possono ora essere eseguite utilizzando questo oggetto Dao con l'oggetto entità.

domanda

```
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();
```

Inserire

```
Users newUser = new User("John","Doe","jdoe@example.com");
usersDao.insert(newUser);
```

Aggiornare

```
// Modify a previously retrieved user object and update
user.setLastname("Dole");
usersDao.update(user);
```

Elimina

```
// Delete a previously retrieved user object
usersDao.delete(user);
```

Leggi GreenDAO online: https://riptutorial.com/it/android/topic/1345/greendao

Capitolo 118: GSON

introduzione

Gson è una libreria Java che può essere utilizzata per convertire oggetti Java nella loro rappresentazione JSON. Gson considera entrambi questi obiettivi di design molto importanti.

Caratteristiche di Gson:

Fornire `toJson()` semplici `toJson()` e `fromJson()` per convertire oggetti Java in JSON e viceversa

Consenti agli oggetti non modificabili preesistenti di essere convertiti da e verso JSON

Ampio supporto di Java Generics

Supporta oggetti arbitrariamente complessi (con gerarchie di ereditarietà profonde e uso estensivo di tipi generici)

Sintassi

- Excluder excluder ()
- FieldNamingStrategy fieldNamingStrategy ()
- <T> T fromJson (JsonElement json, Class <T> classOfT)
- <T> T fromJson (JsonElement json, Type typeOfT)
- <T> T fromJson (lettore JsonReader, tipo typeOfT)
- <T> T fromJson (Reader json, Class <T> classOfT)
- <T> T fromJson (Reader json, Type typeOfT)
- <T> T fromJson (String json, Class <T> classOfT)
- <T> T fromJson (String json, Type typeOfT)
- <T> TypeAdapter <T> getAdapter (tipo di classe <T>)
- <T> TypeAdapter <T> getAdapter (TypeToken <T> type)
- <T> TypeAdapter <T> getDelegateAdapter (TypeAdapterFactory skipPast, TypeToken <T> type)
- JsonReader newJsonReader (Lettore di lettori)
- JsonWriter newJsonWriter (writer writer)
- JsonElement toJsonTree (Object src)
- JsonElement toJsonTree (Object src, Type typeOfSrc)
- booleano 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, JsonWriter writer)

- void toJson (JsonElement jsonElement, writer Appendable)
- void toJson (JsonElement jsonElement, writer JsonWriter)
- void toJson (Object src, Appendable writer)

Examples

Parsing JSON con Gson

L'esempio mostra l'analisi di un oggetto JSON utilizzando la libreria Gson di Google .

Parsing objects:

```
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 = "";
    }


}
```

Quindi, se è necessario eseguire l'analisi, utilizzare quanto segue:

```
String robotJson = "{
                        \"version\": \"JellyBean\",
                        \"age\": 3,
                        \"robotName\": \"Droid\"
                    }";

Gson gson = new Gson();
Robot robot = gson.fromJson(robotJson, Robot.class);
```

Analizzare una lista:

Quando si recupera un elenco di oggetti JSON, spesso si vorrà analizzarli e convertirli in oggetti Java.

La stringa JSON che tenteremo di convertire è la seguente:

```
{
  "owned_dogs": [
    {
      "name": "Ron",
      "age": 12,
      "breed": "terrier"
    },
    {
      "name": "Bob",
      "age": 4,
      "breed": "bulldog"
    },
    {
      "name": "Johny",
      "age": 3,
      "breed": "golden retriever"
    }
  ]
}
```

Questo particolare array JSON contiene tre oggetti. Nel nostro codice Java vorremmo mappare questi oggetti su oggetti `Dog` . Un oggetto Dog dovrebbe assomigliare a questo:

```
private class Dog {
    public String name;
    public int age;

    @SerializedName("breed")
    public String breedName;
}
```

Per convertire l'array JSON in un `Dog[]` :

```
Dog[] arrayOfDogs = gson.fromJson(jsonArrayString, Dog[].class);
```

Conversione di un `Dog[]` in una stringa JSON:

```
String jsonArray = gson.toJson(arrayOfDogs, Dog[].class);
```

Per convertire l'array JSON in un `ArrayList<Dog>` possiamo fare quanto segue:

```
Type typeListOfDogs = new TypeToken<List<Dog>>(){}.getType();
List<Dog> listOfDogs = gson.fromJson(jsonArrayString, typeListOfDogs);
```

Il tipo Type `typeListOfDogs` definisce come dovrebbe apparire un elenco di oggetti `Dog` . GSON può utilizzare questo oggetto di tipo per mappare l'array JSON ai valori corretti.

In alternativa, la conversione di un `List<Dog>` in un array JSON può essere eseguita in modo simile.

```
String jsonArray = gson.toJson(listOfDogs, typeListOfDogs);
```

Analizzare la proprietà JSON per enumare con Gson

Se vuoi analizzare una stringa per enumare con Gson:

> {"status": "open"}

```
public enum Status {
    @SerializedName("open")
    OPEN,
    @SerializedName("waiting")
    WAITING,
    @SerializedName("confirm")
    CONFIRM,
    @SerializedName("ready")
    READY
}
```

Analizzare una lista con Gson

Metodo 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());
```

Ciò è utile per la maggior parte delle classi contenitore generiche, poiché non è possibile ottenere la classe di un tipo parametrizzato (ad esempio: non è possibile chiamare `List<String>.class`).

Metodo 2

```
public class StringList extends ArrayList<String> { }

...

List<String> members = gson.fromJson(json, StringList.class);
```

In alternativa, puoi sempre sottoclassi il tipo che desideri e poi passare in quella classe. Tuttavia, questa non è sempre una buona pratica, poiché ti restituirà un oggetto di tipo `StringList` ;

Serializzazione / deserializzazione JSON con AutoValue e Gson

Importa nel tuo file radice gradle

```
classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
```

Importa nel file dell'app 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'
```

Crea oggetto con autovalore:

```
@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 il tuo convertitore Gson con GsonBuilder

```
Gson gson = new GsonBuilder()
                .registerTypeAdapterFactory(
                    new AutoValueGsonTypeAdapterFactory())
                .create());
```

Deserialize

```
String myJsonData = "{
    \"signin_token\": \"mySigninToken\",
    \"username\": \"myUsername\" }";
SignIn signInData = gson.fromJson(myJsonData, Signin.class);
```

serializzare

```
Signin myData = SignIn.create("myTokenData", "myUsername");
String myJsonData = gson.toJson(myData);
```

Usare Gson è un ottimo modo per semplificare il codice di serializzazione e deserializzazione usando oggetti POJO. L'effetto collaterale è che la riflessione è costosa in termini di prestazioni. Ecco perché l'utilizzo di AutoValue-Gson per generare CustomTypeAdapter eviterà questo costo di riflessione rimanendo molto semplice da aggiornare quando si verifica un cambio API.

Parsing JSON to Generic Class Object con Gson

Supponiamo di avere una stringa JSON:

```
["first","second","third"]
```

Possiamo analizzare questa stringa JSON in un array `String` :

```
Gson gson = new Gson();
String jsonArray = "[\"first\",\"second\",\"third\"]";
String[] strings = gson.fromJson(jsonArray, String[].class);
```

Ma se vogliamo analizzarlo in un oggetto `List<String>` , dobbiamo usare `TypeToken` .

Ecco il campione:

```
Gson gson = new Gson();
String jsonArray = "[\"first\",\"second\",\"third\"]";
List<String> stringList = gson.fromJson(jsonArray, new TypeToken<List<String>>()
{}.getType());
```

Supponiamo di avere due classi qui sotto:

```
public class Outer<T> {
    public int index;
    public T data;
}

public class Person {
    public String firstName;
    public String lastName;
}
```

e abbiamo una stringa JSON che dovrebbe essere analizzata su un oggetto `Outer<Person>` .

Questo esempio mostra come analizzare questa stringa JSON sull'oggetto classe generico correlato:

```
String json = "......";
Type userType = new TypeToken<Outer<Person>>(){}.getType();
Result<User> userResult = gson.fromJson(json,userType);
```

Se la stringa JSON deve essere analizzata su un oggetto `Outer<List<Person>>` :

```
Type userListType = new TypeToken<Outer<List<Person>>>(){}.getType();
Result<List<User>> userListResult = gson.fromJson(json,userListType);
```

Aggiunta di Gson al tuo progetto

```
dependencies {
    compile 'com.google.code.gson:gson:2.8.1'
}
```

Per utilizzare l'ultima versione di Gson

La riga sottostante compilerà l'ultima versione della libreria gson ogni volta che si compila, non è necessario modificare la versione.

Pro: puoi utilizzare le ultime funzionalità, la velocità e meno bug.

Contro: Potrebbe rompere la compatibilità con il tuo codice.

```
compile 'com.google.code.gson:gson:+'
```

Usare Gson per caricare un file JSON dal disco.

Questo caricherà un file JSON dal disco e lo convertirà nel tipo specificato.

```
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);
}
```

Aggiunta di un convertitore personalizzato a Gson

A volte è necessario serializzare o deserializzare alcuni campi in un formato desiderato, ad esempio il proprio back-end può utilizzare il formato "AAAA-MM-gg HH: mm" per le date e si desidera che il POJOS utilizzi la classe DateTime in Joda Time.

Per convertire automaticamente queste stringhe nell'oggetto DateTimes, è possibile utilizzare un convertitore personalizzato.

```
/**
 * 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());
    }
}
```

Per fare in modo che Gson usi il convertitore appena creato, devi assegnarlo durante la creazione dell'oggetto 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
```

Per deserializzare la data in quel formato devi solo definire un campo nel formato DateTime:

```
public class SomePojo {
    private DateTime someDate;
}
```

Quando Gson incontra un campo di tipo DateTime, chiamerà il tuo convertitore per deserializzare il campo.

Utilizzo di Gson come serializzatore con Retrofit

Prima di tutto è necessario aggiungere `GsonConverterFactory` al file build.gradle

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

Quindi, è necessario aggiungere la fabbrica del convertitore durante la creazione del servizio di retrofit:

```
Gson gson = new GsonBuilder().create();
new Retrofit.Builder()
        .baseUrl(someUrl)
        .addConverterFactory(GsonConverterFactory.create(gson))
        .build()
        .create(RetrofitService.class);
```

È possibile aggiungere convertitori personalizzati durante la creazione dell'oggetto Gson che si sta passando alla fabbrica. Ti consente di creare conversioni di tipo personalizzato.

Analizzando l'array json in una classe generica usando Gson

Supponiamo di avere 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"
    }
  ]
}
```

Possiamo analizzare manualmente questo array in un oggetto Tweets personalizzato (contenitore dell'elenco dei tweet), ma è più facile farlo con il metodo `fromJson` :

```
Gson gson = new Gson();
String jsonArray = "....";
Tweets tweets = gson.fromJson(jsonArray, Tweets.class);
```

Supponiamo di avere due classi qui sotto:

```
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;
}
```

e se hai solo bisogno di analizzare un array json puoi usare questo codice nel tuo parsing:

```
String tweetsJsonArray = "[{.....},{.....}]"
List<Tweet> tweets = gson.fromJson(tweetsJsonArray, new TypeToken<List<Tweet>>()
{}.getType());
```

Deserializzatore JSON personalizzato usando Gson

Immagina di avere tutte le date in tutte le risposte in qualche formato personalizzato, ad esempio `/Date(1465935152)/` e vuoi applicare la regola generale per deserializzare tutte le date di Json alle istanze di `Date` java. In questo caso è necessario implementare JS Deserializer personalizzato.

Esempio di JSON:

```
{
  "id": 1,
  "created_on": "Date(1465935152)",
  "updated_on": "Date(1465968945)",
  "name": "Oleksandr"
}
```

Supponiamo di avere questa classe qui sotto:

```
class User {
    @SerializedName("id")
    long id;
    @SerializedName("created_on")
    Date createdOn;
    @SerializedName("updated_on")
    Date updatedOn;
    @SerializedName("name")
    String name;
}
```

Deserializzatore personalizzato:

```
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());
    }
}
```

E l'uso:

```
Gson gson = new GsonBuilder()
                .registerTypeAdapter(Date.class, new DateDeSerializer())
                .create();
String json = "....";
User user = gson.fromJson(json, User.class);
```

Serializza e deserializza le stringhe di Jackson JSON con i tipi di data

Questo vale anche nel caso in cui si desideri rendere compatibile la conversione di Gson Date con Jackson, ad esempio.

Jackson solitamente serializza Data in "millisecondi dall'epoca", mentre Gson utilizza un formato leggibile come `Aug 31, 2016 10:26:17` per rappresentare Data. Ciò porta a JsonSyntaxExceptions in Gson quando si tenta di deserializzare una data in formato Jackson.

Per aggirare questo problema, puoi aggiungere un serializzatore personalizzato e un deserializzatore personalizzato:

```
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();
```

Usare Gson con ereditarietà

Gson non supporta l'ereditarietà fuori dalla scatola.

Diciamo che abbiamo la seguente gerarchia di classi:

```
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;
```

```
    }
}
```

E ora vogliamo serializzare un'istanza di `DerivedClass1` in una stringa JSON

```
DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;

Gson gson = new Gson();
String derivedClass1Json = gson.toJson(derivedClass1);
```

Ora, in un altro posto, riceviamo questa stringa json e vogliamo deserializzarla - ma in fase di compilazione sappiamo solo che si suppone che sia un'istanza di `BaseClass` :

```
BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());
```

Ma GSON non sa che `derivedClass1Json` era in origine un'istanza di `DerivedClass1` , quindi verrà stampato 10.

Come risolvere questo?

È necessario creare il proprio `JsonDeserializer` , che gestisce tali casi. La soluzione non è perfettamente pulita, ma non ho potuto trovare una soluzione migliore.

Innanzitutto, aggiungi il seguente campo alla tua classe base

```
@SerializedName("type")
private String typeName;
```

E inizializzalo nel costruttore della classe base

```
public BaseClass() {
    typeName = getClass().getName();
}
```

Ora aggiungi la seguente classe:

```
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);
 }
}
```

Tutto quello che resta da fare è agganciare tutto -

```
GsonBuilder builder = new GsonBuilder();
 builder
 .registerTypeAdapter(BaseClass.class, new JsonDeserializerWithInheritance<BaseClass>());
 Gson gson = builder.create();
```

E ora, eseguendo il seguente codice-

```
 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());
```

Stamperà 5.

Leggi GSON online: https://riptutorial.com/it/android/topic/4158/gson

Capitolo 119: handler

Osservazioni

Un gestore può essere facilmente utilizzato per eseguire il codice dopo un periodo di tempo ritardato. È anche utile per eseguire il codice ripetutamente dopo un determinato periodo di tempo chiamando nuovamente il metodo Handler.postDelayed () dal metodo run () di Runnable.

Examples

Utilizzo di un gestore per eseguire il codice dopo un periodo di tempo ritardato

Esecuzione del codice dopo 1,5 secondi:

```
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
```

Eseguendo codice ripetutamente ogni 1 secondo:

```
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 e comunicazione tra thread

Poiché i `Handler` vengono utilizzati per inviare `Message` e `Runnable` s alla coda dei messaggi di Thread, è facile implementare la comunicazione basata su eventi tra più thread. Ogni Thread che ha un `Looper` è in grado di ricevere ed elaborare messaggi. Un `HandlerThread` è un Thread che implementa tale `Looper` , ad esempio il Thread principale (Thread UI) implementa le funzionalità di un `HandlerThread` .

Creazione di un gestore per il thread corrente

```
Handler handler = new Handler();
```

Creazione di un gestore per il thread principale (thread UI)

```
Handler handler = new Handler(Looper.getMainLooper());
```

Invia un Runnable da un altro thread alla discussione principale

```
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();
```

Creazione di un gestore per un altro handlerThread e invio di eventi ad esso

```
// 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
    }
});
```

Arresta il gestore dall'esecuzione

Per fermare il gestore dall'esecuzione, rimuovere il callback ad esso collegato utilizzando il runnable in esecuzione al suo interno:

```
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);
}
```

Utilizzare il gestore per creare un timer (simile a javax.swing.Timer)

Questo può essere utile se stai scrivendo un gioco o qualcosa che deve eseguire un pezzo di codice ogni pochi secondi.

```
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 ();
    }
}
```

Esempio di utilizzo:

```
Timer timer = new Timer(new Runnable() {
    public void run() {
        System.out.println("Hello");
    }
}, 1000, true)
```

Questo codice stamperà "Hello" ogni secondo.

Leggi handler online: https://riptutorial.com/it/android/topic/1425/handler

Capitolo 120: HttpURLConnection

Sintassi

- abstract void disconnect ()
- abstract booleano usingProxy ()
- static boolean getFollowRedirects ()
- static void setFollowRedirects (set booleano)
- String getHeaderField (int n)
- String getHeaderFieldKey (int n)
- String getRequestMethod ()
- String getResponseMessage ()
- int getResponseCode ()
- long getHeaderFieldDate (String name, long Default)
- boolean getInstanceFollowRedirects ()
- Autorizzazione getPermission ()
- InputStream getErrorStream ()
- void setChunkedStreamingMode (int chunklen)
- void setFixedLengthStreamingMode (int contentLength)
- void setFixedLengthStreamingMode (long contentLength)
- void setInstanceFollowRedirects (boolean followRedirects)
- void setRequestMethod (metodo String)

Osservazioni

`HttpURLConnection` è il client HTTP standard per Android, utilizzato per inviare e ricevere dati sul web. È un'implementazione concreta di URLConnection per HTTP (RFC 2616).

Examples

Creazione di un HttpURLConnection

Per creare un nuovo client HTTP Android `HttpURLConnection` , chiama `openConnection()` su un'istanza URL. Poiché `openConnection()` restituisce un `URLConnection` , è necessario eseguire il cast esplicito del valore restituito.

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
// do something with the connection
```

Se stai creando un nuovo `URL` , devi anche gestire le eccezioni associate all'analisi degli 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 volta che il corpo della risposta è stato letto e la connessione non è più necessaria, la connessione deve essere chiusa chiamando `disconnect()` .

Ecco un esempio:

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
try {
    // do something with the connection
} finally {
    connection.disconnect();
}
```

Invio di una richiesta GET HTTP

```
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();
}
```

Si prega di notare che le eccezioni non sono gestite nell'esempio sopra. Un esempio completo, inclusa la gestione di un'eccezione (banale), potrebbe essere:

```
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();
    }
}
```

Lettura del corpo di una richiesta GET HTTP

```
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();
}
```

Si prega di notare che le eccezioni non sono gestite nell'esempio sopra.

Utilizzare HttpURLConnection per multipart / form-data

Crea una classe personalizzata per chiamare la richiesta 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;
        }
    }
```

Usalo (modo asincrono)

```
    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;
                }
```

Invio di una richiesta POST HTTP con parametri

Utilizzare una HashMap per memorizzare i parametri che devono essere inviati al server tramite i parametri POST:

```
HashMap<String, String> params;
```

Una volta popolato il `params` HashMap, crea lo StringBuilder che verrà utilizzato per inviarlo al server:

```
    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++;
    }
```

Quindi, crea HttpURLConnection, apri la connessione e invia i parametri 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();
}
```

Quindi ricevi il risultato che il server restituisce:

```
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();
    }
}
```

Carica (POST) file usando HttpURLConnection

Molto spesso è necessario inviare / caricare un file su un server remoto, ad esempio un'immagine, video, audio o un backup del database dell'applicazione su un server remoto privato. Supponendo che il server si aspetti una richiesta POST con il contenuto, ecco un semplice esempio di come completare questa attività in Android.

I caricamenti dei file vengono inviati utilizzando le richieste POST `multipart/form-data` . È molto facile da implementare:

```
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
}
```

Naturalmente, le eccezioni dovranno essere catturate o dichiarate come gettate. Un paio di punti da notare su questo codice:

1. `postTarget` è l'URL di destinazione del POST; `oauthToken` è il token di autenticazione; `fileDescription` è la descrizione del file, che viene inviato come valore della `description` del campo; `file` è il file da inviare - è di tipo `java.io.File` - se hai il percorso del file, puoi usare invece il `new File(filePath)` .
2. Imposta l'intestazione `Authorization` per un'autenticazione oAuth
3. Usa Apache Common `FileUtil` per leggere il file in un array di byte - se hai già il contenuto del file in un array di byte o in qualche altro modo in memoria, allora non c'è bisogno di leggerlo.

Una classe HttpURLConnection multiuso per gestire tutti i tipi di richieste HTTP

La seguente classe può essere utilizzata come singola classe in grado di gestire `GET` , `POST` , `PUT` , `PATCH` e altre richieste:

```
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

Utilizzare uno dei costruttori dati della classe a seconda se è necessario inviare dati `POST` o eventuali intestazioni extra.

Il metodo `onComplete()` verrà chiamato quando il recupero dei dati è completo. I dati vengono restituiti come oggetto della classe `APIResponseObject` , che ha un codice di stato che indica il codice di stato HTTP della richiesta e una stringa contenente la risposta. È possibile analizzare questa risposta nella classe, ad esempio XML o JSON.

Chiama `execute()` sull'oggetto della classe per eseguire la richiesta, come mostrato nell'esempio seguente:

```
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();
}
```

Leggi HttpURLConnection online: https://riptutorial.com/it/android/topic/781/httpurlconnection

Capitolo 121: Il file manifest

introduzione

The Manifest è un file obbligatorio denominato esattamente "AndroidManifest.xml" e si trova nella directory principale dell'app. Specifica il nome dell'app, l'icona, il nome del pacchetto Java, la versione, la dichiarazione di attività, i servizi, le autorizzazioni delle app e altre informazioni.

Examples

Dichiarazione dei componenti

L'attività principale di manifest è informare il sistema in merito ai componenti dell'app. Ad esempio, un file manifest può dichiarare un'attività come segue:

```
<?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>
```

Nell'elemento `<application>` , l'attributo `android:icon` punta alle risorse per un'icona che identifica l'app.

Nell'elemento, l'attributo `android:name` specifica il nome classe completo della sottoclasse Activity e l'attributo android: label specifica una stringa da utilizzare come etichetta visibile dall'utente per l'attività.

Devi dichiarare tutti i componenti dell'app in questo modo:

- `<activity>` elementi per le attività

- `<service>` elementi per i servizi

- `<receiver>` elementi per i ricevitori di trasmissione

- Elementi `<provider>` per i fornitori di contenuti

Attività, servizi e fornitori di contenuti che includi nel tuo codice sorgente ma che non dichiarano nel manifest non sono visibili al sistema e, di conseguenza, non possono mai essere eseguiti. Tuttavia, i ricevitori di broadcast possono essere dichiarati nel manifest o creati dinamicamente nel codice (come oggetti `BroadcastReceiver`) e registrati con il sistema chiamando `registerReceiver()` .

Per ulteriori informazioni su come strutturare il file manifest per la tua app, consulta la

documentazione del file AndroidManifest.xml.

Dichiarare le autorizzazioni nel file manifest

Qualsiasi autorizzazione richiesta dalla tua applicazione per accedere a una parte protetta dell'API o per interagire con altre applicazioni deve essere dichiarata nel tuo file `AndroidManifest.xml` . Questo viene fatto usando il `<uses-permission />` .

Sintassi

```
<uses-permission android:name="string"
   android:maxSdkVersion="integer"/>
```

android: nome: questo è il nome dell'autorizzazione richiesta

android: maxSdkVersion: il più alto livello API a cui questa autorizzazione deve essere concessa per la tua app. L'impostazione di questa autorizzazione è facoltativa e dovrebbe essere impostata solo se l'autorizzazione richiesta dalla tua app non è più necessaria a un determinato livello API.

Esempio 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>
```

* Vedi anche l'argomento Autorizzazioni .

Leggi Il file manifest online: https://riptutorial.com/it/android/topic/1848/il-file-manifest

Capitolo 122: ImageView

introduzione

ImageView (`android.widget.ImageView`) è una vista per la visualizzazione e la manipolazione di risorse immagine, come Drawables e Bitmap.

Alcuni effetti, discussi in questo argomento, possono essere applicati all'immagine. L'origine dell'immagine può essere impostata nel file XML (cartella `layout`) o programmaticamente nel codice Java.

Sintassi

- Il metodo `setImageResource(int resId)` imposta un drawable come contenuto di `ImageView` .
- **Utilizzo:** `imageView.setImageResource(R.drawable.anyImage)`

Parametri

Parametro	Descrizione
`resId`	il nome del file immagine nella cartella `res` (solitamente nella cartella `drawable`)

Examples

Imposta la risorsa immagine

```
<ImageView
 android:id="@+id/imgExample"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 ...
 />
```

imposta un drawable come contenuto di `ImageView` usando l'attributo XML:

```
android:src="@drawable/android2"
```

imposta un drawable a livello di codice:

```
 ImageView imgExample = (ImageView) findViewById(R.id.imgExample);
 imgExample.setImageResource(R.drawable.android2);
```

Imposta alfa

"alpha" è usato per specificare l'opacità di un'immagine.

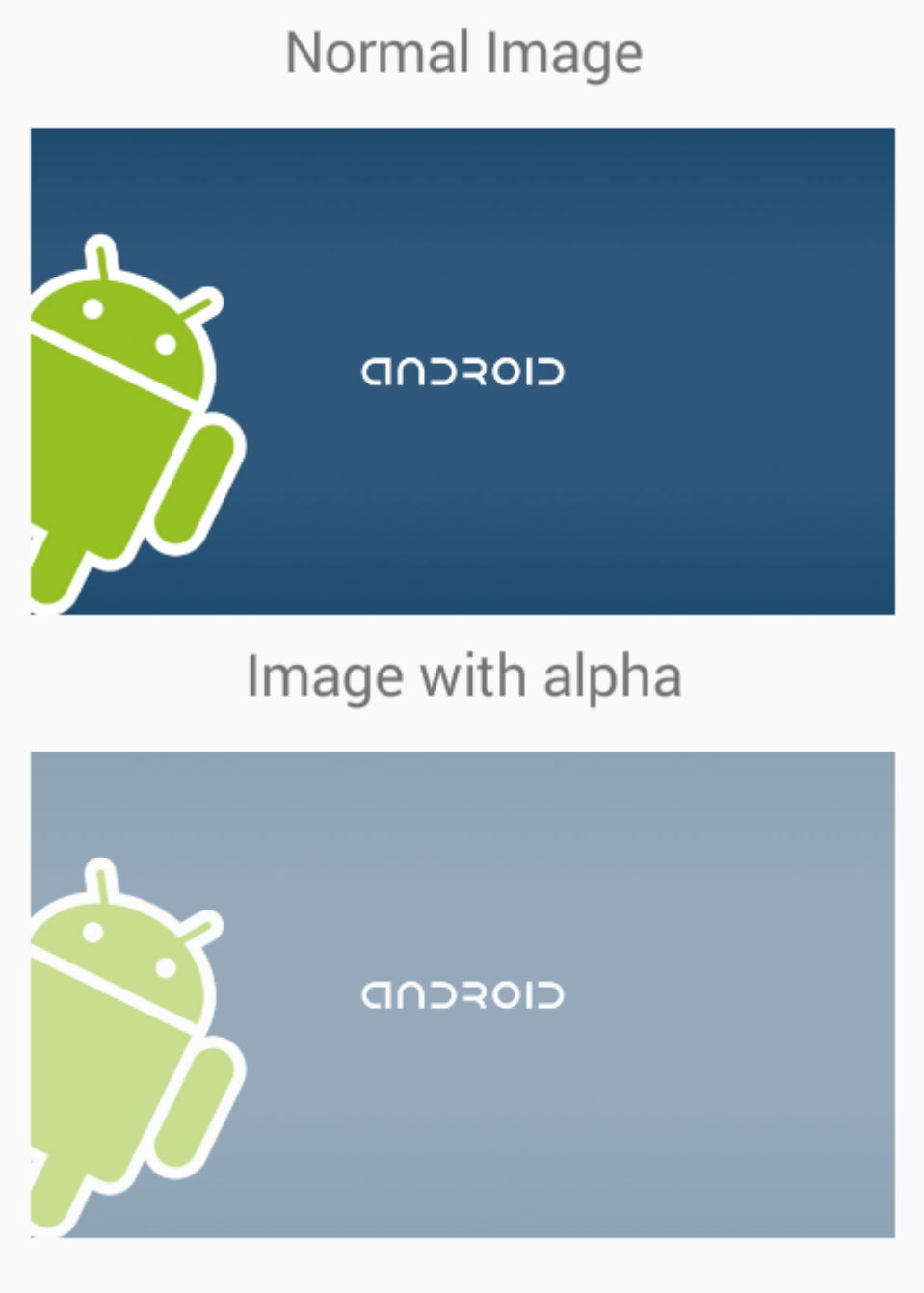
imposta alpha usando l'attributo XML:

```
android:alpha="0.5"
```

Nota: prende il valore float da 0 (trasparente) a 1 (completamente visibile)

imposta alfa a livello di codice:

```
imgExample.setAlpha(0.5f);
```



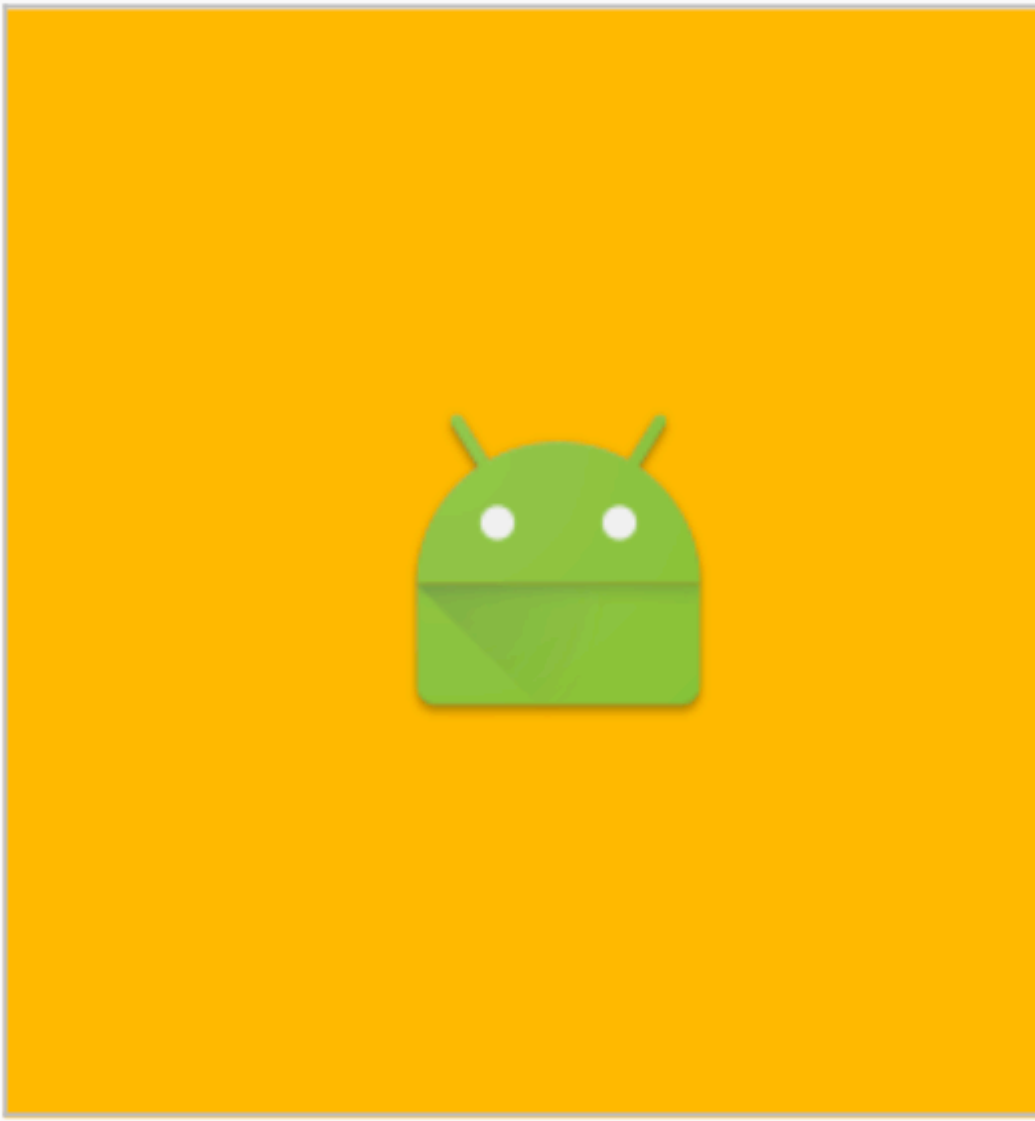
ImageView ScaleType - Center

L'immagine contenuta in ImageView potrebbe non corrispondere alla dimensione esatta fornita al contenitore. In tal caso, il framework consente di ridimensionare l'immagine in vari modi.

Centro

```
    <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"/>
```

Questo non ridimensionerà l'immagine e la centrerà all'interno del contenitore *(Arancione = contenitore)*



`ImageView` .

Attributo XML:

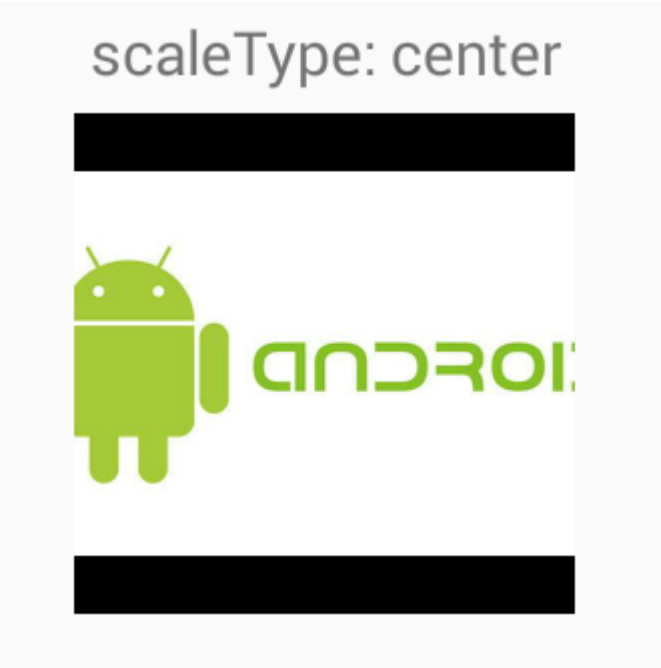
```
android:scaleType="..."
```

Illustrerò diversi tipi di scala con un `ImageView` quadrato che ha uno sfondo nero e vogliamo visualizzare un disegnatore rettangolare con sfondo bianco in `ImageView` .

```
<ImageView
 android:id="@+id/imgExample"
 android:layout_width="200dp"
 android:layout_height="200dp"
 android:background="#000"
 android:src="@drawable/android2"
 android:scaleType="..."/>
```

scaleType deve essere uno dei seguenti valori:

1. `center` : `center` l'immagine nella vista, ma non esegue ridimensionamenti.



2. `centerCrop` : `centerCrop` l'immagine in modo uniforme (mantiene le proporzioni dell'immagine) in modo che entrambe le dimensioni (larghezza e altezza) dell'immagine siano uguali o maggiori della dimensione corrispondente della vista (meno riempimento). L'immagine è quindi centrata nella vista.

3. `centerInside` : `centerInside` l'immagine in modo uniforme (mantiene le proporzioni dell'immagine) in modo che entrambe le dimensioni (larghezza e altezza) dell'immagine siano uguali o inferiori alla dimensione corrispondente della vista (meno riempimento). L'immagine è quindi centrata nella vista.

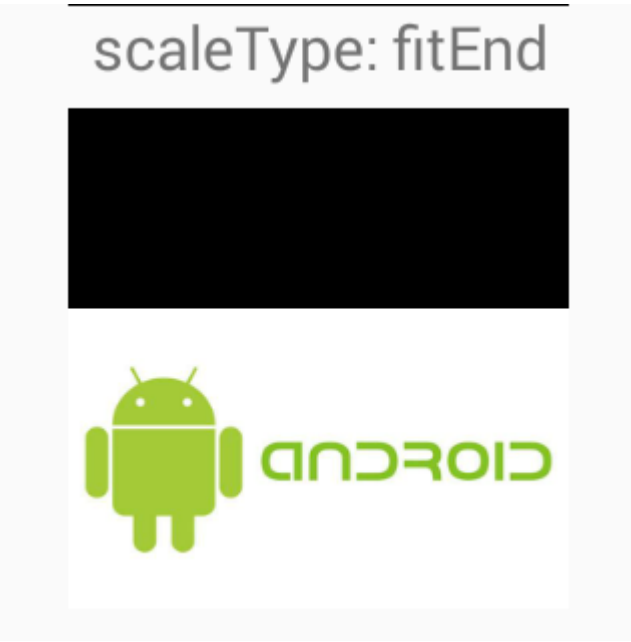
4. `matrix` : scala usando la matrice immagine durante il disegno.

5. `fitXY` : `fitXY` l'immagine usando FILL .

6. `fitStart` : `fitStart` l'immagine usando START .

7. `fitCenter` : `fitCenter` l'immagine usando CENTER .

8. `fitEnd` : `fitEnd` l'immagine usando END .



Imposta la tinta

Imposta un colore per l'immagine. Per impostazione predefinita, la tinta si fonderà usando la modalità `SRC_ATOP` .

imposta la tinta usando l'attributo XML:

```
android:tint="#009c38"
```

Nota: deve essere un valore di colore, sotto forma di `"#rgb"` , `"#argb"` , `"#rrggbb"` o `"#aarrggbb"` .

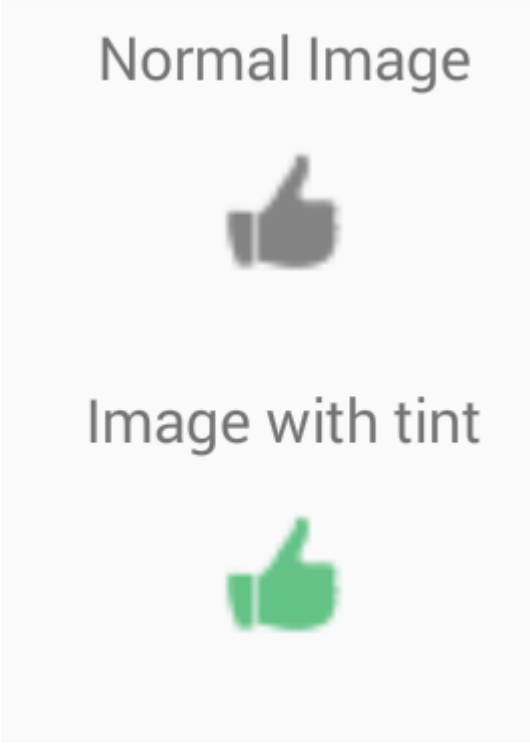
imposta la tinta programmaticamente:

```
imgExample.setColorFilter(Color.argb(255, 0, 156, 38));
```

e puoi cancellare questo filtro colore:

```
imgExample.clearColorFilter();
```

Esempio:



MLRoundedImageView.java

Copia e incolla seguendo la lezione nel tuo pacchetto:

```
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;
    }
}
```

Utilizzare questa classe in XML con il nome del pacchetto invece di `ImageView`

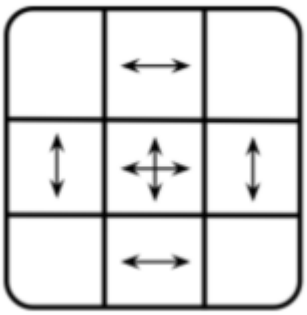
```
<com.androidbuts.example.MLRoundedImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />
```

Leggi ImageView online: https://riptutorial.com/it/android/topic/4709/imageview

Capitolo 123: Immagini 9-Patch

Osservazioni

Un file di immagine a *9 patch* è un file appositamente formattato in modo che Android sappia quali aree / parti dell'immagine possono o non possono essere ridimensionate. Rompe la tua immagine in una griglia 3x3. Gli angoli rimangono non graduati, i lati vengono ridimensionati in una direzione e il centro viene ridimensionato in entrambe le dimensioni.



Un'immagine di Nove Patch (9-Patch) è una bitmap con un bordo di un singolo pixel attorno all'intera immagine. Ignorando i 4 pixel negli angoli dell'immagine. Questo bordo fornisce metadati per la bitmap stessa. I limiti sono contrassegnati da linee nere continue.

Un'immagine di Nove Patch è memorizzata con l'estensione `.9.png` .

Il bordo superiore indica le aree che si estendono orizzontalmente. Il bordo sinistro indica le aree che si estendono verticalmente.

Il bordo inferiore indica il riempimento orizzontale. Il bordo destro indica il riempimento verticale.

I bordi di imbottitura vengono solitamente utilizzati per determinare dove deve essere disegnato il testo.

C'è un eccellente strumento fornito da Google che semplifica *enormemente* la creazione di questi file.

Situato nell'SDK di Android: `android-sdk\tools\lib\draw9patch.jar`

Examples

Angoli arrotondati di base

La chiave per lo stretching corretto è nel bordo superiore e sinistro.

Il bordo superiore controlla lo stiramento orizzontale e il bordo sinistro controlla lo stretching

verticale.

Questo esempio crea angoli arrotondati adatti per un toast.

Le parti dell'immagine che si trovano sotto il *bordo superiore* e a destra del *bordo sinistro* si espandono per riempire tutto lo spazio non utilizzato.

Questo esempio si estenderà a tutte le combinazioni di dimensioni, come mostrato di seguito:

Spinner di base

Lo `Spinner` può essere resettato secondo i tuoi requisiti di stile usando una Nove Patch.

Ad esempio, vedi questa Nove Patch:

Come puoi vedere, ha 3 aree estremamente piccole di stretching segnate.

Il bordo superiore è rimasto solo dell'icona contrassegnata. Ciò indica che voglio il lato sinistro (trasparenza completa) del drawable per riempire la vista `Spinner` fino a quando non viene raggiunta l'icona.

Il bordo sinistro ha marcati segmenti trasparenti nella parte superiore e inferiore dell'icona contrassegnata. Ciò indica che sia la parte superiore che quella inferiore si espanderanno alla dimensione della vista `Spinner` . Questo lascerà l'icona stessa centrata verticalmente.

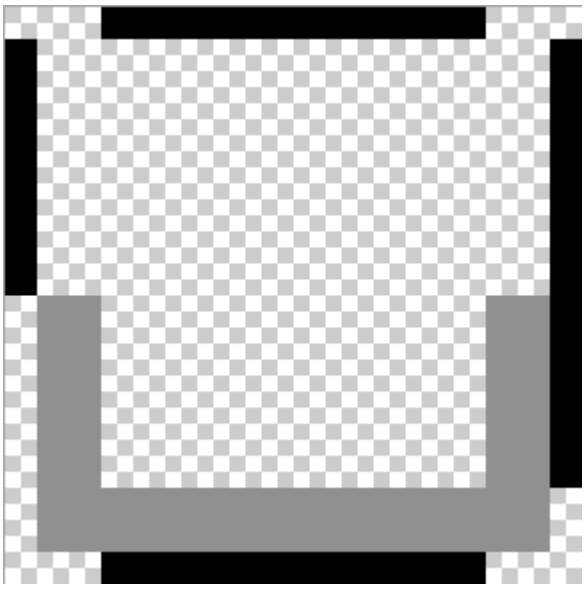
Utilizzando l'immagine senza i metadati di Nove Patch:

Usando l'immagine con i metadati di Nove Patch:

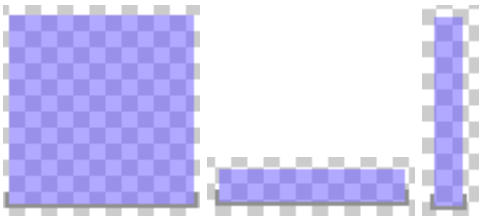
Linee di imbottitura opzionali

Le immagini a nove pezzi consentono la definizione opzionale delle linee di imbottitura nell'immagine. Le linee di imbottitura sono le linee a destra e in basso.

Se una vista imposta l'immagine a 9 patch come sfondo, le linee di riempimento sono utilizzate per definire lo spazio per il contenuto della vista (ad esempio l'immissione di testo in un testo `EditText`). Se le linee di riempimento non sono definite, vengono utilizzate le linee sinistra e superiore.



L'area del contenuto dell'immagine allungata appare così:



Leggi Immagini 9-Patch online: https://riptutorial.com/it/android/topic/461/immagini-9-patch

Capitolo 124: Impaginazione in RecyclerView

introduzione

La paginazione è un problema comune con molte app mobili che devono gestire elenchi di dati. La maggior parte delle app mobili sta iniziando a riprendere il modello della "pagina infinita", in cui lo scorrimento carica automaticamente i nuovi contenuti. CWAC Endless Adapter rende davvero facile utilizzare questo modello nelle applicazioni 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

interfaccia pubblica 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;
```

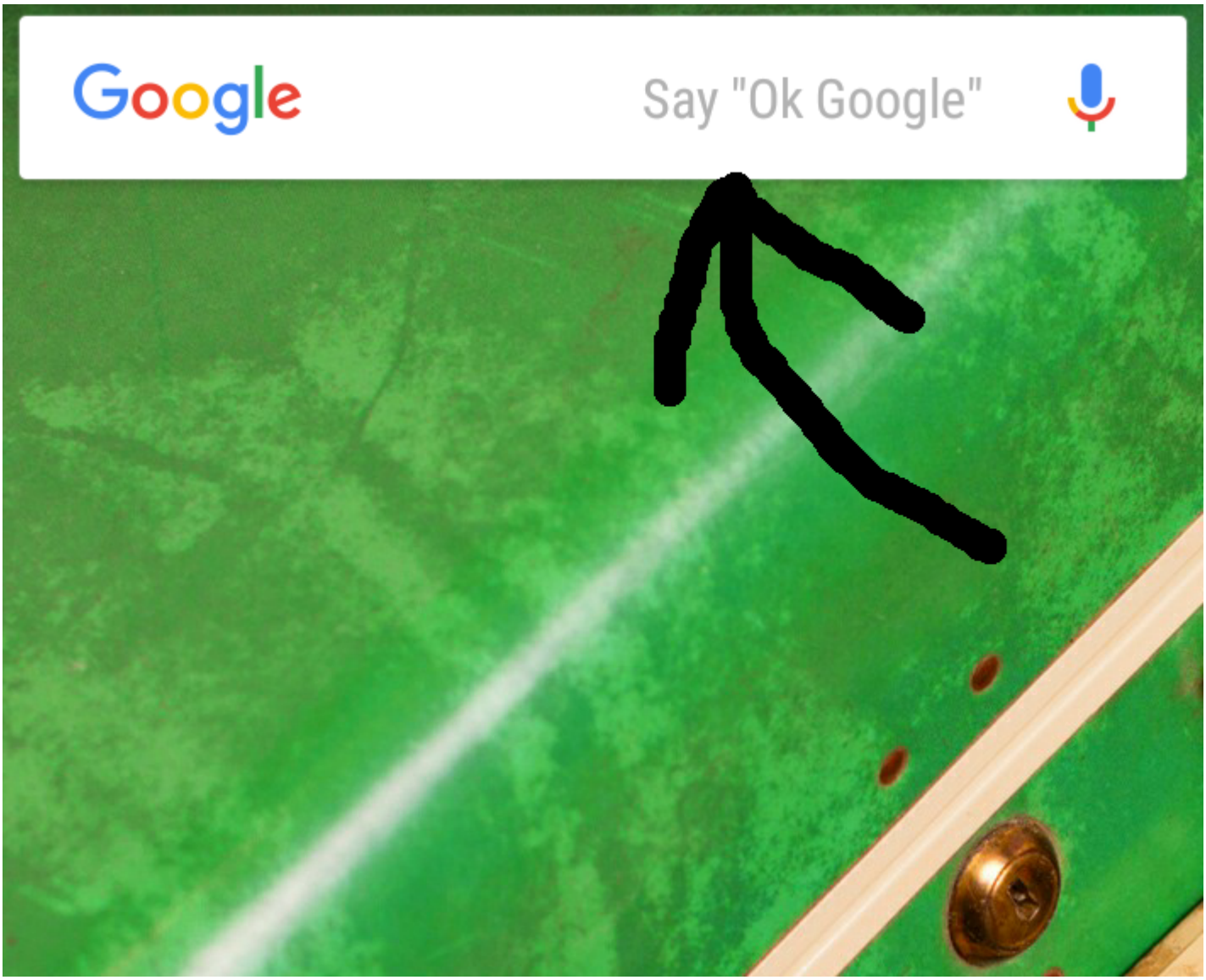
```
        public StudentViewHolder(View v) {
            super(v);
            tvName = (TextView) v.findViewById(R.id.tvName);
            tvEmailId = (TextView) v.findViewById(R.id.tvEmailId);


        }
    }

    public static class ProgressViewHolder extends RecyclerView.ViewHolder {
        public ProgressBar progressBar;

        public ProgressViewHolder(View v) {
            super(v);
            progressBar = (ProgressBar) v.findViewById(R.id.progressBar1);
        }
    }

}
```

Leggi Impaginazione in RecyclerView online:
https://riptutorial.com/it/android/topic/9243/impaginazione-in-recyclerview

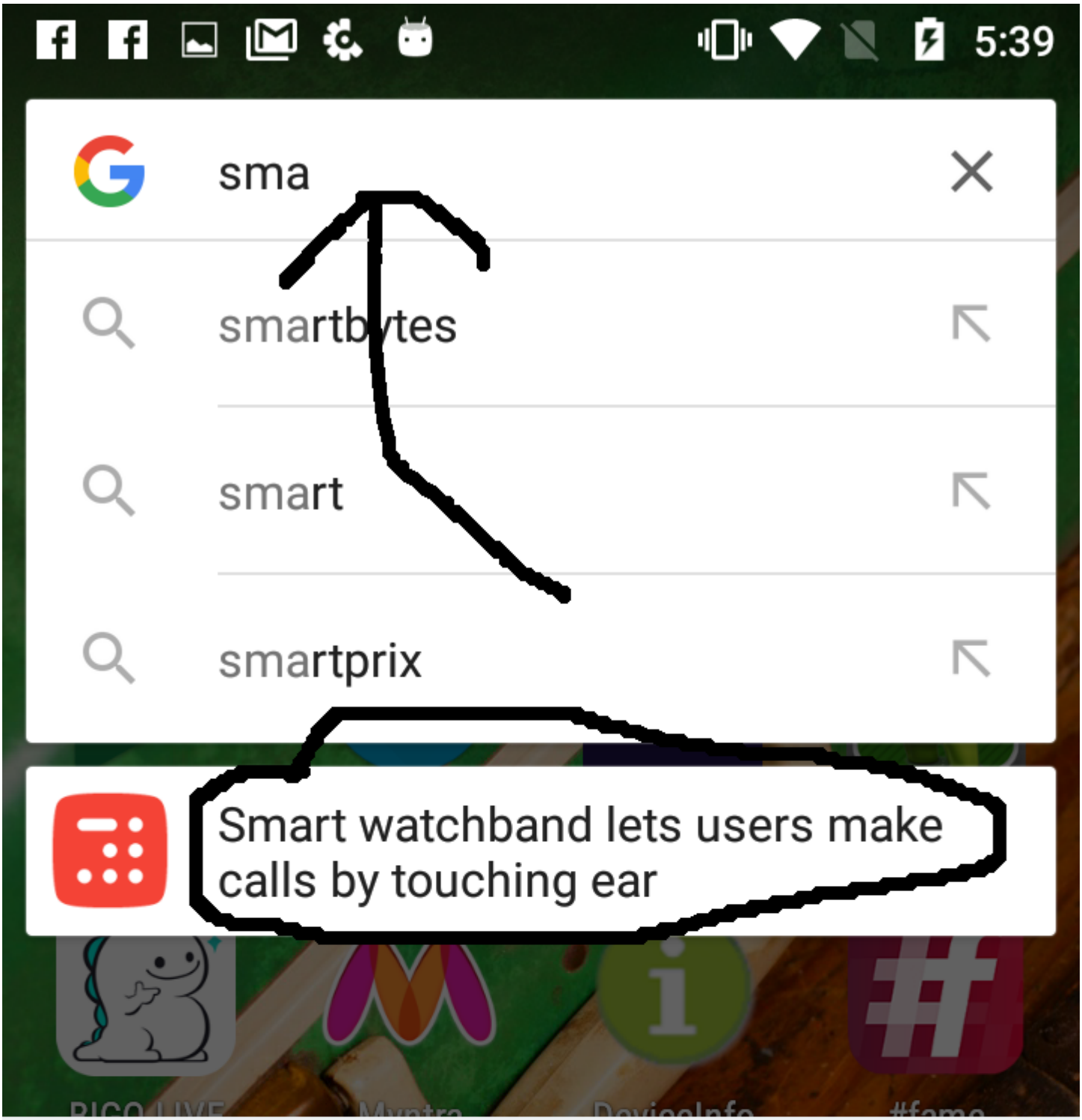
Capitolo 125: Indicizzazione dell'app Firebase

Osservazioni

- Quando decidi di implementare l'indicizzazione delle app, potresti trovare molti blog e documentazione che potrebbero confondervi, in questo caso ti suggerisco di attenersi ai documenti ufficiali forniti da Firebase-Google. Anche se si desidera utilizzare terze parti per farlo, per prima cosa provare a seguire questa documentazione perché questo vi darà un'idea chiara di come funzionano le cose.
- Google impiegherà circa 24 ore per indicizzare i tuoi contenuti. Quindi sii paziente. Puoi fare dei test per far stare bene ogni cosa dalla tua parte.
- Il primo esempio ti consente di supportare l'URL HTTP del tuo sito web per reindirizzare nella tua app. Ciò funzionerà come, hai cercato una query nella ricerca di google, i risultati mostrano uno dei tuoi URL del sito web, i cui collegamenti app sono presenti nella tua app che è già installata. Facendo clic su questo URL, ti reindirizzerà direttamente nella schermata dell'app corrispondente a quel risultato di ricerca. È così che ho scoperto per questo.
- L'aggiunta di AppIndexing API indicizza il contenuto e viene utilizzato nei completamenti automatici nella barra di ricerca di Google. Prendiamo un esempio di applicazione InShorts per ogni pagina, c'è un titolo e una piccola descrizione. Dopo aver letto 2 o 3 titoli, chiudi l'applicazione e spostati su google searchBar.



Prova a inserire il titolo che hai appena visto, riceverai il suggerimento di una pagina di app con quel titolo come titolo. Questo è diverso dai suggerimenti delle app che ottieni durante la ricerca di app. Ciò accade perché hai scritto il codice API AppIndexing per questa particolare pagina e il titolo è lo stesso che hai inizializzato in `onCreate()` .


5:39
sma
smartbytes
smart
smartprix
Smart watchband lets users make calls by touching ear

Specifica l'azione intent ACTION_VIEW in modo che il filtro intent possa essere raggiunto da Ricerca Google.

<data> Aggiungi uno o più tag, in cui ogni tag rappresenta un formato URI che si risolve nell'attività. Al minimo, il tag deve includere l'attributo android: scheme. È possibile aggiungere ulteriori attributi per perfezionare ulteriormente il tipo di URI accettato dall'attività. Ad esempio, potresti avere più attività che accettano URI simili, ma che differiscono semplicemente in base al nome del percorso. In questo caso, utilizzare l'attributo android: path o le sue varianti (pathPattern o pathPrefix) per differenziare l'attività che il sistema dovrebbe aprire per i diversi percorsi URI.

<categoria> Include la categoria BROWSABLE. La categoria BROWSABLE è richiesta affinché il filtro intent sia accessibile da un browser web. Senza di esso, fare clic su un collegamento in un browser non può risolvere la tua app. La categoria DEFAULT è facoltativa, ma consigliata. Senza questa categoria, l'attività può essere avviata solo con un intento esplicito, utilizzando il nome del componente dell'app.

Passaggio 4: - Gestire gli URL in arrivo

```
    @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());
    }
```

}

Passo 5: - Puoi testare questo utilizzando il comando Android Debug Bridge o le configurazioni dello studio. Comando Adb: - Avvia l'applicazione e quindi esegui questo comando: -

```
adb shell am start -a android.intent.action.VIEW -d "{URL}" < package name >
```

Configurazioni di Android Studio: - **Android studio> Costruisci> Modifica configurazione> Avvia opzioni> seleziona URL> quindi digita il tuo URL qui> Applica** e verifica.Esegui la tua applicazione se la finestra "Esegui" mostra errore, quindi devi controllare il formato dell'URL con il tuo i link di app menzionati in manifest altrimenti verranno eseguiti correttamente e reindirizzati alla pagina menzionata nell'URL, se specificato.

Aggiungi API AppIndexing

Per l'aggiunta di questo a progetto è possibile trovare facilmente doc ufficiale, ma in questo esempio ho intenzione di evidenziare alcune delle aree chiave di cui occuparsi.

Passaggio 1: aggiungi il servizio Google

```
dependencies {
      ...
      compile 'com.google.android.gms:play-services-appindexing:9.4.0'
      ...
    }
```

Passaggio 2: - Importa classi

```
import com.google.android.gms.appindexing.Action;
import com.google.android.gms.appindexing.AppIndex;
import com.google.android.gms.common.api.GoogleApiClient;
```

Passaggio 3: Aggiungere le chiamate all'API di indicizzazione delle app

```
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();
}
```

Per verificare ciò basta seguire il punto 4 in Osservazioni indicato di seguito.

Leggi Indicizzazione dell'app Firebase online:
https://riptutorial.com/it/android/topic/5957/indicizzazione-dell-app-firebase

Capitolo 126: Iniziare con OpenGL ES 2.0+

introduzione

Questo argomento riguarda la configurazione e l'uso di **OpenGL ES 2.0+** su Android. OpenGL ES è lo standard per la grafica accelerata 2D e 3D su sistemi embedded, tra cui console, smartphone, elettrodomestici e veicoli.

Examples

Impostazione di GLSurfaceView e OpenGL ES 2.0+

Per utilizzare OpenGL ES nella tua applicazione devi aggiungerlo al manifest:

```
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
```

Crea il tuo GLSurfaceView esteso:

```
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);
        }
    }
}
```

Aggiungi `MyGLSurfaceView` al tuo layout:

```
<com.example.app.MyGLSurfaceView
    android:id="@+id/gles_renderer"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Per utilizzare la versione più recente di OpenGL ES basta cambiare il numero di versione nel manifest, nell'importazione statica e modificare `setEGLContextClientVersion` .

Compilare e collegare gli shaders GLSL-ES dal file di asset

La cartella Risorse è il luogo più comune in cui archiviare i file shader GLSL-ES. Per utilizzarli nell'applicazione OpenGL ES è necessario caricarli in una stringa. Questa funzione crea una stringa dal file di asset:

```
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;
    }
}
```

Ora è necessario creare una funzione che compila uno shader memorizzato in una puntura:

```
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;
}
```

Ora puoi caricare, compilare e collegare i tuoi ombreggiatori:

```
// 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);
}
```

Se non ci sono errori, il tuo programma shader è pronto per l'uso:

```
glUseProgram(shaderProgram);
```

Leggi Iniziare con OpenGL ES 2.0+ online: https://riptutorial.com/it/android/topic/8662/iniziare-con-opengl-es-2-0plus

Capitolo 127: Installazione di app con ADB

Examples

Installa un'app

Scrivi il seguente comando nel tuo terminale:

```
adb install [-rtsdg] <file>
```

Nota che devi passare un file che si trova sul tuo computer e non sul tuo dispositivo.

Se si aggiunge `-r` alla fine, tutti gli apk conflittuali esistenti verranno sovrascritti. Altrimenti, il comando si chiuderà con un errore.

`-g` concederà immediatamente tutte le autorizzazioni di runtime.

`-d` consente il downgrade del codice versione (applicabile solo su pacchetti debuggabili).

Utilizzare `-s` per installare l'applicazione sulla scheda SD esterna.

`-t` permetterà di usare le applicazioni di test.

Disinstallare un'app

Scrivi il seguente comando nel tuo terminale per disinstallare un'app con un nome pacchetto fornito:

```
adb uninstall <packagename>
```

Installa tutti i file apk nella directory

Finestre :

```
for %f in (C:\your_app_path\*.apk) do adb install "%f"
```

Linux:

```
for f in *.apk ; do adb install "$f" ; done
```

Leggi Installazione di app con ADB online:
https://riptutorial.com/it/android/topic/5301/installazione-di-app-con-adb

Capitolo 128: Integra Google Accedi

Sintassi

- newInstance () - Per creare una singola istanza di Google Helper
- initGoogleSignIn () - Per inizializzare il log in di Google
- getGoogleAccountDetails () - Per accedere ai dettagli dell'account
- signOut () - Per disconnettersi
- getGoogleClient () - Per utilizzare GoogleApiClient

Parametri

Parametro	Dettaglio
ETICHETTA	Una stringa utilizzata durante la registrazione
GoogleSignInHelper	Un riferimento statico per helper
AppCompatActivity	Un riferimento di attività
GoogleApiClient	Un riferimento di GoogleAPIClient
RC_SIGN_IN	Un numero intero rappresenta il risultato dell'attività costante
isLoggingOut	Un booleano per verificare se l'attività di disconnessione è in esecuzione o meno

Examples

Google Accedi con la classe Helper

Aggiungi sotto al tuo `build.gradle` dal tag `android` :

```
// Apply plug-in to app.
apply plugin: 'com.google.gms.google-services'
```

Aggiungi sotto la classe helper al tuo pacchetto 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");
    }
}
```

Aggiungi sotto il codice al tuo `OnActivityResult` nel file di attività:

```
 // [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]
```

Leggi Integra Google Accedi online: https://riptutorial.com/it/android/topic/2837/integra-google-accedi

Capitolo 129: Integrare OpenCV in Android Studio

Osservazioni

Le librerie Open CV possono essere trovate sul web usando un motore di ricerca.

I **Gotchas** :

- Se si abbassa la piattaforma di destinazione sotto KitKat alcune delle librerie OpenCV non funzioneranno più, in particolare le classi relative a *org.opencv.android.Camera2Renderer* e altre classi correlate. Probabilmente è possibile aggirare questo problema semplicemente rimuovendo i file OpenCV .java appropriati.
- Se si aumenta la piattaforma di destinazione su Lollipop o sopra il mio esempio di caricamento di un file potrebbe non funzionare perché l'uso di percorsi di file assoluti è disapprovato. Quindi potresti dover cambiare l'esempio per caricare un file dalla galleria o da qualche altra parte. Ci sono numerosi esempi in giro.

Examples

Istruzioni

Testato con AS v1.4.1 ma dovrebbe funzionare anche con le versioni più recenti.

1. Crea un nuovo progetto Android Studio utilizzando la procedura guidata del progetto (Menu: / File / Nuovo progetto):

   - Chiamalo " **cvtest1** "
   - Fattore di forma: **API 19, Android 4.4 (KitKat)**
   - **Attività vuota** denominata **MainActivity**

   Dovresti avere una directory *cvtest1 in* cui è archiviato questo progetto. (la barra del titolo di Android Studio ti mostra dove cvtest1 è quando apri il progetto)

2. Verifica che la tua app funzioni correttamente. Prova a cambiare qualcosa come il testo "Hello World" per confermare che il ciclo di build / test è OK per te. (Sto testando con un emulatore di un dispositivo API 19).

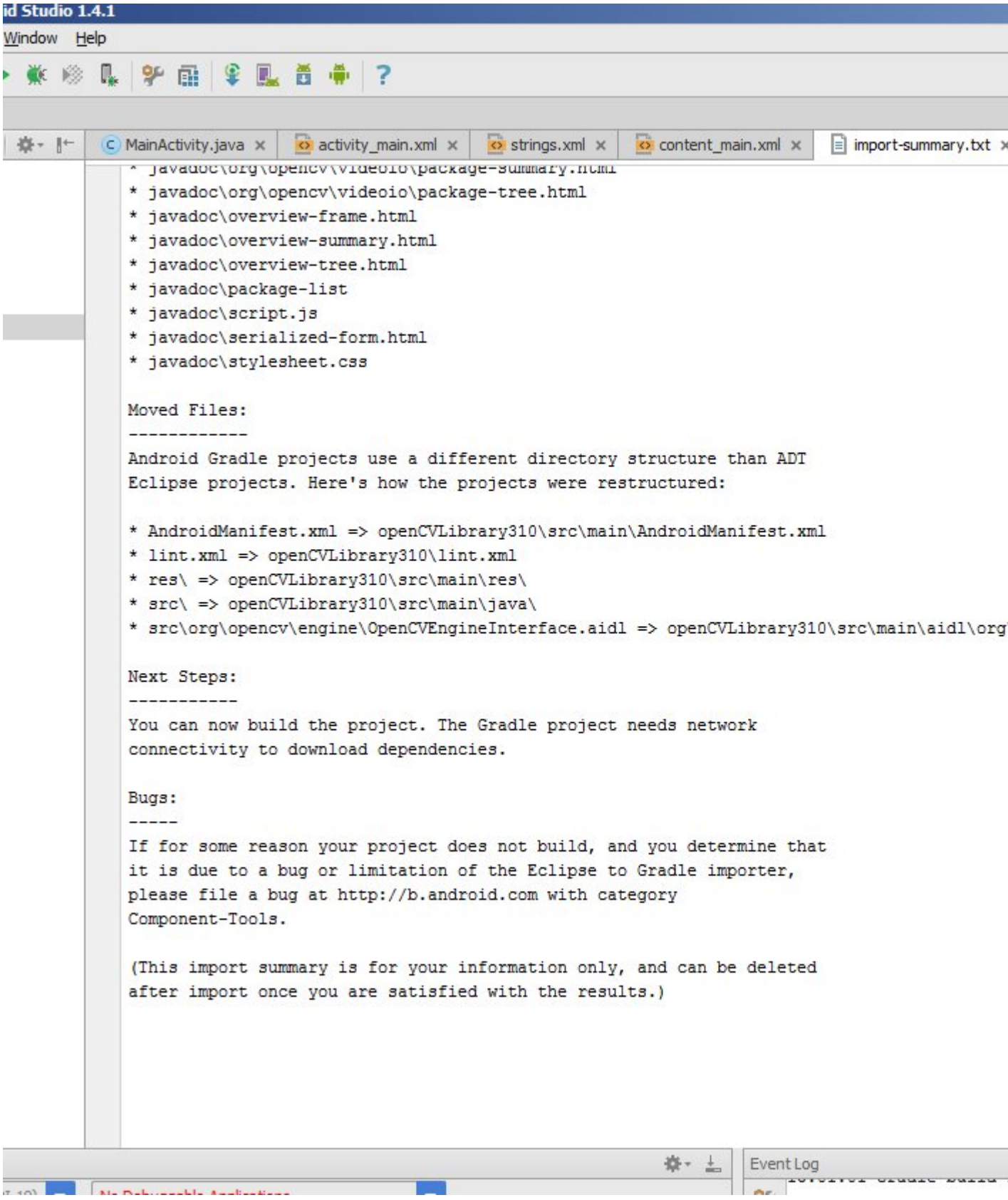
3. Scarica il pacchetto OpenCV per Android v3.1.0 e decomprimilo in qualche directory temporanea da qualche parte. (Assicurati che sia il pacchetto specifico per Android e non solo il pacchetto OpenCV per Java.) Chiamerò questa directory " **unzip-dir** " Sotto **unzip-dir** dovresti avere una directory **sdk / native / libs** con sottodirectory che iniziano con cose come **arm** ..., **mips** ... e **x86** ... (uno per ogni tipo di "architettura" su cui gira Android)

4. Da Android Studio importa OpenCV nel tuo progetto come modulo: **Menu: / File / New /**

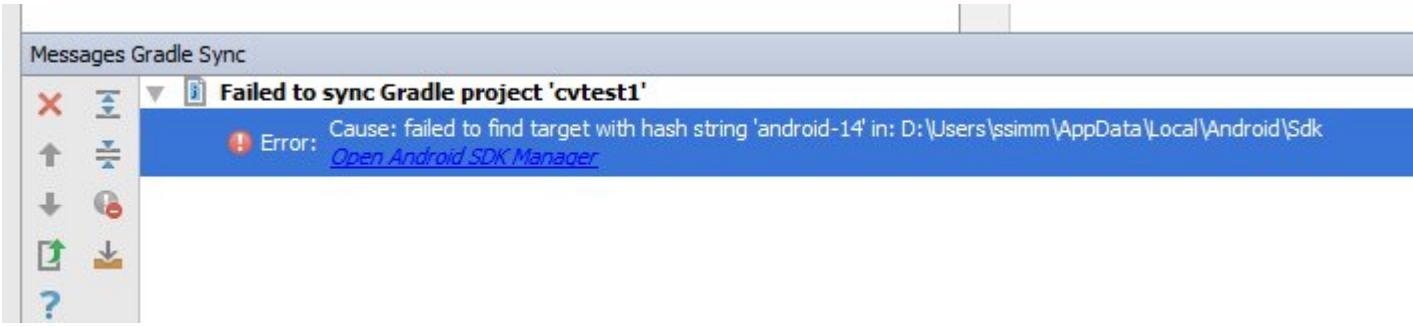
Import_Module :

- Directory di origine: **{unzip-dir} / sdk / java**
- Nome del modulo: Android Studio riempie automaticamente questo campo con **openCVLibrary310** (il nome esatto probabilmente non importa ma andremo con questo).
- Clicca su **Avanti** . Si ottiene una schermata con tre caselle di controllo e domande su barattoli, librerie e opzioni di importazione. Tutti e tre dovrebbero essere controllati. Clicca su **Finisci.**

Android Studio inizia a importare il modulo e viene visualizzato un file **import-summary.txt** con un elenco di ciò che non è stato importato (principalmente file javadoc) e altre informazioni.



Ma ricevi anche un messaggio di errore che dice di **non riuscire a trovare il target con la stringa hash 'android-14' ....** Ciò accade perché il file build.gradle nel file zip OpenCV scaricato dice di compilare utilizzando la versione 14 dell'API di Android, che per impostazione predefinita non si dispone di Android Studio v1.4.1.

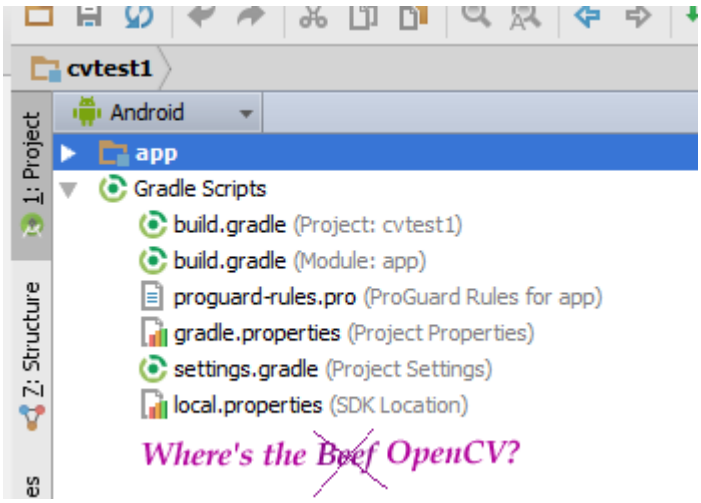


5. Aprire la finestra di dialogo struttura del progetto (**Menu: / File / Struttura_progetto**). Selezionare il modulo "app", fare clic sulla scheda **Dipendenze** e aggiungere **: openCVLibrary310** come dipendenza del modulo. Quando si seleziona **Aggiungi / Module_Dependency** dovrebbe apparire nell'elenco dei moduli che è possibile aggiungere. Verrà ora visualizzato come una dipendenza, ma nel registro eventi verranno visualizzati altri errori **non-find-android-14** .

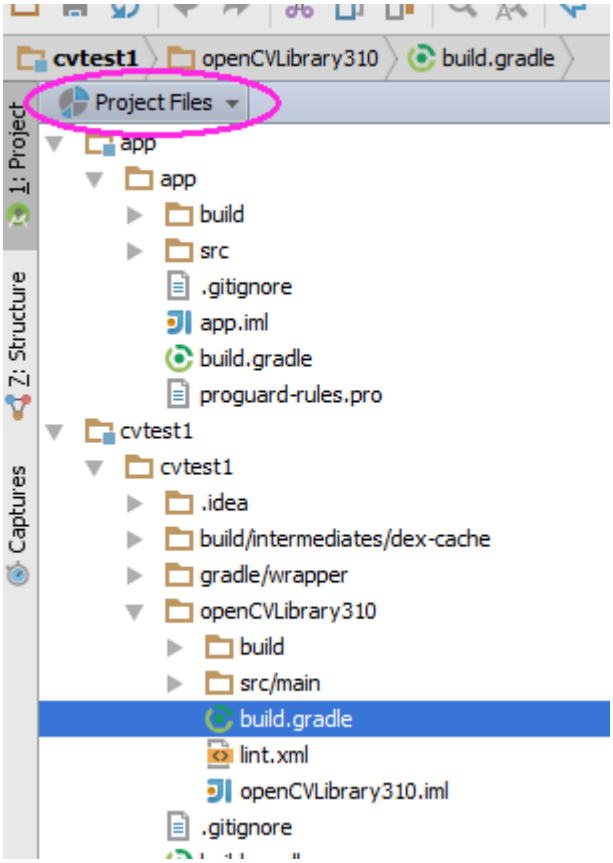
6. Cerca nel file **build.gradle** per il modulo dell'app. Esistono più file build.gradle in un progetto Android. Quello che vuoi è nella directory **cvtest1 / app** e dalla vista del progetto sembra **build.gradle (Module: app)** . Nota i valori di questi quattro campi:

    - compileSDKVersion (il mio dice 23)
    - buildToolsVersion (il mio dice 23.0.2)
    - minSdkVersion (il mio dice 19)
    - targetSdkVersion (il mio dice 23)

7. Il tuo progetto ora ha una **directory cvtest1 / OpenCVLibrary310** ma non è visibile dalla vista del progetto:

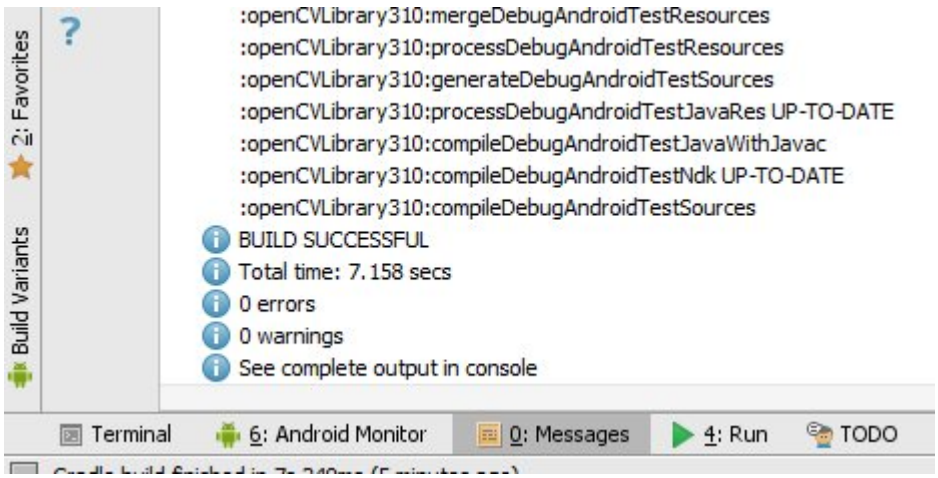


Usa qualche altro strumento, come qualsiasi gestore di file, e vai in questa directory. Puoi anche cambiare la vista del progetto da **Android** a **File di progetto** e puoi trovare questa directory come mostrato in questo screenshot:



All'interno c'è un altro file **build.gradle** (è evidenziato nello screenshot qui sopra). Aggiorna questo file con i quattro valori del passaggio 6.

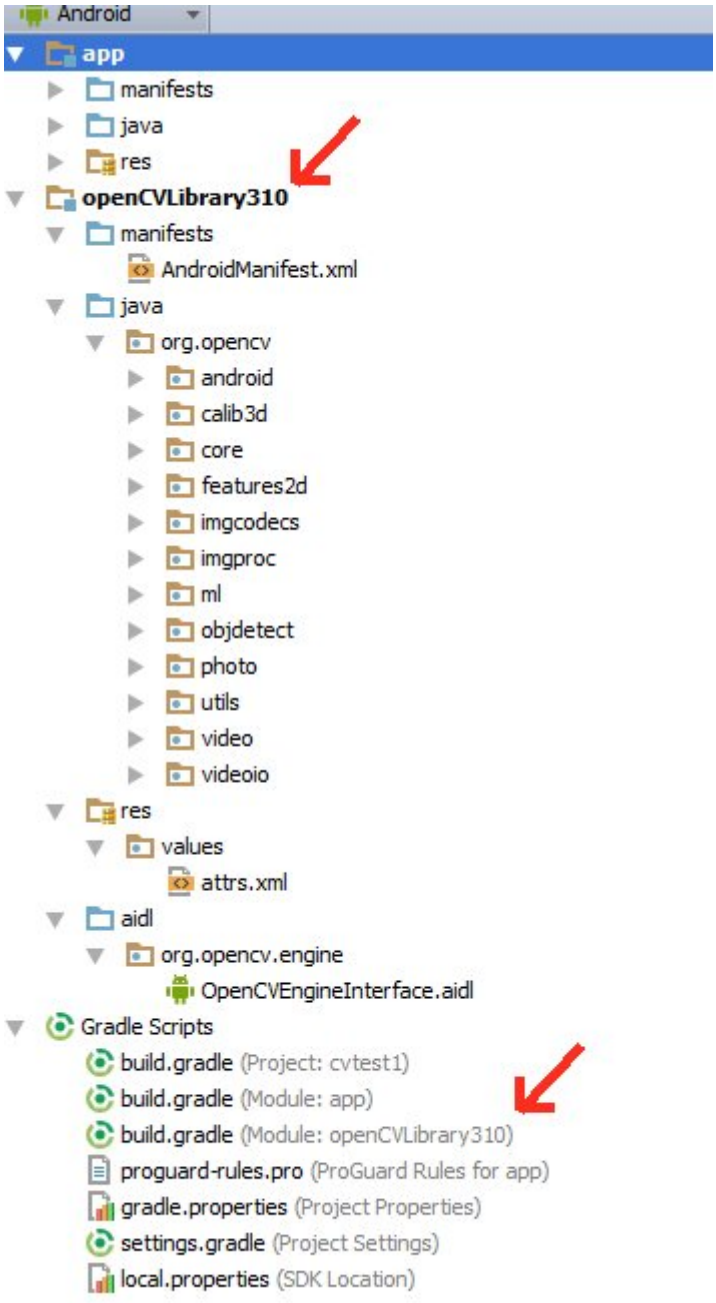
8. Risincronizza il tuo progetto e poi puliscilo / ricostruilo. (**Menu: / Build / Clean_Project**) Dovrebbe essere pulito e **compilato** senza errori e dovresti vedere molti riferimenti a **: openCVLibrary310** nella schermata **0: Messaggi** .



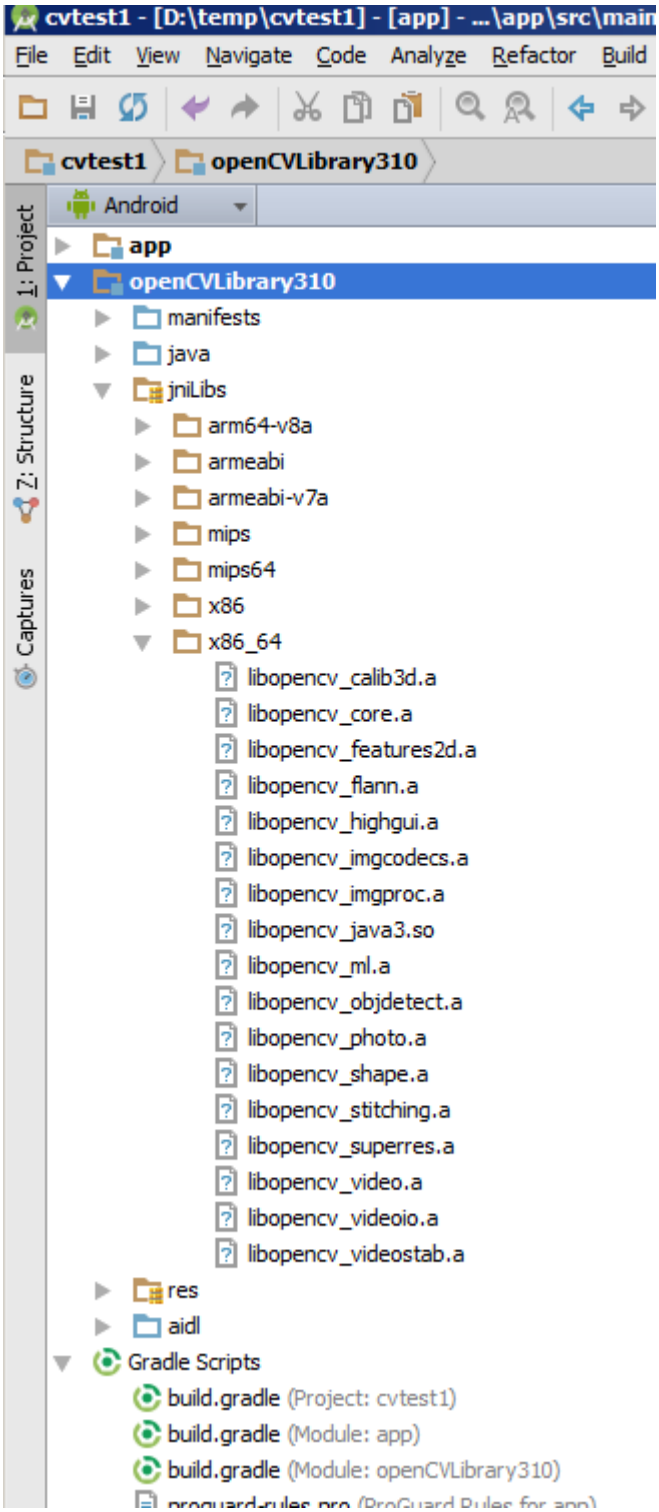
A questo punto il modulo dovrebbe apparire nella gerarchia del progetto come **openCVLibrary310** , proprio come l' **app** . (Si noti che in quel piccolo menu a discesa sono passato dalla **Vista Progetto** alla **vista Android**). Dovresti anche vedere un ulteriore file **build.gradle** in "Gradle Scripts" ma trovo l'interfaccia di Android Studio un po 'problematica e talvolta non lo fa subito. Quindi prova a risincronizzare, pulire, riavviare anche Android Studio.

Dovresti vedere il modulo openCVLibrary310 con tutte le funzioni OpenCV sotto java come

in questo screenshot:



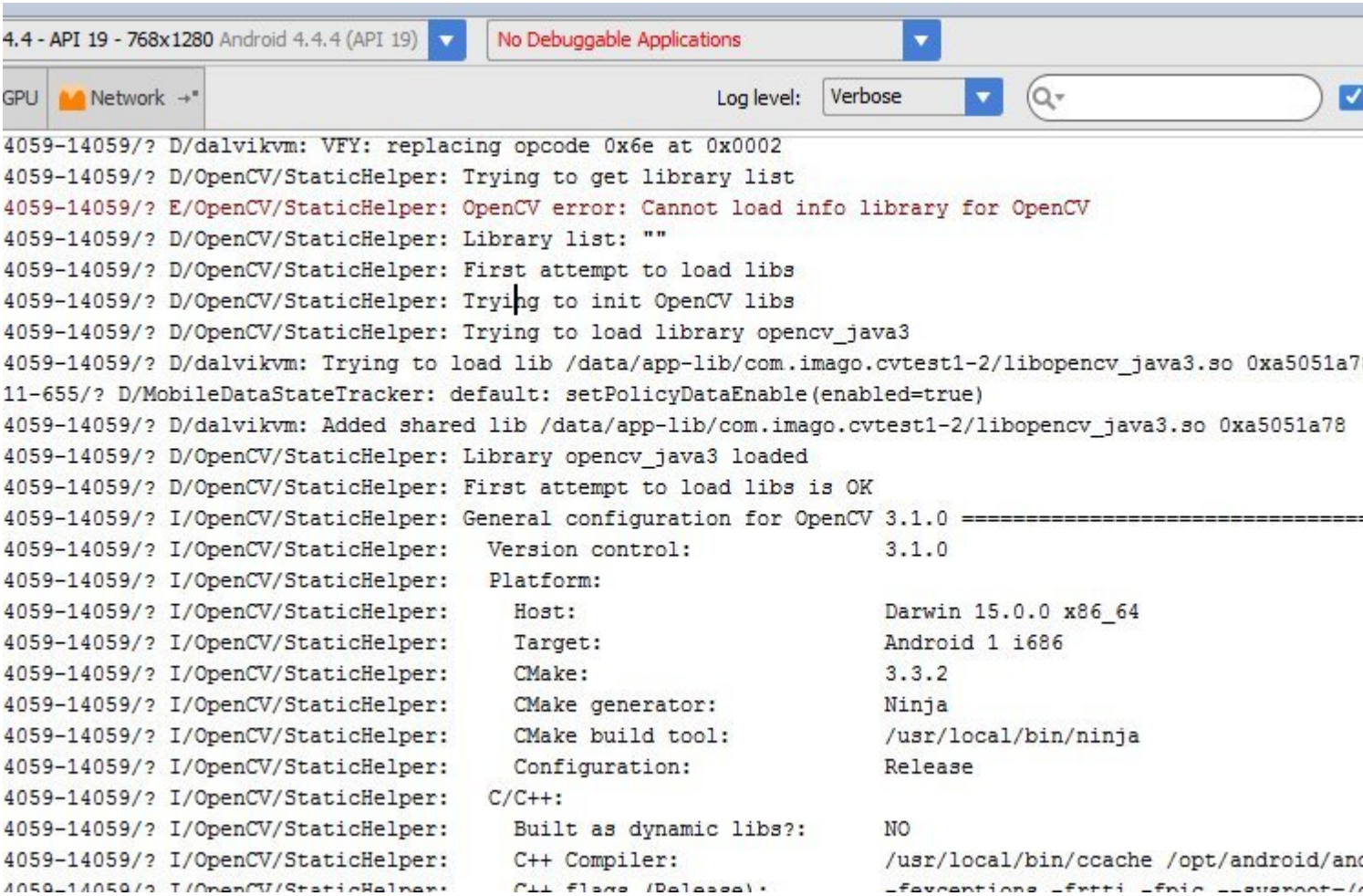
9. Copia la directory **{unzip-dir} / sdk / native / libs** (e tutto sotto di essa) sul tuo progetto Android, su **cvtest1 / OpenCVLibrary310 / src / main /** , quindi rinomina la tua copia da **libs** a **jniLibs** . Ora dovresti avere una **directory cvtest1 / OpenCVLibrary310 / src / main / jniLibs** . Risincronizza il tuo progetto e questa directory dovrebbe ora apparire nella vista del progetto sotto **openCVLibrary310** .



10. Vai al metodo *onCreate* di *MainActivity.java* e aggiungi questo codice:

```
if (!OpenCVLoader.initDebug()) {
        Log.e(this.getClass().getSimpleName(), "  OpenCVLoader.initDebug(), not working.");
    } else {
        Log.d(this.getClass().getSimpleName(), "  OpenCVLoader.initDebug(), working.");
    }
```

Quindi esegui la tua applicazione. Dovresti vedere linee come questa nel monitor Android:



(Non so perché quella riga con il messaggio di errore è lì)

11. Ora prova a utilizzare effettivamente un codice openCV. Nell'esempio seguente ho copiato un file .jpg nella directory cache dell'applicazione cvtest1 sull'emulatore Android. Il codice sotto carica questa immagine, esegue l'algoritmo di rilevamento dei bordi canny e quindi scrive i risultati in un file .png nella stessa directory.

```
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. Esegui la tua applicazione. Il tuo emulatore dovrebbe creare un'immagine "bordo" in bianco e nero. Puoi utilizzare Android Device Monitor per recuperare l'output o scrivere un'attività per mostrarlo.

Leggi Integrare OpenCV in Android Studio online:
https://riptutorial.com/it/android/topic/7068/integrare-opencv-in-android-studio

Capitolo 130: Integrazione del gateway Paypal per Android

Osservazioni

Paypal ci fornisce la propria libreria per il pagamento, quindi ora è molto più sicura e facile da implementare nella nostra applicazione. Di seguito sono riportati i passaggi importanti da fare.

Examples

Imposta paypal nel tuo codice Android

1) Prima vai sul sito web dello sviluppatore di Paypal e crea un'applicazione.

2) Ora apri il file manifest e dai i permessi di sotto

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

3) E alcune attività e servizi richiesti-

```
 <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) Apri la tua classe di attività e imposta la configurazione per la tua app-

```
//set the environment for production/sandbox/no netowrk
 private static final String CONFIG_ENVIRONMENT = PayPalConfiguration.ENVIRONMENT_PRODUCTION;
```

5) Ora imposta l'ID cliente dall'account sviluppatore Paypal - Stringa finale statica privata CONFIG_CLIENT_ID = "METTI IL TUO ID CLIENTE"; 6) All'interno del metodo onCreate chiamare il servizio Paypal - Intento intento = nuovo intento (questo, PayPalService.class); intent.putExtra (PayPalService.EXTRA_PAYPAL_CONFIGURATION, config); StartService (intento);

7) Ora sei pronto per effettuare un pagamento solo sul pulsante premi chiama l'attività di pagamento-

```
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) E infine da onActivityResult ottieni la risposta di pagamento-

```
 @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.");
                   }
               }


           }
```

Leggi Integrazione del gateway Paypal per Android online:
https://riptutorial.com/it/android/topic/5895/integrazione-del-gateway-paypal-per-android

Capitolo 131: Integrazione di accesso Google su Android

introduzione

Questo argomento è basato su Come integrare l'accesso a google, su app Android

Examples

Integrazione di Google Auth nel tuo progetto. (Ottieni un file di configurazione)

In primo luogo ottenere il file di configurazione per l'accesso da

Apri il link qui sotto

[https://developers.google.com/identity/sign-in/android/start-integrating][1]

fai clic su Ottieni un file di configurazione

- Inserisci il nome dell'app e il nome del pacchetto e fai clic su scegli e configura servizi
- fornire SHA1 Abilitare google SIGNIN e generare file di configurazione

Scarica il file di configurazione e posiziona il file in app / cartella del tuo progetto

1. Aggiungi la dipendenza al tuo build.gradle a livello di progetto:

   classpath "com.google.gms: google-services: 3.0.0"

2. Aggiungi il plug-in al tuo build.gradle a livello di app: (in basso)

   applica il plug-in: "com.google.gms.google-services"

3. aggiungi questa dipendenza al tuo file gradle dell'app

   dipendencies {compile 'com.google.android.gms: play-services-auth: 9.8.0'}

Implementazione del codice Google SignIn

- Nel metodo onCreate dell'attività di accesso, configura l'accesso a Google per richiedere i dati utente richiesti dalla tua app.

```
 GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestEmail()
        .build();
```

- crea un oggetto GoogleApiClient con accesso all'API di accesso a Google e alle opzioni specificate.

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
       .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
       .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
       .build();
```

- Ora, quando l'utente fa clic sul pulsante di accesso Google, chiama questa funzione.

```
 private void signIn() {
 Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
 startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

- implementare OnActivityResult per ottenere la risposta.

```
 @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);
    }
}
```

- Ultimo passo Gestire il risultato e ottenere i dati dell'utente

```
 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);
 }
}
```

Leggi Integrazione di accesso Google su Android online:
https://riptutorial.com/it/android/topic/9960/integrazione-di-accesso-google-su-android

Capitolo 132: Intenti impliciti

Sintassi

- Intento()
- Intento (intenzione o)
- Intento (String action)
- Intento (String action, Uri uri)
- Intento (Context packageContext, Class <?> Cls)
- Intento (String action, Uri uri, Context packageContext, Class <?> Cls)

Parametri

parametri	Dettagli
o	Intento
azione	`String`: l'azione Intent, ad esempio `ACTION_VIEW`.
uri	`Uri`: l'URI di dati Intent.
packageContext	`Context`: un contesto del pacchetto dell'applicazione che implementa questa classe.
cls	`Class`: la classe componente che deve essere utilizzata per l'intento.

Osservazioni

- Maggiori informazioni su Intent
- Maggiori informazioni sui tipi di intenti

Examples

Intenti impliciti ed espliciti

Un intento esplicito viene utilizzato per avviare un'attività o un servizio all'interno dello stesso pacchetto dell'applicazione. In questo caso il nome della classe desiderata è esplicitamente menzionato:

```
Intent intent = new Intent(this, MyComponent.class);
startActivity(intent);
```

Tuttavia, un intento implicito viene inviato attraverso il sistema per tutte le applicazioni installate

sul dispositivo dell'utente in grado di gestirlo. Questo è usato per condividere informazioni tra diverse applicazioni.

```
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
}
```

Maggiori dettagli sulle differenze sono disponibili nei documenti per sviluppatori Android qui: Risoluzione intenzionale

Intenti impliciti

Gli intent impliciti non nominano un componente specifico, ma dichiarano invece un'azione generale da eseguire, che consente a un componente di un'altra app di gestirlo.

Ad esempio, se si desidera mostrare all'utente una posizione su una mappa, è possibile utilizzare un intento implicito per richiedere che un'altra app in grado di mostrare una posizione specificata su una mappa.

Esempio:

```
// 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);
}
```

Leggi Intenti impliciti online: https://riptutorial.com/it/android/topic/5336/intenti-impliciti

Capitolo 133: Intento

introduzione

Un intento è un piccolo messaggio passato attorno al sistema Android. Questo messaggio può contenere informazioni sulla nostra intenzione di eseguire un'attività.

È fondamentalmente una struttura dati passiva che contiene una descrizione astratta di un'azione da eseguire.

Sintassi

- Intento intenzionale ()
- Intento intenzionale (intento intenzionale)
- Intento intento (azione String)
- Intent Intent (String action, Uri uri)
- Intento Intento (Context packageContext, Class <?> Cls)
- Intent Intent (String action, Uri uri, Context packageContext, Class <?> Cls)
- void startActivity (intento intenzionale)
- void startActivity (intent intent, opzioni bundle)
- void startActivityForResult (Intent intention, int requestCode)
- void startActivityForResult (Intent intent, int requestCode, Opzioni bundle)
- Intent putExtra (nome stringa, valore doppio [])
- Intent putExtra (nome stringa, valore int)
- Intent putExtra (nome stringa, valore CharSequence)
- Intent putExtra (nome stringa, valore char)
- Intent putExtra (nome stringa, valore bundle)
- Intent putExtra (nome stringa, valore Parcelable [])
- Intent putExtra (nome stringa, valore serializzabile)
- Intent putExtra (nome stringa, valore int [])
- Intent putExtra (nome stringa, valore float)
- Intent putExtra (nome stringa, valore byte [])
- Intent putExtra (nome stringa, valore [] lungo)
- Intent putExtra (Nome stringa, valore Parcelable)
- Intent putExtra (nome stringa, valore float [])
- Intent putExtra (nome stringa, valore lungo)
- Intent putExtra (nome stringa, valore String [])
- Intent putExtra (nome stringa, valore booleano)
- Intent putExtra (nome stringa, valore booleano [])
- Intent putExtra (String name, short value)
- Intent putExtra (String name, double value)
- Intent putExtra (nome stringa, valore [] breve)
- Intent putExtra (nome stringa, valore stringa)
- Intent putExtra (nome stringa, valore byte)

- Intent putExtra (nome stringa, valore char [])
- Intent putExtra (nome stringa, valore CharSequence [])

Parametri

Parametro	Dettagli
intento	L'intento di iniziare
codice richiesto	Numero univoco per identificare la richiesta
opzioni	Ulteriori opzioni su come deve essere avviata l'attività
nome	Il nome dei dati extra
valore	Il valore dei dati extra
CHOOSE_CONTACT_REQUEST_CODE	il codice della richiesta, per identificarlo sul metodo `onActivityResult`
azione	Qualsiasi azione da eseguire tramite questo intento, es: Intent.ACTION_VIEW
uri	dati uri da utilizzare intenzionalmente per eseguire azioni specifiche
packageContext	Contesto da utilizzare per inizializzare l'intento
cls	Classe da utilizzare con questo intento

Osservazioni

Avvertenze sull'uso di intenti impliciti

Quando si chiama un intento implicito è sempre utile verificare se è possibile dal sistema gestirlo.

Questo può essere fatto controllando usando `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
}
```

Attività di partenza che è un `singleTask` o `singleTop`

Quando la modalità di avvio dell'attività è `singleTask` o `singleTop` , l' `onActivityResult` verrà chiamato non appena l'attività viene avviata con un dato null. Per impedire ciò, utilizzare `Intent.setFlags(0)` per reimpostare i flag predefiniti.

Examples

Inizia un'attività

Questo esempio avvierà `DestinationActivity` da `OriginActivity` .

Qui, il costruttore `Intent` prende due parametri:

1. Un contesto come primo parametro (questo viene utilizzato perché la classe Attività è una sottoclasse di Context)
2. La classe del componente dell'app a cui il sistema deve fornire l'intento (in questo caso, l'attività che deve essere avviata)

```
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.
    }
}
```

Un altro modo per creare l' `Intent` di aprire `DestinationActivity` consiste nell'usare il costruttore predefinito per l' `Intent` e utilizzare il metodo `setClass()` per dirgli quale attività aprire:

```
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
```

Trasmissione dei dati tra le attività

Questo esempio illustra l'invio di una `String` con valore come `"Some data!"` da `OriginActivity` a `DestinationActivity` .

NOTA: questo è il modo più semplice per inviare dati tra due attività. Vedere l'esempio sull'uso del pattern di partenza per un'implementazione più robusta.

OriginActivity

```
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);
    }
}
```

DestinationActivity

```
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);
    }
}
```

È anche possibile passare altri tipi di dati `primitive` oltre a `arrays` , dati `Bundle` e `Parcelable` . Passare `Serializable` è anche possibile, ma dovrebbe essere evitato in quanto è più di tre volte più lento di `Parcelable` .

Serializable è `interface` Java standard. Contrassegnare semplicemente una classe come `Serializable` implementando l' `interface Serializable` e Java la serializzerà automaticamente durante le situazioni richieste.

Parcelable è una specifica Android `interface` che può essere implementato su tipi di dati personalizzati (vale a dire i propri oggetti / oggetti POJO), consente l'oggetto sia appiattito e si ricostruiscono, senza la destinazione bisogno di fare nulla. C'è un esempio di documentazione per rendere un oggetto parcellizzabile .

Una volta che hai un oggetto `parcelable` puoi inviarlo come un tipo primitivo, con un oggetto intent:

```
intent.putExtra(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

O in un pacchetto / come argomento per un frammento:

```
bundle.putParcelable(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

e poi anche leggerlo dall'intenzione della destinazione usando getParcelableExtra:

```
final MyParcelableType data = intent.getParcelableExtra(EXTRA_DATA);
```

O quando si legge in un frammento di un pacchetto:

```
final MyParcelableType data = bundle.getParcelable(EXTRA_DATA);
```

Una volta che hai un oggetto `Serializable` puoi metterlo in un oggetto intent:

```
bundle.putSerializable(DestinationActivity.EXTRA_DATA, mySerializableObject);
```

e quindi anche leggerlo dall'oggetto intent alla destinazione come mostrato di seguito:

```
final SerializableType data = (SerializableType)bundle.getSerializable(EXTRA_DATA);
```

Inviando email

```
// 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
}
```

Questo pre-riempirà un'e-mail in un'app di posta elettronica a scelta dell'utente.

Se è necessario aggiungere un allegato, è possibile utilizzare `Intent.ACTION_SEND` anziché `Intent.ACTION_SENDTO` . Per più allegati puoi utilizzare `ACTION_SEND_MULTIPLE`

Una parola di cautela: non tutti i dispositivi dispongono di un provider per `ACTION_SENDTO` e la

chiamata `startActivity()` senza verificare con `resolveActivity()` prima potrebbe lanciare `resolveActivity()` ActivityNotFoundException.

Ottenere un risultato da un'altra attività

Utilizzando `startActivityForResult(Intent intent, int requestCode)` è possibile avviare un'altra `Activity` e quindi ricevere un risultato da tale `Activity` nel `onActivityResult(int requestCode, int resultCode, Intent data)` . Il risultato sarà restituito come `Intent` . Un intento può contenere dati tramite un pacchetto

---

In questo esempio, `MainActivity` avvierà un'attività `DetailActivity` e quindi si aspetta un risultato da essa. Ogni tipo di richiesta deve avere il proprio `int` codice di richiesta, in modo che nel *sovrascritto* `onActivityResult(int requestCode, int resultCode, Intent data)` metodo `MainActivity` , può essere determinato che chiedono di elaborare confrontando valori di `requestCode` e `REQUEST_CODE_EXAMPLE` (anche se in questo esempio, ce n'è solo uno).

Attività principale:

```
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.
            }
        }
    }
}
```

DetailActivity:

```
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();
    }
}
```

Alcune cose che devi sapere:

- I dati vengono restituiti solo dopo aver chiamato `finish()` . Devi chiamare `setResult()` prima di chiamare `finish()` , altrimenti non verrà restituito alcun risultato.

- Assicurati che la tua `Activity` non stia utilizzando `android:launchMode="singleTask"` , altrimenti l' `Activity` verrà eseguita in un'attività separata e pertanto non riceverai un risultato da essa. Se la tua `Activity` utilizza `singleTask` come modalità di avvio, chiamerà `onActivityResult()` immediatamente con un codice risultato di `Activity.RESULT_CANCELED` .

- `android:launchMode="singleInstance"` attenzione quando usi `android:launchMode="singleInstance"` . Sui dispositivi prima di Lollipop (Android 5.0, Livello API 21), le attività non restituiscono un risultato.

- È possibile utilizzare intenti espliciti o impliciti quando si chiama `startActivityForResult()` . Quando si avvia una delle proprie attività per ricevere un risultato, è necessario utilizzare un intento esplicito per assicurarsi di ricevere il risultato previsto. Un `intent` esplicito viene sempre consegnato al suo obiettivo, indipendentemente da cosa contenga; il `filter` non viene consultato. Ma un intento implicito viene consegnato a un componente solo se può passare attraverso uno dei filtri del componente.

Apri un URL in un browser

Apertura con il browser predefinito

Questo esempio mostra come è possibile aprire un URL a livello di programmazione nel browser Web incorporato anziché all'interno dell'applicazione. Ciò consente all'app di aprire una pagina Web senza la necessità di includere l'autorizzazione `INTERNET` nel file manifest.

```
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);
    }
}
```

Chiedere all'utente di selezionare un browser

Nota che questo esempio usa il metodo `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"));

}
```

In alcuni casi, l'URL può iniziare con **"www"** . In questo caso otterrai questa eccezione:

`android.content.ActivityNotFoundException` : Nessuna attività trovata per gestire Intento

L'URL deve sempre iniziare con **"http: //"** o **"https: //"** . Il tuo codice dovrebbe quindi controllarlo, come mostrato nel seguente frammento di codice:

```
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);
}
```

Migliori pratiche

Controlla se sul dispositivo non ci sono app che possono ricevere l'intento implicito. In caso contrario, la tua app si bloccherà quando chiama `startActivity()` . Per prima verificare che esista un'app per ricevere l'intento, chiama `resolveActivity()` sull'oggetto Intent. Se il risultato non è nullo, c'è almeno un'app in grado di gestire l'intento ed è sicuro chiamare `startActivity()` . Se il risultato è nullo, non si dovrebbe usare l'intento e, se possibile, si dovrebbe disabilitare la funzione che richiama l'intento.

Cancellare una pila di attività

A volte potresti voler iniziare una nuova attività rimuovendo le attività precedenti dallo stack posteriore, quindi il pulsante Indietro non ti riporta indietro. Un esempio potrebbe essere l'avvio di un'app sull'attività di accesso, che ti porterà all'attività principale della tua applicazione, ma alla disconnessione vuoi essere reindirizzato al login senza la possibilità di tornare indietro. In un caso del genere puoi impostare il flag `FLAG_ACTIVITY_CLEAR_TOP` per l'intento, ovvero se l'attività avviata è già in esecuzione nell'attività corrente (LoginActivity), quindi anziché avviare una nuova istanza di tale attività, tutte le altre attività in alto di esso sarà chiuso e questo Intento sarà consegnato alla vecchia attività (ora in cima) come un nuovo Intento.

```
Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
```

È inoltre possibile utilizzare le flag `FLAG_ACTIVITY_NEW_TASK` insieme a `FLAG_ACTIVITY_CLEAR_TASK` se si desidera cancellare tutte le attività nel back stack:

```
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);
```

URI intenzionale

Questo esempio mostra come avviare l'intenzione dal browser:

```
<a href="intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end">Start
intent</a>
```

Questo intento avvierà l'app con il pacchetto `com.sample.test` o aprirà google play con questo pacchetto.

Anche questo intento può essere avviato con javascript:

```
var intent = "intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end";
window.location.replace(intent)
```

Nell'attività questo host e percorso possono essere ottenuti dai dati di intenti:

```
@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    Uri data = getIntent().getData(); // returns host.com/path
}
```

Sintassi URI Intent:

```
HOST/URI-path // Optional host
#Intent;
    package=[string];
    action=[string];
    category=[string];
    component=[string];
    scheme=[string];
end;
```

Trasmissione di messaggi ad altri componenti

Gli intenti possono essere utilizzati per trasmettere messaggi ad altri componenti della tua applicazione (come un servizio in background in esecuzione) o all'intero sistema Android.

Per inviare una trasmissione **all'interno dell'applicazione** , utilizzare la classe `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);
```

Per inviare una trasmissione a componenti esterni all'applicazione, utilizzare il metodo `sendBroadcast()` su un oggetto `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);
```

Informazioni sulla *ricezione di* trasmissioni possono essere trovate qui: Broadcast Receiver

CustomTabsIntent per le schede personalizzate di Chrome

4.0.3

Utilizzando `CustomTabsIntent` , è ora possibile configurare le schede personalizzate di Chrome per personalizzare i componenti dell'interfaccia utente chiave nel browser aperto dall'app.

Questa è una buona alternativa all'uso di una WebView per alcuni casi. Permette il caricamento di una pagina web con un intento, con la possibilità aggiunta di iniettare un certo grado di aspetto della tua app nel browser.

Ecco un esempio di come aprire un 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:

Per utilizzare le schede personalizzate, è necessario aggiungere questa dipendenza a build.gradle

```
compile 'com.android.support:customtabs:24.1.1'
```

Condivisione di più file tramite Intent

L'elenco delle stringhe passato come parametro al metodo `share()` contiene i percorsi di tutti i file che si desidera condividere.

In pratica, scorre i percorsi, li aggiunge a Uri e avvia l'attività che può accettare file di questo 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);
    }
```

Motivo di partenza

Questo modello è un approccio più rigoroso per avviare un `Activity` . Il suo scopo è migliorare la leggibilità del codice, riducendo allo stesso tempo la complessità del codice, i costi di manutenzione e l'accoppiamento dei componenti.

L'esempio seguente implementa il pattern di partenza, che di solito viene implementato come metodo statico `Activity` stessa. Questo metodo statico accetta tutti i parametri richiesti, costruisce un `Intent` valido da quei dati e quindi avvia l' `Activity` .

Un `Intent` è un oggetto che fornisce il runtime vincolante tra componenti separati, come due attività. L'intento rappresenta "l'intenzione di fare qualcosa" di un'app. Puoi usare gli intent per una vasta gamma di attività, ma qui, il tuo intento inizia un'altra attività.

```
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);
    }
}
```

Questo modello consente inoltre di forzare la trasmissione di ulteriori dati con l'intento.

L' `ExampleActivity` può quindi essere avviato in questo modo, dove `context` è un contesto di attività:

```
ExampleActivity.start(context, "Some data!");
```

Avvia il servizio non associato utilizzando un intent

Un servizio è un componente che viene eseguito in background (sul thread dell'interfaccia utente) senza interazione diretta con l'utente. Un servizio non associato è appena iniziato e non è legato al ciclo di vita di alcuna attività.

Per avviare un Servizio puoi fare come mostrato nell'esempio qui sotto:

```
// 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);
```

Puoi utilizzare qualsiasi extra dall'intento utilizzando 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;
    }

}
```

Condividi l'intento

Condividi semplici informazioni con diverse app.

```
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)));
```

Condividi un'immagine con diverse app.

```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uriToImage);
shareIntent.setType("image/jpeg");
```

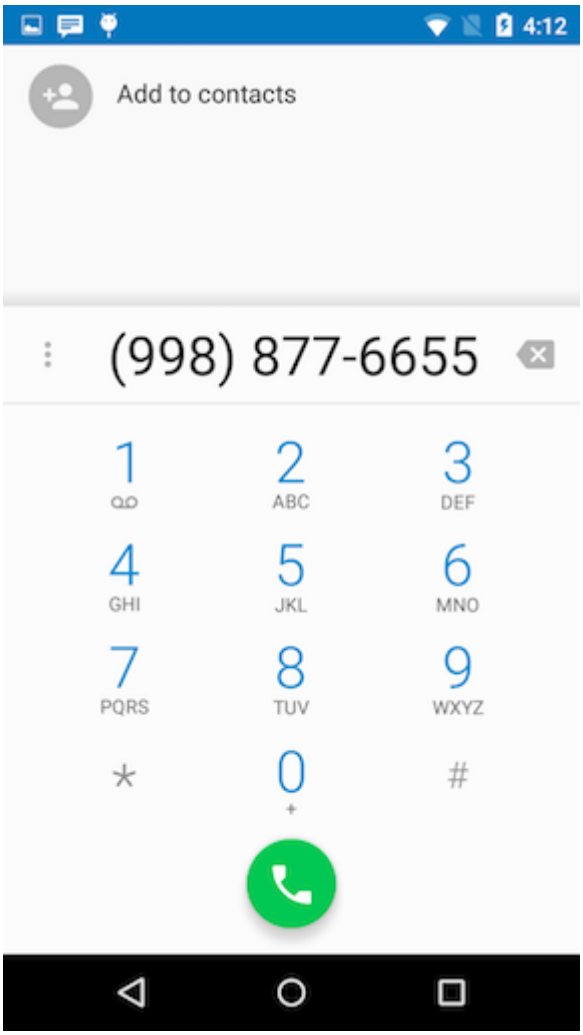
```
startActivity(Intent.createChooser(shareIntent, getResources().getText(R.string.send_to)));
```

Avvia il dialer

Questo esempio mostra come aprire un dialer predefinito (un'app che effettua chiamate regolari) con un numero telefonico fornito già presente:

```
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);
```

Risultato dall'esecuzione del codice qui sopra:



Apri Google map con latitudine, longitudine specificate

Puoi passare latitudine, longitudine dalla tua app 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);
```

Trasmissione di dati diversi tramite Intent in Activity

1. Passare dati interi:

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. Passando doppio dati:

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. Passare dati stringa:

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. Passare i dati di 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. Passare dati oggetto:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("ObjectVariableName", yourObject);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
yourObj = mIntent.getSerializableExtra("ObjectVariableName");
```

> **Nota:** tenere presente che la classe personalizzata deve implementare l'interfaccia `Serializable` .

6. Passaggio di dati di HashMap <String, String>:

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. Passaggio dei dati bitmap:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("image",bitmap);
startActivity(mIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
Bitmap bitmap = mIntent.getParcelableExtra("image");
```

Mostrare un File Chooser e leggere il risultato

Avvio di un'attività Selezione file

```
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);
}
```

Leggendo il risultato

```
@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;
}
```

Passaggio dell'oggetto personalizzato tra le attività

È anche possibile passare il tuo oggetto personalizzato ad altre attività usando la classe `Bundle` .

Ci sono due modi:

- Interfaccia `Serializable` : per Java e Android
- `Parcelable` memoria interfaccia `Parcelable` , solo per Android (consigliato)

Parcelable

L'elaborazione parcelable è molto più veloce di serializzabile. Una delle ragioni di ciò è che siamo espliciti sul processo di serializzazione invece di usare la riflessione per dedurlo. È anche ovvio che il codice è stato fortemente ottimizzato per questo scopo.

```
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);
```

```
        }
    };
}
```

Invio del codice attività

```
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);
```

Ricevere l'oggetto nell'attività di destinazione.

```
//Getting MyObjects
Intent mIntent = getIntent();
MyObjects workorder = (MyObjects) mIntent.getParcelable("UniqueKey");
```

Puoi passare l'oggetto Arrayylist di Parceble come sotto

```
//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:** esistono plugin Android Studio come questo disponibili per generare codice Parcelable

Serializable

Invio del codice attività

```
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);
```

Ricevere l'oggetto nell'attività di destinazione.

```
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` di oggetto serializzabile: uguale al passaggio di un singolo oggetto

L'oggetto personalizzato dovrebbe implementare l'interfaccia `Serializable` .

Ottenere un risultato da Activity to Fragment

Come ottenere un risultato da un'altra attività, devi chiamare il metodo di `Fragment` `startActivityForResult(Intent intent, int requestCode)` . notare che non si dovrebbe chiamare `getActivity().startActivityForResult()` come questo richiederà il risultato di nuovo al `Fragment` genitore 's `Activity` .

La ricezione del risultato può essere eseguita utilizzando il metodo `Fragment` `onActivityResult()` . È necessario assicurarsi che l'attività padre di Fragment sovrascriva `onActivityResult()` e chiama la sua `super` implementazione.

Nell'esempio seguente `ActivityOne` contiene `FragmentOne` , che avvierà `ActivityTwo` e si aspetta un risultato da esso.

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
        }
    }
}
```

ActivityTwo

```
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();
    }
}
```

Leggi Intento online: https://riptutorial.com/it/android/topic/103/intento

Capitolo 134: IntentService

Sintassi

4. <servizio android: name = ". UploadS3IntentService" android: exported = "false" />

Osservazioni

Un `IntentService` fornisce un modo semplice per scaricare il lavoro su un thread in background. Gestisce tutto ciò che riguarda la ricezione delle richieste, la loro messa in coda, l'arresto di se stesso, ecc. È anche facile da implementare, rendendolo la cosa perfetta da usare quando si eseguono operazioni che richiedono tempo e che non appartengono al thread Main (UI).

Examples

Creare un IntentService

Per creare un IntentService, creare una classe che estenda `IntentService` e al suo interno un metodo che sovrascrive `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.
     }
}
```

Servizio di esempio di esempio

Ecco un esempio di `IntentService` che finge di caricare le immagini in background. Tutto ciò che devi fare per implementare un `IntentService` è quello di fornire un costruttore che chiama il costruttore `super(String)` e devi implementare il `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
    }
}
```

Per avviare un `IntentService` , devi inviare un `Intent` ad esso. Puoi farlo da `Activity` , per un esempio. Certo, non sei limitato a questo. Ecco un esempio di come invocare il nuovo `Service` da una classe di `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
```

`IntentService` elabora `IntentService` i dati dai propri `Intent` , in modo da poter inviare più `Intent` senza preoccuparsi che si scontrino tra loro. Solo un `Intent` alla volta viene elaborato, il resto va in coda. Quando tutti i lavori sono completi, `IntentService` si spegnerà automaticamente.

Esempio di Basic IntentService

La classe astratta `IntentService` è una classe base per servizi, che vengono eseguiti in background senza alcuna interfaccia utente. Pertanto, al fine di aggiornare l'interfaccia utente, dobbiamo utilizzare un ricevitore, che può essere un `BroadcastReceiver` o un `ResultReceiver` :

- Un `BroadcastReceiver` deve essere utilizzato se il servizio deve comunicare con più componenti che desiderano ascoltare la comunicazione.
- A `ResultReceiver` : deve essere utilizzato se il servizio deve comunicare solo con l'applicazione padre (ovvero l'applicazione).

All'interno di `IntentService` , abbiamo un metodo chiave, `onHandleIntent()` , in cui faremo tutte le azioni, ad esempio preparando notifiche, creando allarmi, ecc.

Se si desidera utilizzare il proprio `IntentService` , è necessario estenderlo come segue:

```
public class YourIntentService extends IntentService {
    public YourIntentService () {
        super("YourIntentService ");
    }

    @Override
```

```
    protected void onHandleIntent(Intent intent) {
        // TODO: Write your own code here.
    }
}
```

La chiamata / avvio dell'attività può essere eseguita come segue:

```
Intent i = new Intent(this, YourIntentService.class);
startService(i);  // For the service.
startActivity(i); // For the activity; ignore this for now.
```

Simile a qualsiasi attività, è possibile passare informazioni aggiuntive come dati bundle ad esso come segue:

```
Intent passDataIntent = new Intent(this, YourIntentService.class);
msgIntent.putExtra("foo","bar");
startService(passDataIntent);
```

Supponiamo ora di aver passato alcuni dati alla classe `YourIntentService` . Sulla base di questi dati, un'azione può essere eseguita come segue:

```
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.
        }
    }
}
```

Il codice sopra mostra anche come gestire i vincoli nel metodo `OnHandleIntent()` .

Leggi IntentService online: https://riptutorial.com/it/android/topic/5319/intentservice

Capitolo 135: interfacce

Examples

Listener personalizzato

Definisci interfaccia

```
//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);
}
```

Crea listener

Nel passaggio successivo è necessario definire una variabile di istanza nell'oggetto che invierà la richiamata tramite `MyCustomListener` . E aggiungi setter per il nostro ascoltatore.

```
public class SampleClassB {
    private MyCustomListener listener;

    public void setMyCustomListener(MyCustomListener listener) {
        this.listener = listener;
    }
}
```

Implementa l'ascoltatore

Ora, in un'altra classe, possiamo creare un'istanza di `SampleClassB` .

```
public class SomeActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
    }
}
```

successivamente possiamo impostare il nostro listener, su `sampleClass` , in due modi:

da implementa `MyCustomListener` nella nostra classe:

```
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 semplicemente istanziare una classe interiore anonima:

```
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) {

            }
        });
    }
}
```

Trigger listener

```
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());
            }
        });
    }
}
```

Listener di base

Il pattern "listener" o "observer" è la strategia più comune per la creazione di callback asincroni nello sviluppo 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!");
 }
}
```

Ora sulla tua attività:

```
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);
            }
        });
    }
}
```

Leggi interfacce online: https://riptutorial.com/it/android/topic/1785/interfacce

Capitolo 136: Internazionalizzazione e localizzazione (I18N e L10N)

introduzione

Internazionalizzazione (i18n) e localizzazione (L10n) vengono utilizzati per adattare il software in base alle differenze nelle lingue, nelle differenze regionali e nel pubblico di destinazione.

Internazionalizzazione: il processo di pianificazione per la localizzazione futura, ovvero rendere flessibile la progettazione del software in modo tale da adattarlo e adattarsi agli sforzi futuri di localizzazione.

Localizzazione: il processo di adattamento del software a una particolare regione / paese / mercato (locale).

Osservazioni

Per testare un dispositivo per la localizzazione, è possibile riavviare il dispositivo o l'emulatore in un particolare locale usando `adb` come segue:

1. Esegui adb usando il comando: `adb shell`
2. Eseguire il comando seguente al prompt dei comandi di adb: `setprop persist.sys.locale [BCP-47 language tag];stop;sleep 5;start` dove [tag lingua BCP-47] è il codice specifico della lingua come descritto qui: codici BCP47

ad esempio per verificare la localizzazione giapponese nell'app, utilizzare il comando: `setprop persist.sys.locale ja-JP;stop;sleep 5;start`

Examples

Pianificazione per la localizzazione: abilitare il supporto RTL in Manifest

Il supporto RTL (da destra a sinistra) è una parte essenziale nella pianificazione di i18n e L10n. A differenza della lingua inglese scritta da sinistra a destra, molte lingue come arabo, giapponese, ebraico, ecc. Sono scritte da destra a sinistra. Per attirare un pubblico più globale, è una buona idea pianificare i layout in modo da supportare questi linguaggi fin dall'inizio del progetto, in modo che l'aggiunta della localizzazione sia più semplice in seguito.

Il supporto RTL può essere abilitato in un'app Android aggiungendo il tag `supportsRtl` `AndroidManifest` , in questo modo:

```
<application
    ...
    android:supportsRtl="true"
    ...>
```

```
...
</application>
```

Pianificazione per la localizzazione: aggiungi il supporto RTL in Layouts

L'avvio dell'SDK 17 (Android 4.2), il supporto RTL è stato aggiunto nei layout Android ed è una parte essenziale della localizzazione. Andando avanti, la notazione `left/right` nei layout dovrebbe essere sostituita dalla notazione `start/end` . Se, tuttavia, il progetto ha un valore `minSdk` inferiore a `17` , è necessario utilizzare sia la `left/right` che la notazione `start/end` nei layout.

Per i layout relativi, devono essere utilizzati `alignParentStart` e `alignParentEnd` , in questo modo:

```
<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>
```

Per specificare la gravità e la gravità del layout, dovrebbe essere usata una notazione simile, in questo modo:

```
    <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"/>
```

Paddings e margini dovrebbero anche essere specificati di conseguenza, in questo modo:

```
<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"/>
```

Pianificazione per la localizzazione: test di layout per RTL

Per verificare se i layout che sono stati creati sono compatibili RTL, effettuare le seguenti operazioni:

> Vai a Impostazioni -> Opzioni sviluppatore -> Disegno -> Forza la direzione del layout RTL

L'attivazione di questa opzione costringerebbe il dispositivo a utilizzare le impostazioni locali RTL e sarà possibile verificare facilmente tutte le parti dell'app per il supporto RTL. Si noti che non è necessario aggiungere effettivamente alcun nuovo locale / supporto linguistico fino a questo punto.

Coding per localizzazione: creazione di stringhe e risorse predefinite

Il primo passo per la codifica per la localizzazione è creare risorse predefinite. Questo passaggio è così implicito che molti sviluppatori non ci pensano nemmeno. Tuttavia, la creazione di risorse predefinite è importante perché se il dispositivo viene eseguito su una locale non supportata, carica tutte le sue risorse dalle cartelle predefinite. Se manca anche una delle risorse dalle cartelle predefinite, l'app si arresterebbe semplicemente.

L'insieme predefinito di stringhe deve essere inserito nella seguente cartella nel percorso specificato:

```
res/values/strings.xml
```

Questo file dovrebbe contenere le stringhe nella lingua che gli utenti maggioritari dell'app dovrebbero parlare.

Inoltre, le risorse predefinite per l'app devono essere posizionate nelle seguenti cartelle e posizioni:

```
res/drawable/
res/layout/
```

Se la tua app richiede cartelle come `anim` o `xml` , le risorse predefinite devono essere aggiunte alle seguenti cartelle e posizioni:

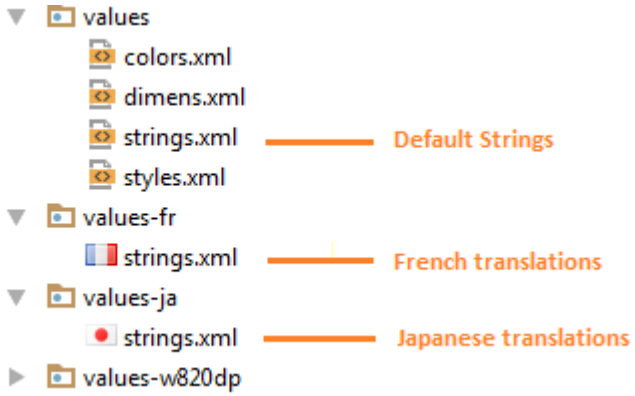
```
res/anim/
res/xml/
res/raw/
```

Codifica per localizzazione: fornitura di stringhe alternative

Per fornire traduzioni in altre lingue (locales), abbiamo bisogno di creare un file `strings.xml` in una cartella separata dalla seguente convenzione:

```
res/values-<locale>/strings.xml
```

Di seguito è riportato un esempio per lo stesso:



In questo esempio, abbiamo le stringhe inglesi predefinite nel file `res/values/strings.xml` , le traduzioni francesi sono fornite nella cartella `res/values-fr/strings.xml` e le traduzioni giapponesi sono fornite nella cartella `res/values-ja/strings.xml`

Altre traduzioni per altre impostazioni locali possono essere aggiunte all'app.

> Un elenco completo dei codici locali può essere trovato qui: codici ISO 639

Stringhe non traducibili:

Il tuo progetto potrebbe avere certe stringhe che non devono essere tradotte. Le stringhe utilizzate come chiavi per SharedPreferences o stringhe utilizzate come simboli rientrano in questa categoria. Queste stringhe devono essere memorizzate solo nel file `strings.xml` predefinito e devono essere contrassegnate con un attributo `translatable="false"` . per esempio

```
<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>
```

Questo attributo è importante perché le traduzioni vengono spesso eseguite da professionisti bilingui. Ciò consentirebbe a queste persone coinvolte nelle traduzioni di identificare stringhe che non devono essere tradotte, risparmiando così tempo e denaro.

Codifica per localizzazione: fornitura di layout alternativi

La creazione di layout specifici per la lingua spesso non è necessaria se è stata specificata la notazione di `start/end` corretta, come descritto nell'esempio precedente. Tuttavia, potrebbero esserci situazioni in cui i layout di default potrebbero non funzionare correttamente per alcune lingue. A volte, i layout da sinistra a destra potrebbero non essere tradotti per le lingue RTL. È

necessario fornire i layout corretti in questi casi.

Per fornire completa ottimizzazione per i layout RTL, siamo in grado di utilizzare i file di layout completamente separati utilizzando il `ldrtl` qualificatore risorsa (`ldrtl` acronimo di layout direzione da destra a sinistra}). Ad esempio, possiamo salvare i file di layout di default in `res/layout/` e i nostri layout ottimizzati RTL in `res/layout-ldrtl/` .

Il qualificatore `ldrtl` è ottimo per risorse estraibili, in modo da poter fornire grafici orientati nella direzione corrispondente alla direzione di lettura.

Ecco un ottimo post che descrive la precedenza dei layout di `ldrtl` : layout specifici per la lingua

Leggi Internazionalizzazione e localizzazione (I18N e L10N) online: https://riptutorial.com/it/android/topic/8796/internazionalizzazione-e-localizzazione--i18n-e-l10n-

Capitolo 137: Jackson

introduzione

Jackson è una libreria Java multiuso per l'elaborazione di JSON. Jackson mira ad essere la migliore combinazione possibile di veloce, corretto, leggero ed ergonomico per gli sviluppatori.

Caratteristiche di Jackson:

Modalità di elaborazione multipla e ottima collaborazione

Non solo annotazioni, ma anche annotazioni miste

Supporta pienamente i tipi generici

Supporta tipi polimorfi

Examples

Esempio di binding completo dei dati

Dati JSON

```
{
  "name" : { "first" : "Joe", "last" : "Sixpack" },
  "gender" : "MALE",
  "verified" : false,
  "userImage" : "keliuyue"
}
```

Occorrono due righe di Java per trasformarla in un'istanza utente:

```
ObjectMapper mapper = new ObjectMapper(); // can reuse, share globally
User user = mapper.readValue(new File("user.json"), User.class);
```

User.class

```
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;
    }
}
```

Effettuare il marshalling verso JSON è altrettanto semplice:

```
mapper.writeValue(new File("user-modified.json"), user);
```

Leggi Jackson online: https://riptutorial.com/it/android/topic/10878/jackson

Capitolo 138: Java su Android

introduzione

Android supporta tutte le funzionalità del linguaggio Java 7 e un sottoinsieme di funzionalità del linguaggio Java 8 che variano in base alla versione della piattaforma. Questa pagina descrive le nuove funzionalità linguistiche che puoi utilizzare, come configurare correttamente il tuo progetto per usarle e tutti i problemi noti che potresti incontrare.

Examples

Funzionalità Java 8 sottoinsiemi con Retrolambda

Retrolambda consente di eseguire codice Java 8 con espressioni lambda, riferimenti al metodo e istruzioni try-with-resources su Java 7, 6 o 5. Lo fa trasformando il codice byte di Java 8 compilato in modo che possa essere eseguito su un runtime Java precedente.

Funzionalità della lingua backport:

- Le espressioni lambda sono supportate convertendole in classi interne anonime. Ciò include l'ottimizzazione dell'utilizzo di un'istanza singleton per espressioni lambda stateless per evitare l'allocazione ripetuta degli oggetti. I riferimenti al metodo sono fondamentalmente solo lo zucchero di sintassi per le espressioni lambda e vengono backportati allo stesso modo.

- Le istruzioni Try-with-resources vengono `Throwable.addSuppressed` rimuovendo le chiamate a `Throwable.addSuppressed` se la versione bytecode di destinazione è inferiore a Java 7. Se si desidera che le eccezioni soppresse vengano registrate anziché ingerite, si prega di creare una richiesta di funzionalità e la faremo configurabile.

- `Objects.requireNonNull` chiamate `Objects.requireNonNull` vengono sostituite con chiamate a `Object.getClass` se la versione bytecode di destinazione è inferiore a Java 7. I controlli Null sintetici generati da JDK 9 utilizzano `Objects.requireNonNull` , mentre le versioni precedenti di JDK utilizzavano `Object.getClass` .

- Opzionalmente anche:

    1. I metodi predefiniti vengono sottoposti a backport copiando i metodi predefiniti in una classe companion (nome interfaccia + "$") come metodi statici, sostituendo i metodi predefiniti nell'interfaccia con metodi astratti e aggiungendo le implementazioni del metodo necessarie a tutte le classi che implementano tale interfaccia .

    2. I metodi statici sulle interfacce vengono sottoposti a backport spostando i metodi statici in una classe companion (nome dell'interfaccia + "$") e cambiando tutte le chiamate di metodi per chiamare il nuovo percorso del metodo.

Limiti noti:

- Non esegue il backport delle API Java 8
- I metodi di backporting predefiniti e i metodi statici sulle interfacce richiedono tutte le interfacce backported e tutte le classi che li implementano o richiamano i loro metodi statici per il backporting insieme, con un'esecuzione di Retrolambda. In altre parole, devi sempre fare una build pulita. Inoltre, i metodi predefiniti per il backport non funzioneranno tra i limiti del modulo o delle dipendenze.
- Può interrompersi se un futuro build JDK 8 interrompe la generazione di una nuova classe per ogni chiamata ad `invokedynamic` . Retrolambda funziona in modo da catturare il bytecode generato da `java.lang.invoke.LambdaMetafactory` modo dinamico, quindi le ottimizzazioni a tale meccanismo potrebbero interrompere Retrolambda.

Retrolambda gradle plugin creerà automaticamente il tuo progetto Android con Retrolambda. L'ultima versione può essere trovata nella pagina dei rilasci .

Uso:

1. Scarica e installa jdk8
2. Aggiungi quanto segue al tuo `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
    }
}
```

Problemi noti:

- Lint non funziona sui file java con lambda. La lanugine di Android non capisce la sintassi di java 8 e fallirà silenziosamente o ad alta voce. Ora c'è un fork sperimentale che risolve il problema.

- L'utilizzo di Google Play Services causa il fallimento di Retrolambda. La versione 5.0.77 contiene il bytecode incompatibile con Retrolambda. Questo dovrebbe essere risolto nelle versioni più recenti dei servizi di gioco, se è possibile aggiornare, questa dovrebbe essere la soluzione preferita. Per ovviare a questo problema, è possibile utilizzare una versione precedente come 4.4.52 o aggiungere `-noverify` agli `-noverify` di jvm.

```
retrolambda {
  jvmArgs '-noverify'
}
```

Leggi Java su Android online: https://riptutorial.com/it/android/topic/9223/java-su-android

Capitolo 139: JCodec

Examples

Iniziare

È possibile ottenere automaticamente JCodec con Maven. Per questo basta aggiungere sotto snippet al tuo pom.xml.

```
<dependency>
    <groupId>org.jcodec</groupId>
    <artifactId>jcodec-javase</artifactId>
    <version>0.1.9</version>
</dependency>
```

Ottenere frame dal film

Ottenere un singolo fotogramma da un film (supporta solo AVC, H.264 in MP4, ISO BMF, contenitore Quicktime):

```
int frameNumber = 150;
BufferedImage frame = FrameGrab.getFrame(new File("filename.mp4"), frameNumber);
ImageIO.write(frame, "png", new File("frame_150.png"));
```

Ottenere una sequenza di fotogrammi da un film (supporta solo AVC, H.264 in MP4, ISO BMF, contenitore 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);
}
```

Leggi JCodec online: https://riptutorial.com/it/android/topic/9948/jcodec

Capitolo 140: JSON in Android con org.json

Sintassi

- **Oggetto** : un oggetto è un insieme non ordinato di coppie nome / valore. Un oggetto inizia con {(parentesi sinistra) e termina con} (parentesi graffa destra). Ogni nome è seguito da: (due punti) e le coppie nome / valore sono separate da, (virgola).
- **Array** : un array è una raccolta di valori ordinata. Un array inizia con [(parentesi quadra sinistra) e termina con] (parentesi quadra destra). I valori sono separati da, (virgola).
- **Valore** : un valore può essere una stringa tra virgolette doppie, un numero o vero o falso o null o un oggetto o un array. Queste strutture possono essere annidate.
- **Stringa** : una stringa è una sequenza di zero o più caratteri Unicode, racchiusa tra virgolette doppie, utilizzando gli escape di backslash. Un personaggio è rappresentato come una singola stringa di caratteri. Una stringa è molto simile a una stringa C o Java.
- **Numero** : Un numero è molto simile a un numero C o Java, tranne per il fatto che i formati ottale ed esadecimale non sono usati.

Osservazioni

Questo argomento riguarda l'utilizzo del pacchetto `org.json` incluso nell'SDK di Android.

Examples

Analizza semplici oggetti JSON

Considera la seguente stringa JSON:

```
{
  "title": "test",
  "content": "Hello World!!!",
  "year": 2016,
  "names" : [
        "Hannah",
        "David",
        "Steve"
   ]
}
```

Questo oggetto JSON può essere analizzato usando il seguente codice:

```
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());
}
```

Ecco un altro esempio con un JSONArray nidificato dentro JSONObject:

```
{
    "books":[
      {
        "title":"Android JSON Parsing",
        "times_sold":186
      }
    ]
}
```

Questo può essere analizzato con il seguente codice:

```
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");
```

Creazione di un oggetto JSON semplice

Creare il `JSONObject` usando il costruttore vuoto e aggiungere campi usando il metodo `put()` , che è sovraccarico in modo che possa essere usato con tipi diversi:

```
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 stringa `JSON` risultante assomiglia a questa:

```
{
   "name":"test",
   "content":"Hello World!!!1",
```

```
   "year":2016,
   "value":3.23,
   "member":true,
   "null_value":null
}
```

Aggiungi 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 stringa JSON risultante assomiglia a questa:

```
{
   "the_array":[
      "ASDF",
      "QWERTY"
   ]
}
```

Creare una stringa JSON con valore null.

Se è necessario produrre una stringa JSON con un valore `null` come questo:

```
{
   "name":null
}
```

Quindi devi usare la costante speciale JSONObject.NULL .

Esempio di funzionamento:

```
jsonObject.put("name", JSONObject.NULL);
```

Lavorando con null-string durante l'analisi di json

```
{
    "some_string": null,
    "ather_string": "something"
}
```

Se useremo in questo modo:

```
JSONObject json = new JSONObject(jsonStr);
String someString = json.optString("some_string");
```

Avremo output:

```
someString = "null";
```

Quindi dobbiamo fornire questa soluzione alternativa:

```
/**
 * 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);
    }
}
```

E poi chiama:

```
JSONObject json = new JSONObject(jsonStr);
String someString = optNullableString(json, "some_string");
String someString2 = optNullableString(json, "some_string", "");
```

E avremo Output come ci aspettavamo:

```
someString = null; //not "null"
someString2 = "";
```

Utilizzo di JsonReader per leggere JSON da un flusso

`JsonReader`

legge un valore codificato JSON come flusso di token.

```
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);
  }
```

Crea un oggetto JSON nidificato

Per produrre un oggetto JSON annidato, devi semplicemente aggiungere un oggetto JSON a un altro:

```
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";
}
```

Ora `mainObject` contiene una chiave chiamata `claim` con l'intero `requestObject` come valore.

Gestione della chiave dinamica per la risposta JSON

Questo è un esempio di come gestire la chiave dinamica per la risposta. Qui `A` e `B` sono chiavi dinamiche e possono essere qualsiasi cosa

Risposta

```
{
  "response": [
    {
      "A": [
        {
          "name": "Tango"
        },
        {
          "name": "Ping"
```

```
        }
      ],
      "B": [
        {
          "name": "Jon"
        },
        {
          "name": "Mark"
        }
      ]
    }
  ]
}
```

Codice 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);
    }
}
```

Verifica la presenza di campi su JSON

A volte è utile verificare se un campo è presente o assente sul tuo JSON per evitare qualche `JSONException` sul tuo codice.

Per `JSONObject#has(String)` , usa `JSONObject#has(String)` o il metodo, come nell'esempio seguente:

Esempio JSON

```
{
```

```
   "name":"James"
}
```

Codice 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".
```

Aggiornamento degli elementi nel JSON

esempio json da aggiornare

```
 {
 "student":{"name":"Rahul", "lastname":"sharma"},
 "marks":{"maths":"88"}
 }
```

Per aggiornare il valore degli elementi in JSON dobbiamo assegnare il valore e l'aggiornamento.

```
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);
}
```

valore aggiornato

```
{
"student":{"name":"Kumar", "lastname":"sharma"},
"marks":{"maths":"88"}
}
```

Leggi JSON in Android con org.json online: https://riptutorial.com/it/android/topic/106/json-in-android-con-org-json

Capitolo 141: layout

introduzione

Un layout definisce la struttura visiva per un'interfaccia utente, come un'attività o un widget.

Un layout è dichiarato in XML, inclusi gli elementi dello schermo che appariranno in esso. Il codice può essere aggiunto all'applicazione per modificare lo stato degli oggetti dello schermo in fase di runtime, compresi quelli dichiarati in XML.

Sintassi

- Android: la gravità = "top | bottom | left | right | center_vertical | fill_vertical | center_horizontal | fill_horizontal | centro | riempire | clip_vertical | clip_horizontal | Inizio | fine"
- Android: layout_gravity = "top | bottom | left | right | center_vertical | fill_vertical | center_horizontal | fill_horizontal | centro | riempire | clip_vertical | clip_horizontal | Inizio | fine"

Osservazioni

LayoutParams e 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

richiede due passaggi di layout per il rendering corretto. Per le gerarchie di viste complesse, ciò può avere un impatto significativo sulle prestazioni. Nesting `RelativeLayouts` rende questo problema ancora peggiore, perché ogni `RelativeLayout` fa aumentare il numero di passaggi di layout.

Examples

LinearLayout

LinearLayout è un `ViewGroup` che organizza i suoi figli in una singola colonna o una singola riga. L'orientamento può essere impostato chiamando il metodo `setOrientation()` o usando l'attributo xml `android:orientation` .

1. **Orientamento verticale** : `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>
```

Ecco uno screenshot di come apparirà questo:

2. **Orientamento orizzontale** : `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` supporta anche l'assegnazione di un peso ai singoli bambini con l'attributo `android:layout_weight` .

RelativeLayout

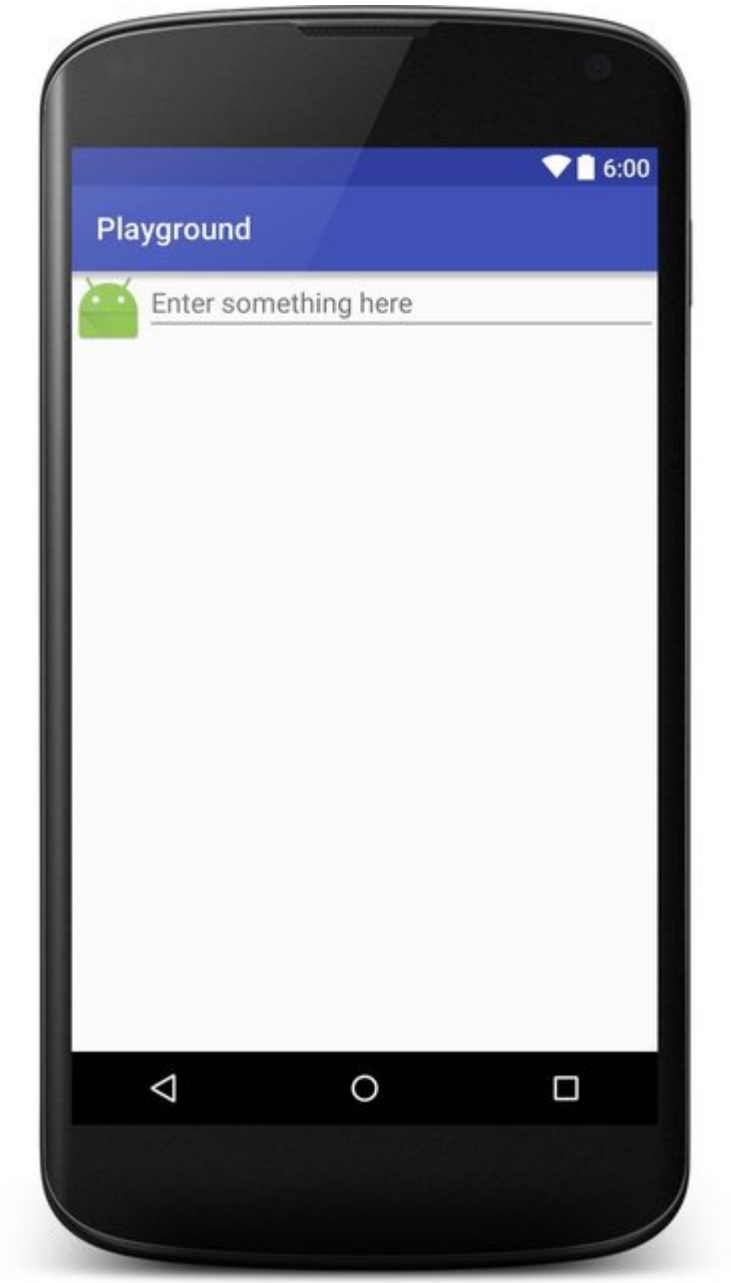
`RelativeLayout` è un `ViewGroup` che visualizza viste `ViewGroup` in posizioni relative. Per impostazione

predefinita, tutte le viste child vengono disegnate nella parte in alto a sinistra del layout, quindi è necessario definire la posizione di ciascuna vista utilizzando le varie proprietà di layout disponibili da `RelativeLayout.LayoutParams` . Il valore per ogni proprietà di layout è o un valore booleano per abilitare una posizione di layout relativa a RelativeLayout padre o un ID che fa riferimento a un'altra vista nel layout rispetto alla quale deve essere posizionata la vista.

Esempio:

```
<?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>
```

Ecco uno screenshot di come apparirà questo:



Gravità e gravità del layout

Android: layout_gravity

- `android:layout_gravity` è usato per impostare la posizione di un elemento nel suo genitore (ad esempio, una `View` un bambino all'interno di un `Layout`).
- Supportato da LinearLayout e FrameLayout

Android: la gravità

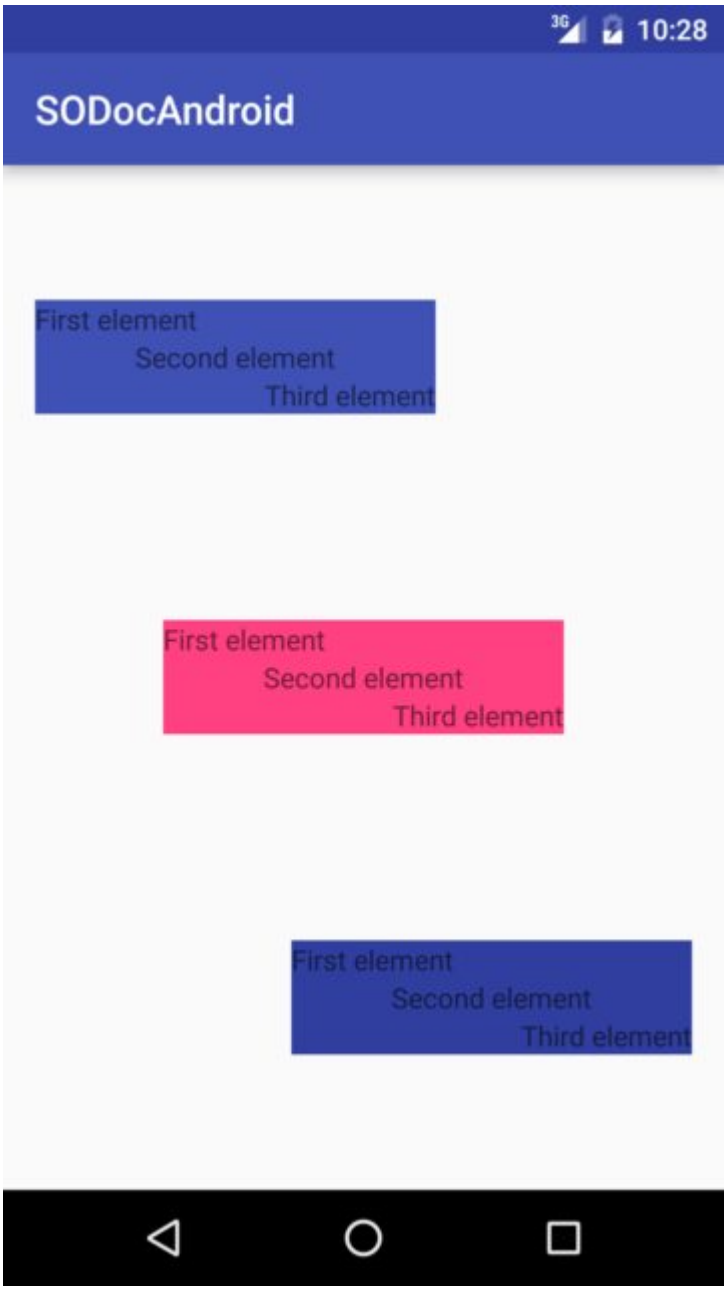
- `android:gravity` viene utilizzata per impostare la posizione del contenuto all'interno di un elemento (ad es. un testo all'interno di una `TextView`).

```
<?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>
```

Che viene reso come segue:

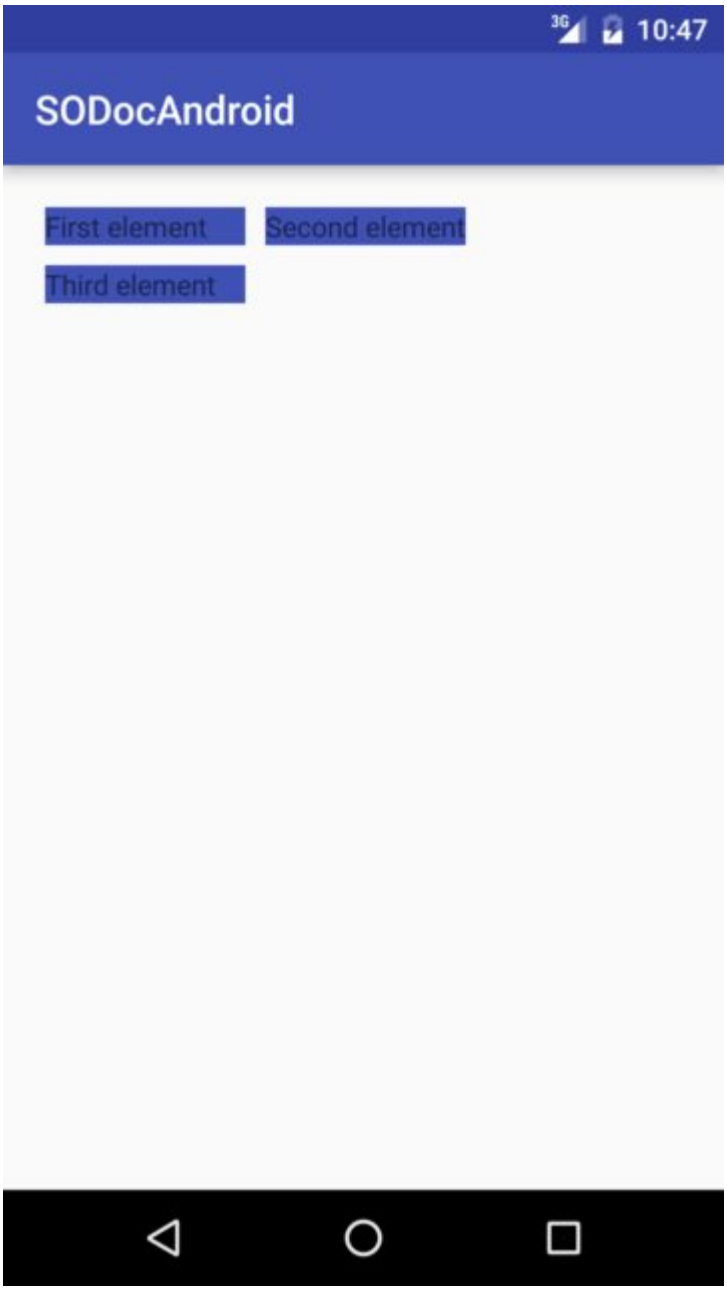


Layout della griglia

GridLayout, come suggerisce il nome, è un layout utilizzato per organizzare le viste in una griglia. Un GridLayout si divide in colonne e righe. Come puoi vedere nell'esempio seguente, la quantità di colonne e / o righe è specificata dalle proprietà `columnCount` e `rowCount` . L'aggiunta di viste a questo layout aggiungerà la prima vista alla prima colonna, la seconda vista alla seconda colonna e la terza vista alla prima colonna della seconda riga.

```
<?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>
```



Layout percentuali

2.3

La libreria di supporto percentuale fornisce `PercentFrameLayout` e `PercentRelativeLayout` , due ViewGroup che forniscono un modo semplice per specificare **dimensioni e margini di visualizzazione** in termini di **percentuale** della dimensione complessiva.

È possibile utilizzare la libreria di supporto percentuale aggiungendo quanto segue alle proprie dipendenze.

```
compile 'com.android.support:percent:25.3.1'
```

Se si desidera visualizzare una vista che riempie lo schermo orizzontalmente ma solo metà dello schermo in verticale, si farebbe in seguito.

```
<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>
```

È inoltre possibile definire le percentuali in un file XML separato con codice come:

```
<fraction name="margin_start_percent">25%</fraction>
```

E fai riferimento a loro nei tuoi layout con `@fraction/margin_start_percent` .

Contengono anche la possibilità di impostare un **rapporto aspetto** personalizzato tramite `app:layout_aspectRatio` .
Ciò ti consente di impostare solo una singola dimensione, ad esempio solo la larghezza, e l'altezza verrà determinata automaticamente in base al rapporto aspetto che hai definito, che sia 4: 3 o 16: 9 o anche un quadrato 1: 1 proporzioni.

Per esempio:

```
 <ImageView
    app:layout_widthPercent="100%"
    app:layout_aspectRatio="178%"
    android:scaleType="centerCrop"
    android:src="@drawable/header_background"/>
```

FrameLayout

`FrameLayout` è progettato per bloccare un'area sullo schermo per visualizzare un singolo oggetto. Tuttavia, puoi aggiungere più bambini a FrameLayout e controllare la loro posizione all'interno di FrameLayout assegnando la gravità a ciascun bambino, utilizzando l'attributo android: layout_gravity .

In genere, `FrameLayout` viene utilizzato per contenere una singola vista figlio. Casi di uso comune stanno creando segnaposto per gonfiare `Fragments` in `Activity` , sovrapporre viste o applicare in primo piano alle viste.

Esempio:

```
<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>
```

Sembrerà così:

CoordinatorLayout

2.3

Il `CoordinatorLayout` è un contenitore un po 'simile a `FrameLayout` ma con capacità extra, è chiamato `FrameLayout` super-potenziato nella documentazione ufficiale.

Associando un `CoordinatorLayout.Behavior` a un figlio diretto di CoordinatorLayout, sarete in grado di intercettare gli eventi tattili, le finestre, le misure, il layout e lo scorrimento annidato.

Per poterlo usare, dovrai prima aggiungere una dipendenza per la libreria di supporto nel tuo file gradle:

```
compile 'com.android.support:design:25.3.1'
```

Il numero dell'ultima versione della libreria può essere trovato qui

Un caso pratico di utilizzo di `CoordinatorLayout` è la creazione di una vista con un `FloatingActionButton` . In questo caso specifico, creeremo un `RecyclerView` con `SwipeRefreshLayout` e `FloatingActionButton` . Ecco come puoi farlo:

```
<?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>
```

Nota come FloatingActionButton è ancorato al CoordinatorLayout con
`app:layout_anchor="@id/coord_layout"`

CoordinatorLayout Comportamento di scorrimento

2.3-2.3.2

È possibile utilizzare un `CoordinatorLayout` consente di ottenere effetti di scorrimento del disegno del materiale quando si utilizzano layout interni che supportano lo scorrimento annidato, come `NestedScrollView` o `RecyclerView` .

Per questo esempio:

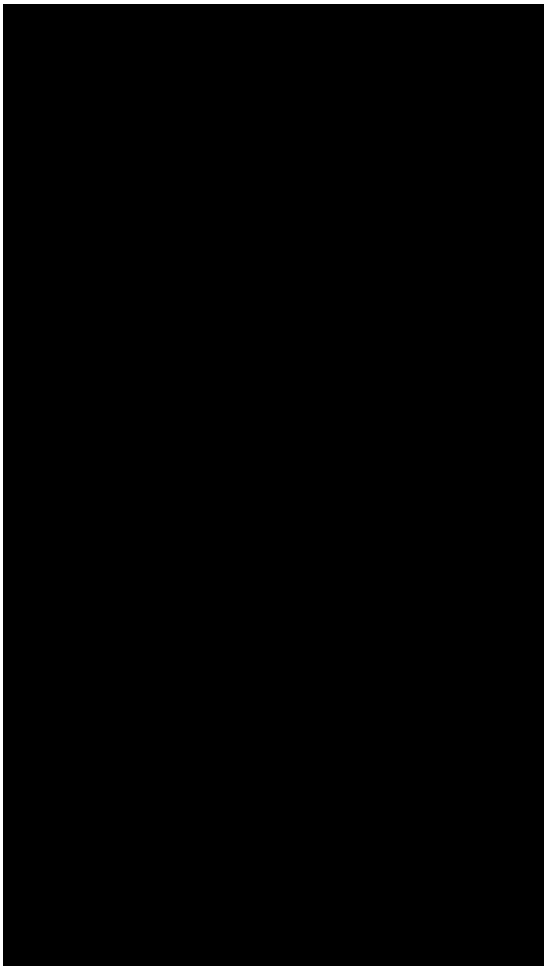
- `app:layout_scrollFlags="scroll|enterAlways"` viene utilizzato nelle proprietà della barra degli strumenti
- `app:layout_behavior="@string/appbar_scrolling_view_behavior"` viene utilizzato nelle proprietà ViewPager
- Un RecyclerView viene utilizzato nei frammenti di ViewPager

Ecco il file xml di layout utilizzato in un'attività:

```
<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>
```

Risultato:



Visualizza peso

Uno degli attributi più utilizzati per LinearLayout è il peso delle sue viste secondarie . Peso definisce quanto spazio occuperà una vista rispetto ad altre viste all'interno di un LinearLayout.

Il peso viene utilizzato quando si desidera assegnare uno spazio dello schermo specifico a un componente rispetto all'altro.

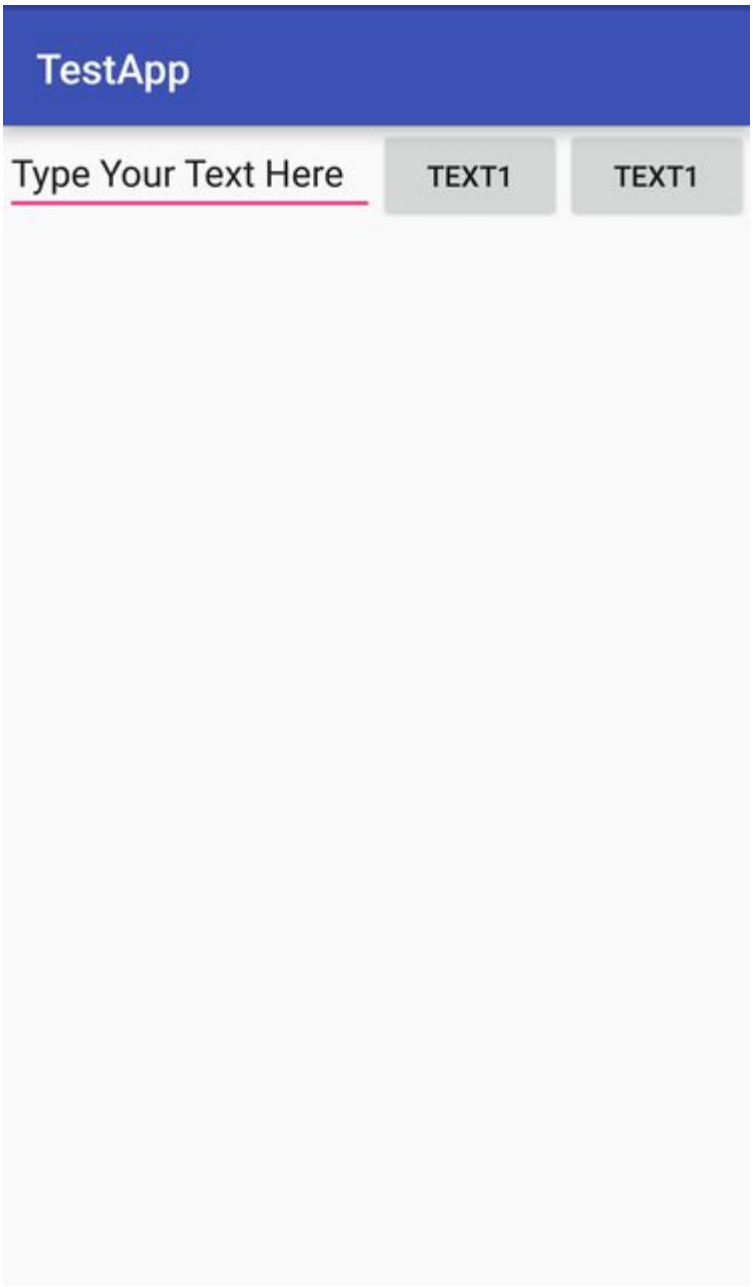
Proprietà chiave :

- `weightSum` è la somma complessiva dei pesi di tutte le visualizzazioni figlio. Se non si specifica `weightSum` , il sistema calcolerà da solo la somma di tutti i pesi.
- `layout_weight` specifica la quantità di spazio fuori dalla somma di peso totale occupata dal widget.

Codice:

```
<?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>
```

L'output è:



Ora, anche se la dimensione del dispositivo è maggiore, EditText richiederà 2/4 dello spazio dello schermo. Quindi l'aspetto della tua app è visto coerente su tutti gli schermi.

Nota: qui `layout_width` viene mantenuto `0dp` poiché lo spazio del widget è diviso orizzontalmente. Se i widget devono essere allineati verticalmente, `layout_height` sarà impostato su `0dp` . Questo viene fatto per aumentare l'efficienza del codice, perché durante il runtime il sistema non tenterà di calcolare rispettivamente la larghezza o l'altezza in quanto viene gestita dal peso. Se invece si utilizza `wrap_content` il sistema `wrap_content` di calcolare prima la larghezza / altezza prima di applicare l'attributo weight che causa un altro ciclo di calcolo.

Creazione di LinearLayout a livello di codice

Gerarchia

```
- LinearLayout(horizontal)
   - ImageView
```

```
  - LinearLayout(vertical)
    - TextView
    - TextView
```

Codice

```
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

Ogni singolo `ViewGroup` (ad es. `LinearLayout` , `RelativeLayout` , `CoordinatorLayout` , ecc.) `LinearLayout` memorizzare le informazioni sulle proprietà dei propri figli. Riguardo al modo in cui i suoi figli sono disposti nel `ViewGroup` . Queste informazioni sono memorizzate in oggetti di una classe wrapper `ViewGroup.LayoutParams` .

Per includere parametri specifici per un particolare tipo di layout, i `ViewGroups` utilizzano sottoclassi della classe `ViewGroup.LayoutParams` .

Ad esempio per

- `LinearLayout` è `LinearLayout.LayoutParams`
- `RelativeLayout` è `RelativeLayout.LayoutParams`
- `CoordinatorLayout` è `CoordinatorLayout.LayoutParams`
- ...

La maggior parte dei `ViewGroups` riutilizza la possibilità di impostare i `margins` per i propri figli, in modo che non sottoclassi direttamente `ViewGroup.LayoutParams` , ma eseguono la sottoclasse di `ViewGroup.MarginLayoutParams` (che a sua volta è una sottoclasse di `ViewGroup.LayoutParams`).

`LayoutParams` in xml

`LayoutParams`

oggetti `LayoutParams` vengono creati in base al file `xml` layout gonfiato.

```
<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>
```

Tutti i parametri che iniziano con `layout_` specificano come il layout **racchiude** dovrebbe funzionare. Quando il layout viene gonfiato, questi parametri vengono avvolti in un oggetto `LayoutParams` appropriato, che in seguito verrà utilizzato dal `Layout` per posizionare correttamente una particolare `View` all'interno del `ViewGroup` . Altri attributi di una `View` sono direttamente correlati alla `View` e vengono elaborati dalla `View` stessa.

Per `TextView` :

- `layout_width` , `layout_height` e `layout_gravity` verranno archiviati in un oggetto `LinearLayout.LayoutParams` e utilizzati da `LinearLayout`
- `gravity` , `text` e `textColor` saranno utilizzati da `TextView` stesso

Per `ImageView` :

- `layout_width` , `layout_height` e `layout_weight` verranno archiviati in un oggetto `LinearLayout.LayoutParams` e utilizzati da `LinearLayout`
- `background` , `scaleType` e `src` saranno utilizzati da `ImageView` stesso

Ottenere oggetto `LayoutParams`

`getLayoutParams` è una `View's` metodo che permette di recuperare una corrente `LayoutParams` oggetto.

Poiché l'oggetto `LayoutParams` è direttamente correlato al `ViewGroup` **che** lo **racchiude** , questo metodo restituirà un valore non null solo quando `View` è collegato al `ViewGroup` . È necessario tenere a mente che questo oggetto potrebbe non essere presente in ogni momento. Soprattutto non dovresti dipendere dal fatto di averlo nel costruttore `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();
        }
    }

    //...
}
```

Se vuoi dipendere dall'avere oggetto `LayoutParams` , dovresti usare invece il metodo `onAttachedToWindow` .

```
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();
        }
    }

    //...
}
```

Oggetto Casting `LayoutParams`

Potrebbe essere necessario utilizzare funzionalità specifiche per un particolare `ViewGroup` (ad

esempio, potrebbe essere necessario modificare a livello di `ViewGroup` regole di un `RelativeLayout`). A tale scopo è necessario sapere come eseguire correttamente il cast dell'oggetto `ViewGroup.LayoutParams` .

Questo potrebbe essere un po 'di confusione quando si ottiene un oggetto `LayoutParams` per un figlio `View` che in realtà è un altro `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: il tipo di oggetto `LayoutParams` è direttamente correlato al tipo del **ViewGroup** `ViewGroup` .

Casting ***errato*** :

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
FrameLayout.LayoutParams par = (FrameLayout.LayoutParams) innerLayout.getLayoutParams();
                                        // INCORRECT! This will produce ClassCastException
```

Cast corretta :

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
LinearLayout.LayoutParams par = (LinearLayout.LayoutParams) innerLayout.getLayoutParams();
                                       // CORRECT! the enclosing layout is a LinearLayout
```

Leggi layout online: https://riptutorial.com/it/android/topic/94/layout

Capitolo 142: Leakcanary

introduzione

Leak Canary è una libreria Android e Java utilizzata per rilevare perdite nell'applicazione

Osservazioni

Puoi vedere l'esempio nel link sottostante

https://github.com/square/leakcanary

Examples

Implementazione di un Leak Canary nell'applicazione Android

Nel tuo *build.gradle* devi aggiungere le seguenti dipendenze:

```
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'
```

Nella tua classe di `Application` devi aggiungere il seguente codice all'interno di `onCreate()` :

```
LeakCanary.install(this);
```

Questo è tutto ciò che devi fare per *LeakCanary* , mostrerà automaticamente le notifiche quando c'è una perdita nella tua build.

Leggi Leakcanary online: https://riptutorial.com/it/android/topic/10041/leakcanary

Capitolo 143: Lettura codice a barre e QR code

Osservazioni

QRCodeReaderView

ZXing

Examples

Utilizzo di QRCodeReaderView (basato su Zxing)

QRCodeReaderView implementa una vista Android che mostra la telecamera e avvisa quando c'è un codice QR all'interno dell'anteprima.

Utilizza la libreria di elaborazione di immagini di codici a barre 1D / 2D multi-formato e open source zxing .

Aggiungere la libreria al tuo progetto

Aggiungi dipendenza QRCodeReaderView al tuo build.gradle

```
dependencies{
    compile 'com.dlazaro66.qrcodereaderview:qrcodereaderview:2.0.0'
}
```

Primo utilizzo

- Aggiungi al tuo layout un `QRCodeReaderView`

```
<com.dlazaro66.qrcodereaderview.QRCodeReaderView
       android:id="@+id/qrdecoderview"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />
```

- Creare un'attività che implementa `onQRCodeReadListener` e utilizzarla come listener di `QrCodeReaderView` .
- Assicurati di disporre delle autorizzazioni della videocamera per poter utilizzare la libreria. (https://developer.android.com/training/permissions/requesting.html)

Quindi, nella tua attività, puoi utilizzarlo come segue:

```
    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();
    }
}
```

Leggi Lettura codice a barre e QR code online: https://riptutorial.com/it/android/topic/6067/lettura-codice-a-barre-e-qr-code

Capitolo 144: Library Dagger 2: Iniezione delle dipendenze nelle applicazioni

introduzione

Dagger 2, come spiegato su GitHub , è un approccio evolutivo in fase di compilazione all'iniezione di dipendenza. Prendendo l'approccio iniziato in Dagger 1.x fino alla sua conclusione definitiva, Dagger 2.x elimina tutto il riflesso e migliora la chiarezza del codice rimuovendo il tradizionale `ObjectGraph` / `Injector` a favore delle interfacce `@Component` specificate `@Component` .

Osservazioni

1. Configurazione della libreria in applicazione (per progetti maven, gradle, java)
2. Vantaggi dell'uso di Dragger
3. Link importanti (per documentazione e demo)
4. Come integrare e utilizzare i componenti di Dragger

API di Dagger 2:

Dagger 2 espone un numero di annotazioni speciali:

@Module per le classi i cui metodi forniscono dipendenze

@ **Fornisce** i metodi all'interno delle classi @Module

@Inject per richiedere una dipendenza (un costruttore, un campo o un metodo)

@Component è un'interfaccia bridge tra i moduli e l'iniezione

Link importanti:

GitHub: https://github.com/google/dagger

UserGuide (Google): https://google.github.io/dagger/users-guide.html

Video: https://google.github.io/dagger/resources.html

Vogella Tutorial: http://www.vogella.com/tutorials/Dagger/article.html

Tutorial Codepath: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Examples

Creare la classe @Module e l'annotazione @Singleton per Object

```
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());
    }
}
```

Ogni provider (o metodo) deve avere l'annotazione `@Provides` e la classe deve avere l'annotazione `@Module` . L'annotazione `@Singleton` indica che ci sarà una sola istanza dell'oggetto.

Richiedi dipendenze in oggetti dipendenti

Ora che hai i provider per i tuoi diversi modelli, devi richiederli. Proprio come `Vehicle` bisogno di `Motor` , devi aggiungere l'annotazione `@Inject` nel costruttore `Vehicle` come segue:

```
@Inject
public Vehicle(Motor motor){
    this.motor = motor;
}
```

È possibile utilizzare l'annotazione `@Inject` per richiedere dipendenze nel costruttore, nei campi o nei metodi. In questo esempio, sto mantenendo l'iniezione nel costruttore.

Connettere @Modules con @Inject

La connessione tra il fornitore di dipendenze, `@Module` e le classi che li richiedono tramite `@Inject` viene effettuata usando `@Component` , che è un'interfaccia:

```
import javax.inject.Singleton;
import dagger.Component;

@Singleton
@Component(modules = {VehicleModule.class})
public interface VehicleComponent {
    Vehicle provideVehicle();
}
```

Per l'annotazione `@Component` , devi specificare quali moduli saranno utilizzati. In questo esempio viene utilizzato `VehicleModule` , che è definito in questo esempio . Se hai bisogno di usare più moduli, aggiungili semplicemente usando una virgola come separatore.

Utilizzo dell'interfaccia @Component per ottenere oggetti

Ora che hai pronta ogni connessione, devi ottenere un'istanza di questa interfaccia e invocare i suoi metodi per ottenere l'oggetto di cui hai bisogno:

```
VehicleComponent component = Dagger_VehicleComponent.builder().vehicleModule(new
VehicleModule()).build();
vehicle = component.provideVehicle();
Toast.makeText(this, String.valueOf(vehicle.getSpeed()), Toast.LENGTH_SHORT).show();
```

Quando si tenta di creare un nuovo oggetto dell'interfaccia con l'annotazione `@Component` , è necessario farlo utilizzando il prefisso `Dagger_<NameOfTheComponentInterface>` , in questo caso `Dagger_VehicleComponent` , e quindi utilizzare il metodo builder per chiamare ogni modulo all'interno.

Leggi Library Dagger 2: Iniezione delle dipendenze nelle applicazioni online: https://riptutorial.com/it/android/topic/9079/library-dagger-2--iniezione-delle-dipendenze-nelle-applicazioni

Capitolo 145: Libreria di associazione dati

Osservazioni

Impostare

Prima di utilizzare l'associazione dati, è necessario abilitare il plug-in in `build.gradle` .

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

Nota: il collegamento dati è stato aggiunto al plug-in Android Gradle nella versione 1.5.0

Nomi delle classi vincolanti

Il plug-in di associazione dei dati genera un nome di classe di associazione convertendo il nome del file del layout in caso Pascal e aggiungendo "Binding" alla fine. Quindi `item_detail_activity.xml` genererà una classe chiamata `ItemDetailActivityBinding` .

risorse

- Documentazione ufficiale

Examples

Binding del campo di testo di base

Gradle (Module: app) Configuration

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

Modello di dati

```
public class Item {
    public String name;
    public String description;

    public Item(String name, String description) {
        this.name = name;
        this.description = description;
```

```
    }
}
```

Layout XML

Il primo passo è il wrapping del layout in un tag `<layout>` , aggiungendo un elemento `<data>` e aggiungendo un elemento `<variable>` per il tuo modello dati.

Quindi è possibile associare gli attributi XML ai campi nel modello dati utilizzando `@{model.fieldname}` , dove `model` è il nome della variabile e `fieldname` è il campo a cui si desidera accedere.

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>
```

Per ogni file di layout XML configurato correttamente con i binding, il plug-in Android Gradle genera una classe corrispondente: binding. Poiché abbiamo un layout denominato *item_detail_activity* , la corrispondente classe di associazione generata è chiamata `ItemDetailActivityBinding` .

Questo legame può quindi essere utilizzato in un'attività come questa:

```
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);
    }
}
```

Associazione con un metodo di accesso

Se il tuo modello ha metodi privati, la libreria di associazione ti consente comunque di accedervi nella tua vista senza utilizzare il nome completo del metodo.

Modello di dati

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Layout 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>
```

Classi di riferimento

Modello di dati

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Layout XML

È necessario importare le classi di riferimento, proprio come faresti in 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: il pacchetto `java.lang.*` Viene importato automaticamente dal sistema. (Lo stesso è fatto da `JVM` per `Java`)

Databinding in Fragment

Modello di dati

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name){
        this.name = name;
    }

}
```

Layout 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>
```

Frammento

```
@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();
}
```

Associazione dati bidirezionale integrata

Il collegamento dati bidirezionale supporta i seguenti attributi:

Elemento	Proprietà
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	Proprietà
Switch	android:checked

uso

```
<layout ...>
    <data>
        <variable type="com.example.myapp.User" name="user"/>
    </data>
    <RelativeLayout ...>
        <EditText android:text="@={user.firstName}" .../>
    </RelativeLayout>
</layout>
```

Si noti che l'espressione Binding `@={}` **ha un valore aggiuntivo** `=` , che è necessario per il **Binding a due vie** . Non è possibile utilizzare i metodi nelle espressioni Binding bidirezionali.

Associazione dati in Adattatore RecyclerView

È anche possibile utilizzare l'associazione dati all'interno del tuo Adattatore `RecyclerView` .

Modello di dati

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Layout XML

```
<TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@{item.name}"/>
```

Classe dell'adattatore

```
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();
        }
    }
}
```

Clicca listener con Binding

Crea un'interfaccia per clickHandler

```
public interface ClickHandler {
    public void onButtonClick(View v);
}
```

Layout 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>
```

Gestisci l'evento nella tua attività

```
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 personalizzato usando l'espressione lambda

Definisci interfaccia

```
public interface ClickHandler {
    public void onButtonClick(User user);
}
```

Crea una classe del modello

```
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;
    }
}
```

Layout 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>
```

Codice attività:

```
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();
    }
}
```

Per alcuni listener di visualizzazione che non sono disponibili nel codice xml ma possono essere impostati in codice java, possono essere associati con l'associazione personalizzata.

Classe personalizzata

```
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);
    }
}
```

Classe dell'handler

```
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>
```

Valore predefinito in Associazione dati

> Il riquadro Anteprima mostra i valori predefiniti per le espressioni di associazione dati, se fornite.

Per esempio :

```
android:layout_height="@{@dimen/main_layout_height, default=wrap_content}"
```

`wrap_content` progettazione occorrerà `wrap_content` e agirà come `wrap_content` nel riquadro di anteprima.

Un altro esempio è

```
android:text="@{user.name, default=`Preview Text`}"
```

`Preview Text` di anteprima nel riquadro di anteprima ma, quando lo si esegue in dispositivo / emulatore, verrà visualizzato il testo effettivo associato ad esso

DataBinding con variabili personalizzate (int, booleano)

A volte abbiamo bisogno di eseguire operazioni di base come nascondere / mostrare la vista in base al singolo valore, per quella singola variabile non possiamo creare il modello o non è una buona pratica creare un modello per quello. DataBinding supporta i tipi di dati di base per eseguire tali oprazioni.

```
<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>
```

e imposta il suo valore dalla classe java.

```
binding.setSelected(true);
```

Databinding in Dialog

```
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();
}
```

Passa il widget come riferimento in BindingAdapter

Layout 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>
```

Metodo 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);
}
```

Leggi Libreria di associazione dati online: https://riptutorial.com/it/android/topic/111/libreria-di-associazione-dati

Capitolo 146: Localizzazione con risorse in Android

Examples

Moneta

```
Currency currency = Currency.getInstance("USD");
NumberFormat format = NumberFormat.getCurrencyInstance();
format.setCurrency(currency);
format.format(10.00);
```

Aggiunta di traduzione alla tua app Android

Devi creare un file `strings.xml` diverso per ogni nuova lingua.

1. Fare clic con il tasto destro sulla cartella *res*
2. Scegli *Nuovo* → *File delle risorse dei valori*
3. Seleziona una locale dai qualificatori disponibili
4. Fare clic sul pulsante *Avanti* (>>)
5. Seleziona una lingua
6. Denominare il file *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>
```

Impostazione della lingua in modo programmatico:

```
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 funzione sopra cambierà i campi di testo a cui si fa riferimento da *strings.xml* . Ad esempio, supponi di avere le seguenti due visualizzazioni di testo:

```
<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"/>
```

Quindi, dopo aver modificato le `app_name` internazionali, le stringhe della lingua con gli ids `app_name` e `hello` verranno modificate di conseguenza.

Tipo di directory di risorse nella cartella "res"

Quando è necessario localizzare diversi tipi di risorse, ognuna delle quali ha la propria casa nella struttura del progetto Android. Di seguito sono riportate le diverse directory che possiamo inserire nella directory `\res` . I tipi di risorse inseriti in ciascuna di queste directory sono spiegati nella tabella seguente:

elenco	Tipo di risorsa
animatore/	File XML che definiscono le animazioni delle proprietà.
Anim /	File XML che definiscono le animazioni interpolate. (Le animazioni delle proprietà possono anche essere salvate in questa directory, ma la cartella animatore / è preferibile per le animazioni delle proprietà per distinguere tra i due tipi.)
colore/	File XML che definiscono una lista di stati di colori. Vedere la risorsa Elenco stato colore
drawable /	"File bitmap (.png, .9.png, .jpg, .gif) o file XML che sono compilati nei seguenti sottotipi di risorse estraibili:: `Bitmap files - Nine-Patches (re-sizable bitmaps) - State lists - Shapes - Animation drawables - Other drawables - "`
mipmap /	File disegnabili per diverse densità di icone di avvio. Per ulteriori informazioni sulla gestione delle icone di avvio con mipmap / cartelle, vedere Gestione dei

elenco	Tipo di risorsa
	progetti Panoramica.
disposizione/	File XML che definiscono un layout dell'interfaccia utente. Vedi la risorsa di layout.
menu/	File XML che definiscono i menu delle applicazioni, ad esempio un menu Opzioni, un menu contestuale o un sottomenu. Vedi la risorsa del menu.
crudo/	File arbitrari da salvare nella loro forma originale. Per aprire queste risorse con un InputStream non elaborato, chiamare Resources.openRawResource () con l'ID risorsa, che è R.raw.filename.
	Tuttavia, se è necessario accedere ai nomi dei file e alla gerarchia dei file originali, è possibile prendere in considerazione il salvataggio di alcune risorse nella directory assets / (anziché inres / raw /). File nelle risorse / non viene fornito un ID risorsa, quindi è possibile leggerli solo tramite AssetManager.
valori/	File XML che contengono valori semplici, come stringhe, numeri interi e colori, nonché stili e temi
xml /	File XML arbitrari che possono essere letti in fase di esecuzione chiamando Resources.getXML (). Qui è necessario salvare diversi file di configurazione XML, come una configurazione ricercabile.

Tipi di configurazione e nomi dei qualificatori per ciascuna cartella nella directory "res"

Ogni directory di risorse nella cartella `res` (elencata nell'esempio sopra) può avere diverse varianti delle risorse contenute in una directory con nome simile, con suffissi diversi `qualifier-values` per ogni `configuration-type` .

Esempio di variazioni della directory `` con diversi valori di qualificatore suffissi che sono spesso visti nei nostri progetti Android:

- drawable /
- drawable-it /
- drawable-fr-RCA /
- drawable-en-port /
- drawable-en-NoTouch-12key /
- drawable-port-ldpi /
- drawable-port-NoTouch-12key /

Elenco esaustivo di tutti i diversi tipi di configurazione e dei loro valori di qualificatore per le risorse Android:

Configurazione	Valori del Qualificatore
MCC e MNC	Esempi:
	mcc310
	mcc310-mnc004
	mcc208-mnc00
	eccetera.
Lingua e regione	Esempi:
	it
	fr
	en-rus
	fr-RFR
	fr-rCA
Direzione del layout	ldrtl
	ldltr
smallestWidth	swdp
	Esempi:
	sw320dp
	sw600dp
	sw720dp
Larghezza disponibile	WDP
	w720dp
	w1024dp
Altezza disponibile	hdp
	h720dp
	h1024dp
Dimensione dello schermo	piccolo

Configurazione	Valori del Qualificatore
	normale
	grande
	XLarge
Aspetto dello schermo	lungo
	notlong
Schermo rotondo	il giro
	notround
Orientamento schermo	porta
	terra
Modalità dell'interfaccia utente	auto
	scrivania
	televisione
	ApplianceWatch
Modalità notturna	notte
	notnight
Densità pixel dello schermo (dpi)	ldpi
	MDPI
	hdpi
	xhdpi
	xxhdpi
	xxxhdpi
	nodpi
	tvdpi
	anydpi
Tipo di touchscreen	non toccare

Configurazione	Valori del Qualificatore
	dito
Disponibilità della tastiera	keysexposed
	keyshidden
	keyssoft
Metodo di immissione del testo primario	nokeys
	QWERTY
	12key
Disponibilità della chiave di navigazione	navexposed
	navhidden
Metodo di navigazione non tattile principale	nonav
	DPAD
	trackball
	ruota
Versione piattaforma (livello API)	Esempi:
	v3
	v4
	v7

Cambia la localizzazione dell'applicazione Android programmaticamente

Negli esempi precedenti capisci come localizzare le risorse dell'applicazione. L'esempio seguente spiega come modificare la locale dell'applicazione all'interno dell'applicazione, non dal dispositivo. Per modificare solo le impostazioni locali dell'applicazione, è possibile utilizzare l'utilità di localizzazione sottostante.

```
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();
    }
}
```

Inizializza le impostazioni locali preferite dall'utente, dalla classe Application.

```
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());
    }
}
```

È inoltre necessario creare un'attività di base ed estendere questa attività a tutte le altre attività in modo che sia possibile modificare le impostazioni internazionali dell'applicazione solo in un punto come segue:

```
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: inizializzare sempre la locale nel costruttore.

Ora puoi usare LocalizationActivity come segue.

```
public class MainActivity extends LocalizationActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
    }
}
```

> Nota: quando si modifica la localizzazione dell'applicazione a livello di programmazione, è necessario riavviare l'attività per ottenere l'effetto della modifica delle impostazioni locali Per poter funzionare correttamente per questa soluzione e utilizzare le `android:name=".LocaleApp"` locali dalle preferenze condivise all'avvio dell'app, `android:name=".LocaleApp"` in tu Manifest.xml.

A volte il programma di controllo Lint richiede di creare la build di rilascio. Per risolvere questo problema, segui le seguenti opzioni.

Primo:

Se si desidera disabilitare la conversione solo per alcune stringhe, quindi aggiungere il seguente attributo a default string.xml

```
<string name="developer" translatable="false">Developer Name</string>
```

Secondo:

Ignora tutta la traduzione mancante dal file di risorse aggiungi il seguente attributo È l'attributo ignore dello spazio dei nomi degli strumenti nel file delle stringhe, come segue:

```
<?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>
```

Terzo:

Un altro modo per disabilitare la stringa non traducibile

http://tools.android.com/recent/non-translatablestrings

Se si dispone di molte risorse che non devono essere tradotte, è possibile inserirle in un file denominato donottranslate.xml e Lint considererà tutte le risorse non traducibili.

Il quarto:

È inoltre possibile aggiungere impostazioni internazionali nel file di risorse

```
<resources
xmlns:tools="http://schemas.android.com/tools"
    tools:locale="en" tools:ignore="MissingTranslation">
```

Puoi anche disabilitare il controllo della traduzione mancante per lint da app / build.gradle

```
lintOptions {

        disable 'MissingTranslation'
    }
```

Leggi Localizzazione con risorse in Android online:
https://riptutorial.com/it/android/topic/3345/localizzazione-con-risorse-in-android

Capitolo 147: Looper

introduzione

Un `Looper` è una classe Android utilizzata per eseguire un loop di messaggi per un thread, che di solito non ne ha uno associato.

Il `Looper` più comune in Android è il ciclo principale, noto anche come thread principale. Questa istanza è unica per un'applicazione e può essere acceduta staticamente con `Looper.getMainLooper()` .

Se un `Looper` è associato al thread corrente, può essere recuperato con `Looper.myLooper()` .

Examples

Crea un semplice LooperThread

Un esempio tipico dell'implementazione di un thread `Looper` fornito dalla documentazione ufficiale utilizza `Looper.prepare()` e `Looper.loop()` e associa un `Handler` con il loop tra queste chiamate.

```
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();
    }
}
```

Esegui un ciclo con un HandlerThread

Un `HandlerThread` può essere utilizzato per avviare un thread con un `Looper` . Questo looper può quindi essere utilizzato per creare un `Handler` per le comunicazioni con esso.

```
HandlerThread thread = new HandlerThread("thread-name");
thread.start();
Handler handler = new Handler(thread.getLooper());
```

Leggi Looper online: https://riptutorial.com/it/android/topic/10593/looper

Capitolo 148: LRUCache

Osservazioni

È necessario utilizzare Lru Cache in applicazioni in cui carichi di risorse ripetitivi potrebbero influire sul corretto funzionamento dell'app. Ad esempio una galleria fotografica con miniature grandi (128x128).

Fai sempre attenzione con le dimensioni della cache Lru poiché l'impostazione troppo alta potrebbe influire sull'app.

Dopo che la Lru Cache non è più utile, evita di tenere qualsiasi riferimento ad essa per consentire a Garbage Collector di ripulirlo dalla memoria.

Per prestazioni ottimali, ricordati di caricare risorse come le bitmap usando le migliori pratiche come selezionare un appropriato inSampleSize prima di aggiungerlo alla cache Lru.

Examples

Inizializzazione della cache

The Lru Cache memorizzerà tutte le risorse aggiunte (valori) per l'accesso rapido fino a raggiungere un limite di memoria, nel qual caso farà cadere la risorsa meno utilizzata (valore) per memorizzare quella nuova.

Per inizializzare la cache Lru è necessario fornire un valore di memoria massimo. Questo valore dipende dai requisiti dell'applicazione e dalla criticità della risorsa per mantenere un uso regolare dell'app. Ad esempio, un valore consigliato per una galleria di immagini sarebbe 1/8 della memoria massima disponibile.

Si noti inoltre che Lru Cache funziona su una base di valori-chiave. Nell'esempio seguente, la chiave è una `String` e il valore è una `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();
    }
};
```

Aggiunta di una bitmap (risorsa) alla cache

Per aggiungere una risorsa alla cache è necessario fornire una chiave e la risorsa. Innanzitutto assicurati che il valore non sia già nella cache

```
public void addResourceToMemoryCache(String key, Bitmap resource) {
    if (memoryCache.get(key) == null)
        memoryCache.put(key, resource);
}
```

Ottenere una bitmap (Resouce) dalla cache

Per ottenere una risorsa dalla cache basta passare la chiave della tua risorsa (String in questo esempio)

```
public Bitmap getResourceFromMemoryCache(String key) {
    memoryCache.get(key);
}
```

Leggi LRUCache online: https://riptutorial.com/it/android/topic/7709/lrucache

Capitolo 149: Macchina fotografica e galleria

Examples

Scattare foto a grandezza naturale dalla fotocamera

Per fare una foto, prima dobbiamo dichiarare le autorizzazioni richieste in `AndroidManifest.xml` . Abbiamo bisogno di due permessi:

- `Camera` : per aprire l'app Fotocamera. Se l'attributo `required` è impostato su `true` , non sarà possibile installare questa app se non si dispone di una videocamera hardware.
- `WRITE_EXTERNAL_STORAGE` - Questa autorizzazione è necessaria per creare un nuovo file, in cui verrà salvata la foto catturata.

AndroidManifest.xml

```
<uses-feature android:name="android.hardware.camera"
          android:required="true" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

L'idea principale di scattare foto a grandezza naturale dalla fotocamera è che abbiamo bisogno di creare un nuovo file per foto, prima di aprire l'app per fotocamera e acquisire 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();
    }
}
```

Fare foto

Aggiungi un'autorizzazione per accedere alla videocamera al file AndroidManifest:

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

File 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>
```

Attività

```
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;
    }
}
```

Come avviare la fotocamera o la galleria e salvare i risultati della fotocamera nella memoria

Prima di tutto hai bisogno di cartelle `Uri` e temp e codici di richiesta:

```
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;
```

Quindi inizia 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 degli intenti di `Camera` e `Gallery` :

```
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);
    }

}
```

Quindi nel metodo `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);
}
```

Hai bisogno di queste autorizzazioni in `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" />
```

E devi gestire permessi di runtime come lettura / scrittura di archiviazione esterna ecc ...

Sto controllando `READ_EXTERNAL_STORAGE` autorizzazione `READ_EXTERNAL_STORAGE` nel mio metodo `openGallery` :

Il mio metodo `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);
    }
}
```

Quindi sovrascrivi il metodo `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);
    }
}
```

metodo `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();
}
```

Imposta la risoluzione della fotocamera

Imposta ad alta risoluzione a livello di codice.

```
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);
```

Decodifica bitmap correttamente ruotata dall'ur e scaricata con l'intento

```
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;
}
```

Leggi Macchina fotografica e galleria online: https://riptutorial.com/it/android/topic/4789/macchina-fotografica-e-galleria

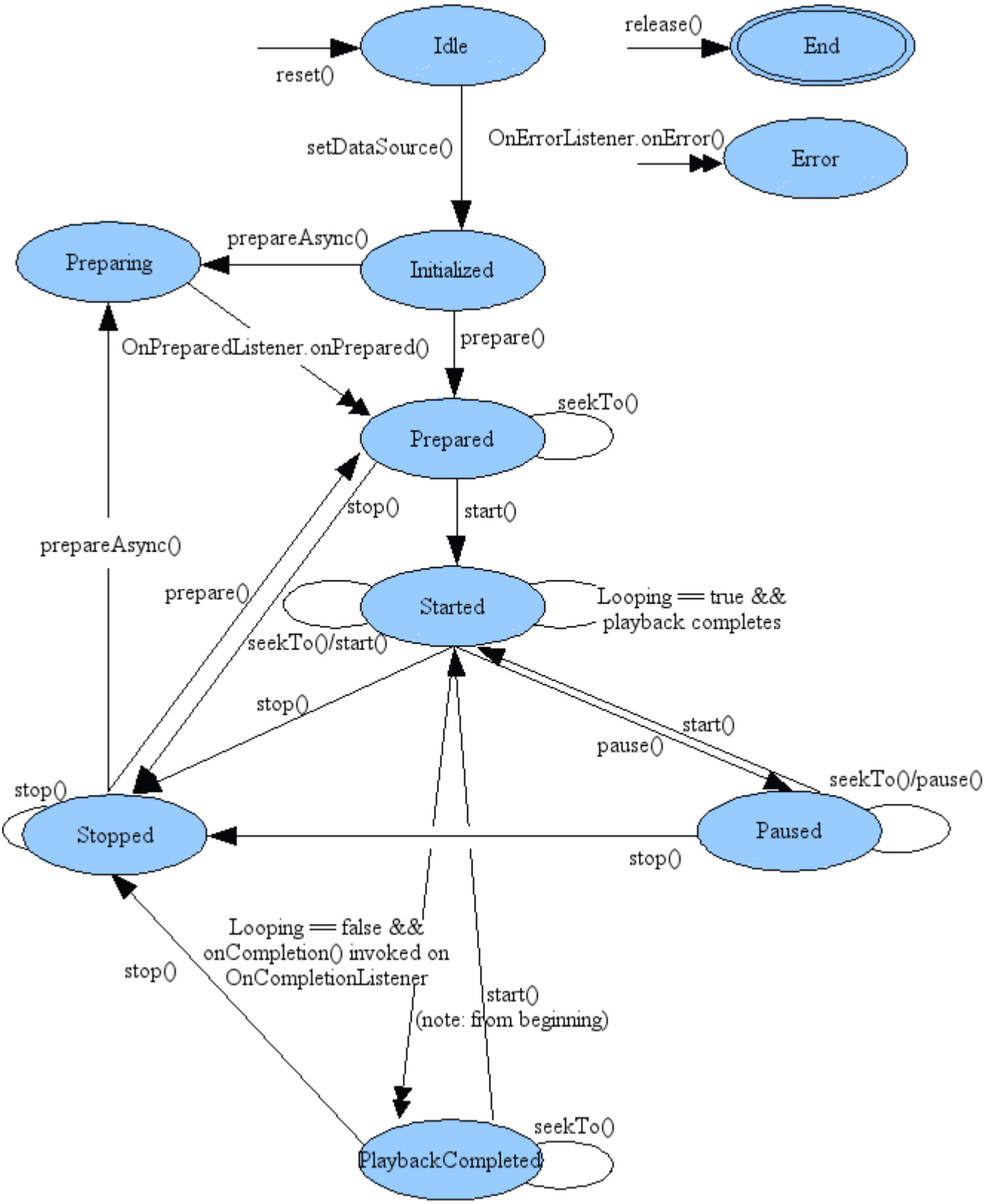
Capitolo 150: Media Player

Sintassi

- void setAudioStreamType (int streamtype)
- void setDataSource (Contesto contesto, Uri uri)
- vuoto prepara ()
- void prepareAsync ()
- void start ()
- void stop ()

Osservazioni

L'utilizzo di `MediaPlayer` si basa principalmente sul diagramma dello stato:



Ciò significa che per riprodurre audio / video è necessario che una determinata sequenza di azioni avvenga in un ordine specifico. Indica anche quali azioni possono essere fatte in quale stato .

L'API di MediaPlayer non ha flessibilità (aggiunta di decoder e logica di rendering personalizzati) e manca sSupport per Dynamic Adaptive Streaming su HTTP (DASH) e SmoothStreaming. Per questi, guarda in ExoPlayer .

Examples

Creazione e riproduzione di base

La classe MediaPlayer può essere utilizzata per controllare la riproduzione di file e flussi audio / video.

La creazione dell'oggetto MediaPlayer può essere di tre tipi:

1. Media dalla risorsa locale

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.resource);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

2. Dall'URI locale (ottenuto da ContentResolver)

```
Uri myUri = ....; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

3. Da URL esterno

```
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();
```

Preparazione asincrona

`MediaPlayer$prepare()` è una chiamata bloccante e bloccherà l'interfaccia utente fino al completamento dell'esecuzione. Per risolvere questo problema, è possibile utilizzare `MediaPlayer$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
```

Nelle operazioni sincrone, gli errori vengono normalmente segnalati con un'eccezione o un codice di errore, ma ogni volta che si utilizzano risorse asincrone, è necessario assicurarsi che l'applicazione venga notificata correttamente degli errori. Per 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
        }
});
```

Ottenere suonerie di sistema

Questo esempio mostra come recuperare l'URI delle suonerie di 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;
    }
```

L'elenco dipende dal tipo di suoneria richiesta. Le possibilità sono:

- `RingtoneManager.TYPE_RINGTONE`
- `RingtoneManager.TYPE_NOTIFICATION`
- `RingtoneManager.TYPE_ALARM`
- `RingtoneManager.TYPE_ALL = TYPE_RINGTONE | TYPE_NOTIFICATION | TYPE_ALARM`

Per ottenere le suonerie come `android.media.Ringtone` ogni `Uri` deve essere risolto dal `RingtoneManager` :

```
android.media.Ringtone osRingtone = RingtoneManager.getRingtone(context, uri);
```

Per riprodurre il suono, utilizzare il metodo:

```
public void setDataSource(Context context, Uri uri)
```

da `android.media.MediaPlayer` . `MediaPlayer` deve essere inizializzato e preparato secondo il diagramma di stato

Ottenere e impostare il volume del sistema

Tipi di flusso audio

Esistono diversi profili di flussi di suonerie. Ognuno di loro ha il suo volume diverso.

Ogni esempio qui è scritto per il tipo di flusso `AudioManager.STREAM_RING` . Tuttavia questo non è l'unico. I tipi di flusso disponibili sono:

- `STREAM_ALARM`
- `STREAM_DTMF`
- `STREAM_MUSIC`
- `STREAM_NOTIFICATION`
- `STREAM_RING`
- `STREAM_SYSTEM`
- `STREAM_VOICE_CALL`

Impostazione del volume

Per ottenere il volume del profilo specifico, chiama:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int currentVolume = audioManager.getStreamVolume(AudioManager.STREAM_RING);
```

Questo valore è molto poco utile, quando il valore massimo per il flusso non è noto:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int streamMaxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM_RING);
```

Il rapporto tra questi due valori darà un volume relativo (0 <volume <1):

```
float volume = ((float) currentVolume) / streamMaxVolume
```

Regolazione del volume di un passo

Per aumentare il volume dello stream di un passo, chiama:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_RAISE, 0);
```

Per ridurre il volume dello stream di un passo, chiama:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_LOWER, 0);
```

Impostazione di MediaPlayer per utilizzare il tipo di stream specifico

Per fare ciò è disponibile una funzione di supporto dalla classe `MediaPlayer` .
Basta chiamare `void setAudioStreamType(int streamtype)` :

```
MediaPlayer mMedia = new MediaPlayer();
mMedia.setAudioStreamType(AudioManager.STREAM_RING);
```

Lettore multimediale con avanzamento del buffer e posizione di riproduzione

```
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();
```

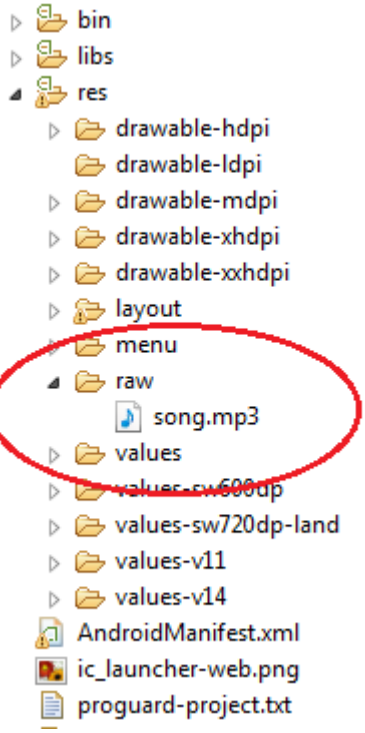
```
                mediaPlayer.reset();
            }
        });
        mediaPlayer.setOnBufferingUpdateListener(new MediaPlayer.OnBufferingUpdateListener() {
            @Override
            public void onBufferingUpdate(MediaPlayer mp, int percent) {
                progress_bar.setSecondaryProgress(percent);
            }
        });
        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>
```

Importa audio in Androidstudio e riproducilo

Questo è un esempio di come ottenere la riproduzione di un file audio che hai già sul tuo pc / laptop. Per prima cosa crea una nuova directory sotto res e chiamala come raw come questa



copia l'audio che vuoi riprodurre in questa cartella. Può essere un file .mp3 o .wav.

Ora, ad esempio, sul pulsante clic si desidera riprodurre questo suono, ecco come è fatto

```
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();
        }
    });
}
}
```

Questo riprodurrà la canzone solo una volta quando si fa clic sul pulsante, se si desidera riprodurre la canzone su ogni pulsante, fare clic su un codice di scrittura come questo

```
 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();
     }
   });
 }
}
```

Leggi Media Player online: https://riptutorial.com/it/android/topic/1851/media-player

Capitolo 151: MediaSession

Sintassi

- void mediaSessionCompat.setFlags (int flags)
- void mediaSessionCompat.setMediaButtonReceiver (PendingIntent mbr)
- void mediaSessionCompat.setCallback (callback MediaSessionCompat.Callback)
- void mediaSessionCompat.setActive (attivo booleano)
- MediaSessionCompat.Token mediaSessionCompat.getSessionToken ()
- void mediaSessionCompat.release ()
- void mediaSessionCompat.setPlaybackState (stato PlaybackStateCompat)
- void mediaSessionCompat.setMetadata (metadati MediaMetadataCompat)

Osservazioni

Per la migliore pratica, utilizzare la libreria **media-compat** . La libreria si occupa della compatibilità con le versioni precedenti traducendo i metodi di sessione dei media in metodi equivalenti su versioni di piattaforme precedenti, se disponibili.

Examples

Ricezione e gestione degli eventi del pulsante

Questo esempio crea un oggetto `MediaSession` all'avvio di un `Service` . L'oggetto `MediaSession` viene rilasciato quando il `Service` viene distrutto:

```
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();
    }
}
```

Il seguente metodo crea un'istanza e configura i callback del pulsante `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);
}
```

Il seguente metodo invia metadati (memorizzati in una `HashMap`) al dispositivo usando 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);
}
```

Il seguente metodo imposta il `PlaybackState` . Imposta inoltre quali azioni del `MediaSession` risponderà a `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);
}
```

Leggi MediaSession online: https://riptutorial.com/it/android/topic/6250/mediasession

Capitolo 152: MediaStore

Examples

Recupera file audio / MP3 da una cartella specifica del dispositivo o recupera tutti i file

Innanzitutto, aggiungi le seguenti autorizzazioni al manifest del tuo progetto per abilitare l'accesso allo storage del dispositivo:

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

Quindi, crea il file *AudioModel.class* e inserisci la seguente classe di modello per consentire di recuperare e impostare gli elementi dell'elenco:

```
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;
    }
}
```

Quindi, utilizzare il seguente metodo per leggere tutti i file MP3 da una cartella del dispositivo o leggere tutti i file del 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;
}
```

Il codice sopra restituirà un elenco di tutti i file MP3 con nome, percorso, artista e album della musica. Per maggiori dettagli, consultare la documentazione di MediaStore.Audio .

Per leggere i file di una cartella specifica, utilizzare la seguente query (è necessario sostituire il nome della cartella):

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    MediaStore.Audio.Media.DATA + " like ? ",
    new String[]{"%yourFolderName%"}, // Put your device folder / file location here.
    null);
```

Se si desidera recuperare tutti i file dal dispositivo, utilizzare la seguente query:

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    null,
    null,
    null);
```

Nota: non dimenticare di abilitare le autorizzazioni di accesso alla memoria.

Ora, tutto ciò che devi fare è chiamare il metodo sopra per ottenere i file MP3:

```
getAllAudioFromDevice(this);
```

Esempio con attività

```
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;
    }
}
```

Leggi MediaStore online: https://riptutorial.com/it/android/topic/7136/mediastore

Capitolo 153: Memorizzazione di file in Archiviazione interna ed esterna

Sintassi

- FileOutputStream openFileInput (nome stringa)
- FileOutputStream openFileOutput (Nome stringa, modalità int)
- File (dir file, nome stringa)
- File (percorso stringa)
- File getExternalStoragePublicDirectory (String type)
- File getExternalFilesDir (Tipo di stringa)

Parametri

Parametro	Dettagli
nome	Il nome del file da aprire. NOTA: non può contenere separatori di percorso
modalità	Modalità operativa. Utilizzare `MODE_PRIVATE` per l'operazione predefinita e `MODE_APPEND` per aggiungere un file esistente. Altre modalità includono `MODE_WORLD_READABLE` e `MODE_WORLD_WRITEABLE` , entrambe deprecate in API 17.
dir	Directory del file per creare un nuovo file in
sentiero	Percorso per specificare la posizione del nuovo file
genere	Tipo di directory dei file da recuperare. Può essere `null` o uno dei seguenti elementi: `DIRECTORY_MUSIC` , `DIRECTORY_PODCASTS` , `DIRECTORY_RINGTONES` , `DIRECTORY_ALARMS` , `DIRECTORY_NOTIFICATIONS` , `DIRECTORY_PICTURES` o `DIRECTORY_MOVIES`

Examples

Utilizzo dell'archiviazione interna

Per impostazione predefinita, qualsiasi file salvato in Archiviazione interna è privato per la tua applicazione. Non è possibile accedervi da altre applicazioni, né dall'utente in circostanze normali. **Questi file vengono cancellati quando l'utente disinstalla l'applicazione** .

Scrivere un testo in un file

```
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();
}
```

Per aggiungere testo a un file esistente

Utilizzare `Context.MODE_APPEND` per il parametro mode di `openFileOutput`

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

Uso dell'archiviazione esterna

L'archiviazione "esterna" è un altro tipo di archiviazione che è possibile utilizzare per salvare i file sul dispositivo dell'utente. Ha alcune differenze chiave rispetto allo storage "interno", vale a dire:

- Non è sempre disponibile. Nel caso di un supporto rimovibile (scheda SD), l'utente può semplicemente rimuovere la memoria.
- Non è privato. L'utente (e altre applicazioni) hanno accesso a questi file.
- Se l'utente disinstalla l'app, i file salvati nella directory recuperata con `getExternalFilesDir()` verranno rimossi.

Per utilizzare Storage esterno, dobbiamo prima ottenere le autorizzazioni appropriate. Dovrai usare:

- `android.permission.WRITE_EXTERNAL_STORAGE` per la lettura e la scrittura
- `android.permission.READ_EXTERNAL_STORAGE` per la sola lettura

Per concedere queste autorizzazioni, dovrai identificarle nel tuo `AndroidManifest.xml` come tale

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

> NOTA: poiché si tratta di autorizzazioni pericolose se si utilizza **API di livello 23** o superiore, sarà necessario richiedere le **autorizzazioni in fase di runtime** .

Prima di tentare di scrivere o leggere da Storage esterno, è necessario verificare sempre che il supporto di memorizzazione sia disponibile.

```
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
}
```

Quando si scrivono file nell'archivio esterno, è necessario decidere se il file deve essere riconosciuto come pubblico o privato. Sebbene entrambi questi tipi di file siano ancora accessibili all'utente e ad altre applicazioni sul dispositivo, esiste una distinzione fondamentale tra di essi.

I file pubblici devono rimanere sul dispositivo quando l'utente disinstalla l'app. Un esempio di un file che dovrebbe essere salvato come pubblico sarebbero le foto che sono state prese attraverso la tua applicazione.

I file privati devono essere rimossi tutti quando l'utente disinstalla l'app. Questi tipi di file potrebbero essere specifici per le app e non essere utili all'utente o ad altre applicazioni. Ex. file temporanei scaricati / utilizzati dalla tua applicazione.

Ecco come accedere alla directory `Documents` per file pubblici e privati.

Pubblico

```
// 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();
```

Privato

```
// 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: archiviazione interna ed esterna - Chiarimento terminologico

Gli sviluppatori Android (principalmente principianti) sono stati confusi riguardo la terminologia di archiviazione interna ed esterna. Ci sono molte domande su Stackoverflow riguardanti lo stesso. Ciò è dovuto principalmente al fatto che la *terminologia* secondo la documentazione Google / ufficiale di Android è molto diversa da quella del normale utente del sistema operativo Android. Quindi ho pensato che documentarlo sarebbe stato d'aiuto.

Cosa pensiamo - User's Terminology (UT)

Memoria interna (UT)	Memoria esterna (UT)
memoria interna incorporata nel telefono	scheda Secure Digital (SD) rimovibile o memoria micro SD
Esempio: memoria interna da 32 GB del Nexus 6P.	**Esempio:** spazio di archiviazione in schede SD rimovibili fornite da venditori come Samsung, Sandisk, Stronzio, Trascendi e altri

Ma, **secondo la documentazione / guida di Android - Google's Terminology (GT)**

Memoria interna (GT):

> Per impostazione predefinita, i file salvati nella memoria interna sono privati per l'applicazione e altre applicazioni non possono accedervi (né l'utente può).

Memoria esterna (GT):

> Può trattarsi di un supporto di memorizzazione rimovibile (come una scheda SD) o di una memoria interna (non rimovibile).

External Storage (GT) può essere classificato in due tipi:

Storage esterno primario	Memoria esterna secondaria o memoria rimovibile (GT)
Questo è lo stesso della memoria interna incorporata del telefono (o) della memoria interna (UT)	Questo è lo stesso della memoria micro SD rimovibile (o) Memoria esterna (UT)
Esempio: memoria interna da 32 GB del Nexus 6P.	**Esempio:** spazio di archiviazione in schede SD rimovibili fornite da venditori come Samsung, Sandisk, Stronzio, Trascendi e altri
È possibile accedere a questo tipo di archiviazione su Windows PC collegando il telefono al PC tramite cavo USB e selezionando *Fotocamera (PTP)* nella notifica delle opzioni USB.	È possibile accedere a questo tipo di archiviazione su Windows PC collegando il telefono al PC tramite cavo USB e selezionando *Trasferimento file* nella notifica delle opzioni USB.

In poche parole,

Storage esterno (GT) = Internal Storage (UT) e External Storage (UT)

Archivi rimovibili (GT) = Storage esterno (UT)

Internal Storage (GT) non ha un termine in UT.

Lasciami spiegare chiaramente

Internal Storage (GT): per impostazione predefinita, i file salvati nella memoria interna sono privati per l'applicazione e altre applicazioni non possono accedervi. Anche l'utente dell'app non può accedervi utilizzando il file manager; anche dopo aver abilitato l'opzione "mostra file nascosti" nel file manager. Per accedere ai file in Archiviazione interna (GT), devi eseguire il root del tuo telefono Android. Inoltre, quando l'utente disinstalla l'applicazione, questi file vengono rimossi / eliminati.

Quindi Internal Storage (GT) **NON** è ciò che pensiamo come memoria interna 32/64 GB di Nexus

6P

Generalmente, la posizione di **Internal Storage (GT)** sarebbe:
`/data/data/your.application.package.appname/someDirectory/`

Storage esterno (GT):

> Ogni dispositivo compatibile con Android supporta una "memoria esterna" condivisa che è possibile utilizzare per salvare i file. I file salvati nella memoria esterna sono leggibili da tutto il mondo e possono essere modificati dall'utente quando abilitano la memorizzazione di massa USB per trasferire file su un computer.

Posizione di archiviazione esterna (GT): potrebbe trovarsi in *qualsiasi punto* della memoria interna (UT) o nella memoria rimovibile (GT), ad esempio la scheda micro SD. Dipende dall'OEM del tuo telefono e anche dalla versione del SO Android.

Per leggere o scrivere file su Storage esterno (GT), l'app deve acquisire le autorizzazioni di sistema `READ_EXTERNAL_STORAGE` o `WRITE_EXTERNAL_STORAGE` .

Per esempio:

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

> Se è necessario sia leggere che scrivere file, è necessario richiedere solo l'autorizzazione `WRITE_EXTERNAL_STORAGE` , poiché richiede implicitamente anche l'accesso in lettura.

In **External Storage (GT)** , puoi anche salvare i file che sono **app-private**

Ma,

> Quando l'utente disinstalla l'applicazione, questa directory e tutto il suo contenuto vengono cancellati.

Quando è necessario salvare i file che sono **app-private** in **External Storage (GT)** ?

> Se gestisci file che non sono destinati ad altre app (come trame grafiche o effetti sonori utilizzati solo dalla tua app), dovresti utilizzare una directory di archiviazione privata nella memoria esterna
>
> A partire da Android 4.4, leggere o scrivere file nelle directory private della tua app non richiede le autorizzazioni `READ_EXTERNAL_STORAGE` o `WRITE_EXTERNAL_STORAGE` . Quindi puoi dichiarare che il permesso dovrebbe essere richiesto solo nelle versioni inferiori di Android aggiungendo l'attributo `maxSdkVersion` :

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

```
    ...
</manifest
```

Metodi per memorizzare in Internal Storage (GT):

Entrambi questi metodi sono presenti nella classe Context

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

File getFilesDir ()
```

Metodi per archiviare nello Storage esterno primario ie Internal Storage (UT):

```
File getExternalStorageDirectory ()

File getExternalFilesDir (String type)

File getExternalStoragePublicDirectory (String type)
```

All'inizio, tutti usavano Environment.getExternalStorageDirectory () , che puntava alla **radice** di **Primary External Storage** . Di conseguenza, lo Storage esterno primario era pieno di contenuti casuali.

Successivamente, sono stati aggiunti questi due metodi:

1. Nella classe `Context` , hanno aggiunto getExternalFilesDir () , che punta a una **directory specifica dell'applicazione** su Primary External Storage. Questa directory e il suo contenuto **verranno eliminati** quando l'app viene disinstallata.

2. Environment.getExternalStoragePublicDirectory () per luoghi centralizzati per archiviare tipi di file ben noti, come foto e filmati. Questa directory e il suo contenuto **NON saranno cancellati** quando l'app viene disinstallata.

Metodi per archiviare in Archivi rimovibili (GT) cioè micro SD card

Prima del **livello API 19** , non esisteva **un modo ufficiale** di memorizzare nella scheda SD. Ma molti potrebbero farlo usando librerie o API non ufficiali.

Ufficialmente, un metodo è stato introdotto nella classe `Context` in livello API 19 (versione Android 4.4 - Kitkat).

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

> Restituisce percorsi assoluti alle directory specifiche dell'applicazione su tutti i dispositivi di archiviazione condivisi / esterni in cui l'applicazione può posizionare i file permanenti che possiede. Questi file sono interni all'applicazione e in genere non sono visibili all'utente come supporto.

Ciò significa che restituirà percorsi per **entrambi i** tipi di memoria esterna (GT) - memoria interna e scheda Micro SD. Generalmente il **secondo percorso** sarebbe il percorso di archiviazione della

scheda micro SD (ma non sempre). Quindi è necessario verificarlo eseguendo il codice con questo metodo.

Esempio con lo snippet di codice:

Ho creato un nuovo progetto Android con attività vuote, ho scritto il seguente codice all'interno

`protected void onCreate(Bundle savedInstanceState)` metodo di `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);
```

Dopo aver eseguito il codice sopra,

Produzione:

```
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: ho collegato il mio telefono a PC Windows; abilitato entrambe le opzioni sviluppatore, debug USB e quindi eseguito questo codice. Se **non si collega il telefono** ; ma eseguirlo su un **emulatore Android** , l'output potrebbe variare. Il mio modello di telefono è Coolpad Note 3 - in

esecuzione su Android 5.1

Posizioni di archiviazione sul mio telefono:

Posizione di memoria Micro SD : `/storage/sdcard1`

Posizione di archiviazione interna (UT) : `/storage/sdcard0` .

Nota che `/sdcard` e `/storage/emulated/0` puntano anche a Internal Storage (UT). Ma questi sono `/storage/sdcard0` simbolici a `/storage/sdcard0` .

Per comprendere chiaramente diversi percorsi di archiviazione in Android, perfavore segui questa risposta

Dichiarazione di non responsabilità: tutti i percorsi di archiviazione menzionati sopra sono percorsi sul **mio** telefono. I tuoi file potrebbero **non** essere memorizzati su stessi percorsi di archiviazione. Perché, le posizioni / percorsi di archiviazione possono variare su altri telefoni cellulari a seconda del produttore, del produttore e delle diverse versioni del sistema operativo Android.

Salva database su scheda SD (backup DB su 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;
    }
}
```

chiama questo metodo in questo modo:

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

Recupera la directory dei dispositivi:

Prima di tutto Aggiungi autorizzazione di archiviazione per leggere / recuperare la directory del dispositivo.

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

Crea classe modello

```
//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;
    }
}
```

Crea una lista usando il modello di directory per aggiungere i dati della directory.

```
//define list to show directory

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

Recupera la directory usando il seguente metodo.

```
//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();
        }
    }
```

Stampa l'elenco delle directory nel 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 );
```

Per recuperare i file interni / cartella della directory specifica utilizzare lo stesso metodo basta cambiare argomento, passare il percorso corrente selezionato in argomento e gestire la risposta per lo stesso.

Per ottenere l'estensione del file:

```
private String getExtension(String filename) {
```

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

    return extension;
}
```

Leggi Memorizzazione di file in Archiviazione interna ed esterna online: https://riptutorial.com/it/android/topic/150/memorizzazione-di-file-in-archiviazione-interna-ed-esterna

Capitolo 154: Menu

Sintassi

- inflater.inflate (R.menu.your_xml_file, menu);

Parametri

Parametro	Descrizione
`inflate(int menuRes, Menu menu)`	Gonfiare una gerarchia di menu dalla risorsa XML specificata.
`getMenuInflater ()`	Restituisce un `MenuInflater` con questo contesto.
`onCreateOptionsMenu (Menu menu)`	Inizializza i contenuti del menu delle opzioni standard di Activity. È necessario posizionare le voci di menu nel menu.
`onOptionsItemSelected (MenuItem item)`	Questo metodo viene chiamato ogni volta che viene selezionato un elemento nel menu delle opzioni

Osservazioni

Per saperne di più su **Menu** , leggi questo . Spero che sia d'aiuto!

Examples

Menu delle opzioni con divisori

In Android c'è un menu di opzioni predefinito, che può richiedere un certo numero di opzioni. Se è necessario visualizzare un numero maggiore di opzioni, è opportuno raggruppare tali opzioni per mantenere la chiarezza. Le opzioni possono essere raggruppate inserendo divisori (cioè linee orizzontali) tra di loro. Per consentire divisori, è possibile utilizzare il seguente 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>
```

Modificando il tema, i divisori possono essere aggiunti a un menu.

Applica il carattere personalizzato al Menu

```
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);
}
```

e poi nell'attività:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    applyFontToMenu(menu,this);
    return true;
}
```

Creazione di un menu in un'attività

Per definire il tuo menu, crea un file XML all'interno della cartella `res/menu/` directory del tuo progetto e crea il menu con i seguenti elementi:

- `<menu>` : definisce un Menu, che contiene tutte le voci del menu.
- `<item>` : crea un oggetto Menu, che rappresenta un singolo elemento in un menu. Possiamo anche creare un elemento annidato per creare un sottomenu.

Passo 1:

Crea il tuo file xml come segue:

In `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>
```

Passo 2:

Per specificare il menu opzioni, sovrascrivere `onCreateOptionsMenu()` nella tua *attività* .

In questo metodo, puoi gonfiare la risorsa del tuo menu (definita nel tuo file XML, ad esempio, `res/menu/main_menu.xml`)

```
 @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }
```

Quando l'utente seleziona una voce dal menu delle opzioni, il sistema chiama override *del* vostro *attività* `onOptionsItemSelected()` metodo.

- Questo metodo passa il MenuItem selezionato.
- È possibile identificare l'elemento chiamando `getItemId()` , che restituisce l'ID univoco per la voce di menu (definita dall'account `android:id attribute` nel menu resource - `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);
    }
}
```

Avvolgendo!

Il tuo codice `Activity` dovrebbe essere simile a quanto segue:

```
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);
        }
    }
}
```

Screenshot di come appare il tuo menu:

Leggi Menu online: https://riptutorial.com/it/android/topic/2028/menu

Capitolo 155: Metriche di visualizzazione del dispositivo

Examples

Ottieni le dimensioni in pixel dello schermo

Per recuperare la larghezza e l'altezza dei pixel in pixel, possiamo utilizzare le metriche di visualizzazione di WindowManager .

```
// Get display metrics
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
```

Questi DisplayMetrics contengono una serie di informazioni sullo schermo dei dispositivi, come la sua densità o dimensione:

```
// Get width and height in pixel
Integer heightPixels = metrics.heightPixels;
Integer widthPixels = metrics.widthPixels;
```

Ottieni la densità dello schermo

Per ottenere la densità dello schermo, possiamo anche utilizzare WindowMaker DisplayMetrics . Questo è un rapido esempio:

```
// Get density in dpi
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
int densityInDpi =  metrics.densityDpi;
```

Conversione da Formula px a dp, da dp a px

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);
}
```

Leggi Metriche di visualizzazione del dispositivo online:
https://riptutorial.com/it/android/topic/4207/metriche-di-visualizzazione-del-dispositivo

Capitolo 156: Miglioramento delle finestre di dialogo degli avvisi

introduzione

Questo argomento riguarda il miglioramento di un `AlertDialog` con funzionalità aggiuntive.

Examples

Finestra di avviso contenente un link cliccabile

Per mostrare una finestra di avviso contenente un link che può essere aperto facendo clic su di esso, è possibile utilizzare il seguente codice:

```
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
```

Leggi Miglioramento delle finestre di dialogo degli avvisi online:
https://riptutorial.com/it/android/topic/10163/miglioramento-delle-finestre-di-dialogo-degli-avvisi

Capitolo 157: Miglioramento delle prestazioni di Android utilizzando i font icona

Osservazioni

I caratteri di icona sono come tipi di carattere normali che contengono simboli anziché lettere. Può essere utilizzato nella tua applicazione con la massima facilità.

Loro sono:

- Flessibile
- Scalabile
- Vettori
- Veloce processabile
- Peso leggero
- Accessibile

Effetto sulle dimensioni

Esportare un'immagine in varie dimensioni per dispositivi Android costerebbe alla tua app, una dimensione aggiuntiva di circa 30 KB per immagine. Mentre aggiungere un file di font (.ttf) con circa 36 icone costerebbe solo 9kB. Immagina il caso se stai aggiungendo 36 singoli file di varie configurazioni che sarebbero circa 1000kB. È una quantità ragionevole di spazio che verrà salvato utilizzando i caratteri icona.

Limitazioni dei caratteri Icon.

- I caratteri icona possono essere utilizzati nel cassetto di navigazione. Non è possibile utilizzarli nelle viste di navigazione come icona delle voci di menu poiché non è possibile creare il file di menu senza specificare il titolo. Quindi è consigliabile usare i file svg come risorse per queste icone.
- I caratteri icona non possono essere utilizzati nel pulsante di azione mobile. in quanto non hanno un attributo `setText()` .
- I caratteri esterni non possono essere applicati da xml. Devono essere specificati usando il file java. Oppure è necessario estendere la vista di base e creare una vista come specificato in questo post

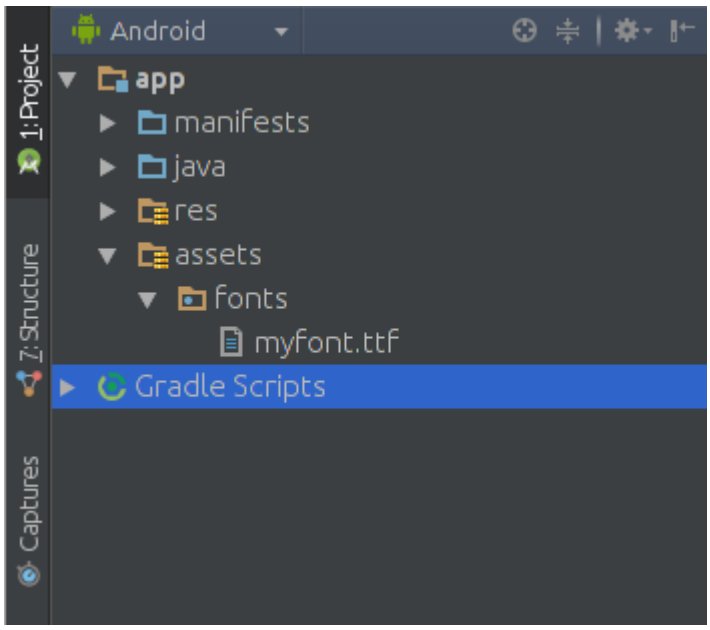
Examples

Come integrare i font Icon

Per utilizzare i font di icone, procedi nel seguente modo:

- **Aggiungi il file di font al tuo progetto**

  È possibile creare il file dell'icona del carattere da siti Web online come icomoon , in cui è possibile caricare i file SVG delle icone richieste e quindi scaricare il carattere dell'icona creato. Quindi, inserisci il file di font *.ttf* in una cartella denominata *fonts* (*nominalo* come desideri) nella cartella assets:



- **Crea una classe di supporto**

  Ora, crea la seguente classe helper, in modo che tu possa evitare di ripetere il codice di inizializzazione per il font:

```
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);
  }
}
```

  È possibile utilizzare la classe `Typeface` per prelevare il carattere dalle risorse. In questo modo è possibile impostare il carattere tipografico su varie viste, ad esempio su un pulsante:

```
Button button=(Button) findViewById(R.id.button);
Typeface iconFont=FontManager.getTypeface(getApplicationContext());
button.setTypeface(iconFont);
```

  Ora, il carattere tipografico del pulsante è stato modificato con il carattere dell'icona appena creato.

- **Raccogli le icone che vuoi**

  Apri il file *styles.css* collegato al carattere dell'icona. Lì troverai gli stili con i caratteri Unicode delle tue icone:

```
.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";
}
```

Questo file di risorse fungerà da dizionario, che associa il carattere Unicode associato a un'icona specifica a un nome leggibile dall'uomo. Ora, crea le risorse stringa come segue:

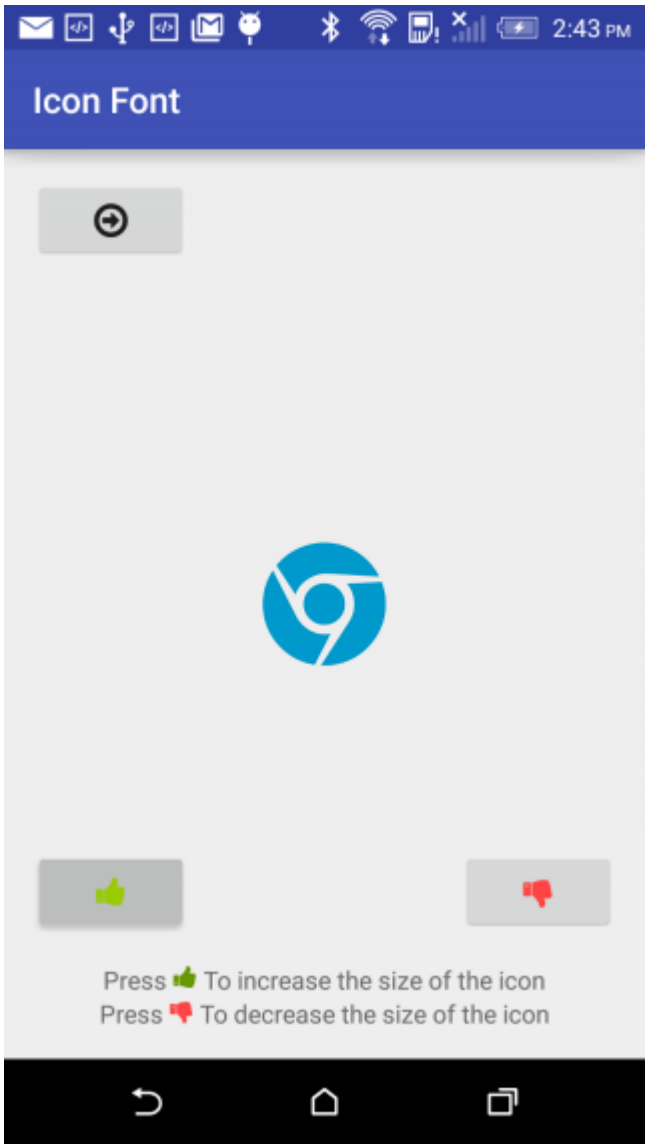
```
<resources>
  <! — Icon Fonts -->
  <string name="icon_arrow_circle_down">&#xe001; </string>
  <string name="icon_arrow_circle_left">&#xe002; </string>
  <string name="icon_arrow_circle-o_down">&#xe003; </string>
  <string name="icon_arrow_circle_o_left">&#xe004; </string>
</resources>
```

- **Usa le icone nel tuo codice**

  Ora, puoi usare il tuo font in varie viste, ad esempio, come segue:

```
button.setText(getString(R.string.icon_arrow_circle_left))
```

  Puoi anche creare viste di testo pulsante utilizzando i caratteri icona:



TabLayout con caratteri icona

```
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;
}

}
```

- In questo esempio, myfont.ttf si trova nella cartella Assets. Creazione della cartella Risorse
- Nella tua classe di attività

```
//..
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);
    //..
```

Leggi Miglioramento delle prestazioni di Android utilizzando i font icona online: https://riptutorial.com/it/android/topic/3642/miglioramento-delle-prestazioni-di-android-utilizzando-i-font-icona

Capitolo 158: Modalità PorterDuff

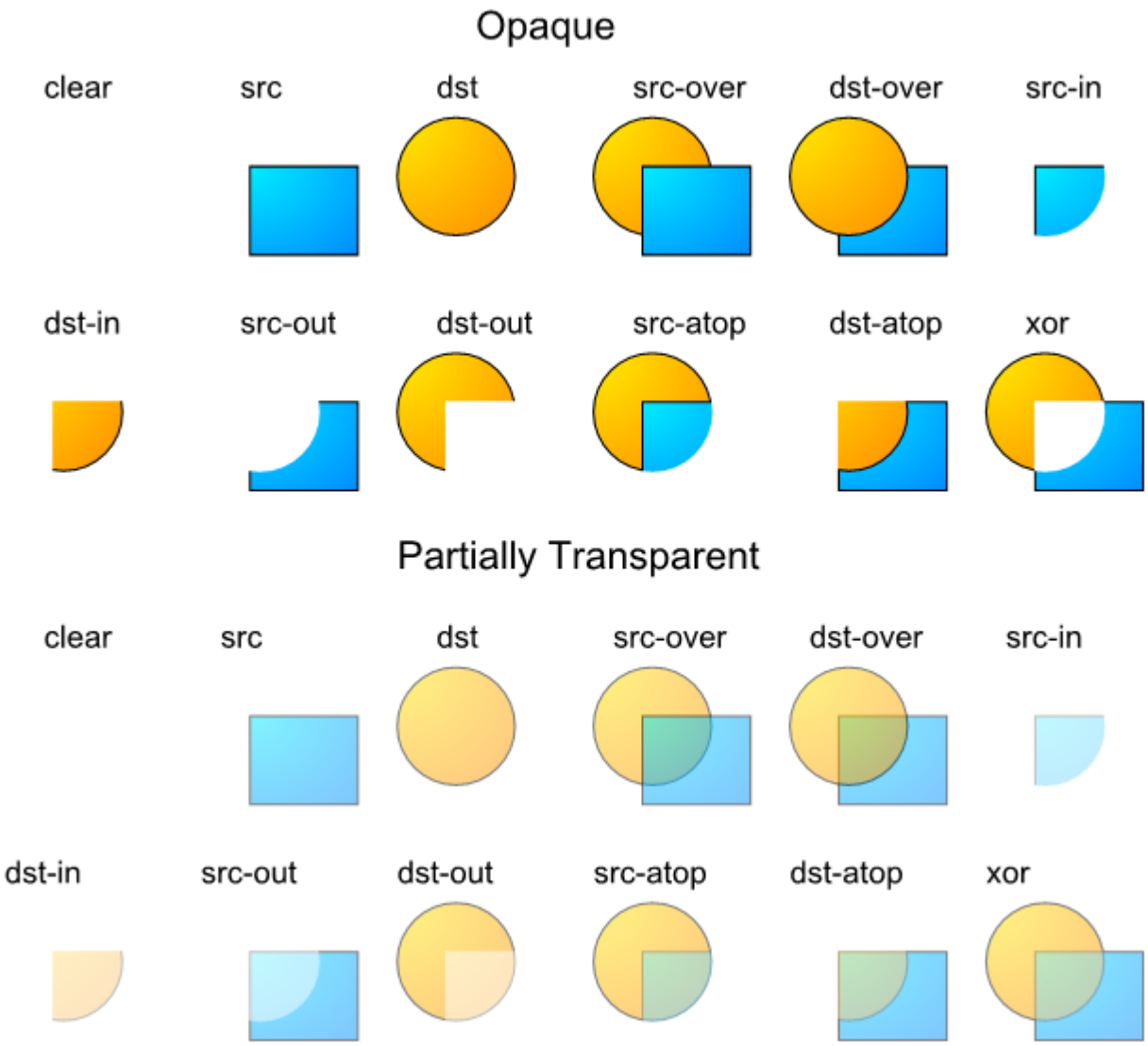
introduzione

PorterDuff è descritto come un modo di combinare le immagini come se fossero "pezzi di cartone di forma irregolare" sovrapposti l'uno sull'altro, nonché uno schema per miscelare le parti sovrapposte

Osservazioni

"Porter Duff" in sé è una tecnica di compositing alfa che prende il nome da un articolo di Thomas Porter e Tom Duff.

Per riassumere, la tecnica prende due immagini con canale alfa e genera l'immagine di output combinando i valori dei pixel di due immagini. Le varie modalità di combinazione producono immagini di output differenti. Ad esempio, nell'immagine seguente, la forma blu (origine, pixel esistenti) viene combinata con la forma gialla (destinazione, nuovi pixel) in diverse modalità:



Examples

Creazione di un filtro colorato PorterDuff

`PorterDuff.Mode` viene utilizzato per creare un `PorterDuffColorFilter` . Un filtro colore modifica il colore di ciascun pixel di una risorsa visiva.

```
ColorFilter filter = new PorterDuffColorFilter(Color.BLUE, PorterDuff.Mode.SRC_IN);
```

Il filtro precedente colorerà i pixel non trasparenti con il colore blu.

Il filtro colore può essere applicato a un `Drawable` :

```
drawable.setColorFilter(filter);
```

Può essere applicato a un `ImageView` :

```
imageView.setColorFilter(filter);
```

Inoltre, può essere applicato a un `Paint` , in modo che il colore che viene disegnato usando quella vernice, venga modificato dal filtro:

```
paint.setColorFilter(filter);
```

Creazione di un XferMode PorterDuff

Un `Xfermode` (si pensi alla modalità "trasferimento") funziona come una fase di trasferimento nella pipeline di disegno. Quando un `Xfermode` viene applicato a un `Paint` , i pixel disegnati con la vernice vengono combinati con i pixel sottostanti (già disegnati) secondo la modalità:

```
paint.setColor(Color.BLUE);
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
```

Ora abbiamo una tinta blu. Qualsiasi forma disegnata colorerà i pixel già esistenti non trasparenti nell'area della forma.

Applicare una maschera radiale (vignetta) a una bitmap utilizzando 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);
}
```

Leggi Modalità PorterDuff online: https://riptutorial.com/it/android/topic/377/modalita-porterduff

Capitolo 159: Modelli di progettazione

introduzione

I modelli di progettazione sono le migliori pratiche formalizzate che il programmatore può utilizzare per risolvere problemi comuni durante la progettazione di un'applicazione o di un sistema.

I modelli di progettazione possono accelerare il processo di sviluppo fornendo paradigmi di sviluppo collaudati e comprovati.

Riutilizzare i modelli di progettazione aiuta a prevenire problemi sottili che possono causare problemi importanti e migliora anche la leggibilità del codice per i programmatori e gli architetti che hanno familiarità con i modelli.

Examples

Esempio di classe Singleton

Java Singleton Pattern

Per implementare il modello Singleton, abbiamo approcci diversi ma tutti seguono concetti comuni.

- Costruttore privato per limitare l'istanza della classe da altre classi.
- Variabile statica privata della stessa classe che è l'unica istanza della classe.
- Metodo statico pubblico che restituisce l'istanza della classe, questo è l'accesso globale
- punta al mondo esterno per ottenere l'istanza della classe 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;
```

```
    }
}
```

Modello di osservatore

Il modello di osservatore è un modello comune, ampiamente utilizzato in molti contesti. Un vero esempio può essere preso da YouTube: quando ti piace un canale e vuoi ricevere tutte le notizie e guardare nuovi video da questo canale, devi iscriverti a quel canale. Quindi, ogni volta che questo canale pubblica notizie, tu (e tutti gli altri abbonati) riceverai una notifica.

Un osservatore avrà due componenti. Uno è un broadcaster (canale) e l'altro è un ricevitore (tu o qualsiasi altro utente). L'emittente gestirà tutte le istanze del destinatario che l'hanno sottoscritta. Quando l'emittente emette un nuovo evento, lo annuncerà a tutte le istanze del destinatario. Quando il destinatario riceve un evento, dovrà reagire a quell'evento, ad esempio attivando YouTube e riproducendo il nuovo video.

Implementare il modello di osservatore

1. L'emittente deve fornire metodi che consentano ai destinatari di iscriversi e cancellarsi da esso. Quando l'emittente emette un evento, gli abbonati devono essere avvisati che si è verificato 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. Il destinatario deve implementare un metodo che gestisca l'evento dall'emittente:

```
interface Subscriber {
    void doSubscribe(Channel channel);
    void doUnsubscribe(Channel channel);
    void handleEvent();  // Process the new event.
}
```

Leggi Modelli di progettazione online: https://riptutorial.com/it/android/topic/9949/modelli-di-progettazione

Capitolo 160: Modifica il testo

Examples

Lavorare con EditTexts

EditText è il widget di inserimento testo standard nelle app Android. Se l'utente deve inserire del testo in un'app, questo è il modo principale per farlo.

Modifica il testo

Ci sono molte proprietà importanti che possono essere impostate per personalizzare il comportamento di un EditText. Molti di questi sono elencati di seguito. Consulta la guida ai campi di testo ufficiali per ulteriori dettagli sul campo di input.

uso

Un EditText viene aggiunto a un layout con tutti i comportamenti predefiniti con il seguente XML:

```
<EditText
    android:id="@+id/et_simple"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
</EditText>
```

Nota che EditText è semplicemente un'estensione sottile di TextView e eredita tutte le stesse proprietà.

Recupero del valore

Ottenere il valore del testo inserito in un EditText è il seguente:

```
EditText simpleEditText = (EditText) findViewById(R.id.et_simple);
String strValue = simpleEditText.getText().toString();
```

Ulteriore personalizzazione dell'iscrizione

Potremmo voler limitare l'immissione a una singola riga di testo (evitare le newline):

```
<EditText
  android:singleLine="true"
  android:lines="1"
/>
```

Puoi limitare i caratteri che possono essere inseriti in un campo usando l'attributo cifre:

```
<EditText
  android:inputType="number"
  android:digits="01"
```

```
/>
```

Ciò limiterebbe le cifre inserite solo a "0" e "1". Potremmo voler limitare il numero totale di caratteri con:

```
<EditText
  android:maxLength="5"
/>
```

Usando queste proprietà possiamo definire il comportamento di input previsto per i campi di testo.

Regolazione dei colori

Puoi regolare il colore dello sfondo di evidenziazione del testo selezionato all'interno di un EditText con la proprietà `android:textColorHighlight` :

```
<EditText
    android:textColorHighlight="#7cff88"
/>
```

Visualizzazione dei suggerimenti segnaposto

È possibile che si desideri impostare il suggerimento per il controllo EditText per richiedere all'utente un input specifico con:

```
<EditText
    ...
    android:hint="@string/my_hint">
</EditText>
```

suggerimenti

Modifica del colore della linea di fondo

Supponendo che si stia utilizzando la libreria AppCompat, è possibile sovrascrivere gli stili colorControlNormal, colorControlActivated e 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>
```

Se non vedi questi stili applicati all'interno di un DialogFragment, c'è un bug noto quando si usa il LayoutInflater passato al metodo onCreateView ().

Il problema è già stato risolto nella libreria AppCompat v23. Vedi questa guida su come aggiornare. Un'altra soluzione temporanea consiste nell'utilizzare il gonfiaggio del layout dell'attività anziché quello passato nel metodo onCreateView ():

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = getActivity().getLayoutInflater().inflate(R.layout.dialog_fragment,
container);
  }
```

Ascoltare l'input di EditText

Scopri le note di base degli ascoltatori di eventi di base per sapere come ascoltare le modifiche a un EditText ed eseguire un'azione quando si verificano tali cambiamenti.

Visualizzazione del feedback delle etichette mobili

Tradizionalmente, EditText nasconde il messaggio di suggerimento (spiegato sopra) dopo che l'utente inizia a digitare. Inoltre, qualsiasi messaggio di errore di convalida doveva essere gestito manualmente dallo sviluppatore.

Con `TextInputLayout` è possibile impostare un'etichetta mobile per visualizzare suggerimenti e messaggi di errore. Puoi trovare maggiori dettagli qui .

Personalizzazione di InputType

I campi di testo possono avere diversi tipi di input, come numero, data, password o indirizzo e-mail. Il tipo determina il tipo di caratteri consentiti all'interno del campo e può richiedere alla tastiera virtuale di ottimizzare il layout per i caratteri utilizzati di frequente.

Per impostazione predefinita, qualsiasi contenuto di testo all'interno di un controllo `EditText` viene visualizzato come testo normale. Impostando l'attributo `inputType` , possiamo facilitare l'inserimento di diversi tipi di informazioni, come numeri di telefono e password:

```
<EditText
    ...
    android:inputType="phone">
</EditText>
```

I tipi di input più comuni includono:

genere	Descrizione
textUri	Testo che verrà utilizzato come URI
textEmailAddress	Testo che verrà utilizzato come indirizzo e-mail
textPersonName	Testo che è il nome di una persona
textPassword	Testo che è una password che dovrebbe essere oscurata
numero	Un campo solo numerico
Telefono	Per inserire un numero di telefono

genere	Descrizione
Data	Per inserire una data
tempo	Per entrare in un tempo
textMultiLine	Consentire più righe di testo nel campo

`android:inputType` consente inoltre di specificare determinati comportamenti della tastiera, ad esempio la possibilità di utilizzare tutte le nuove parole in maiuscolo o utilizzare funzioni come i suggerimenti di completamento automatico e di ortografia.
Ecco alcuni dei valori dei tipi di input comuni che definiscono i comportamenti della tastiera:

genere	Descrizione
textCapSentences	Tastiera di testo normale che capitalizza la prima lettera per ogni nuova frase
textCapWords	Tastiera di testo normale che capitalizza ogni parola. Buono per titoli o nomi di persone
textAutoCorrect	Tastiera di testo normale che corregge le parole comunemente errate

È possibile impostare più attributi `inputType` se necessario (separati da '|').
Esempio:

```
<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" />
```

Puoi vedere un elenco di tutti i tipi di input disponibili qui .

attributo `inputype`

attributo `inputype` nel widget `EditText` : *(testato su Android 4.4.3 e 2.3.3)*

```
<EditText android:id="@+id/et_test" android:inputType="?????"/>
```

textLongMessage = Tastiera: alfabeto / predefinito. Tasto Invio: Invia / Avanti. Emozione: sì Caso: minuscolo. Suggerimento: sì. Inserisci. carbonizzazione: **e.** e ogni cosa

textFilter = Tastiera: alfabeto / predefinito. Tasto Invio: Invia / Avanti. Emozione: sì Caso: minuscolo. **Suggerimento: no** . Inserisci. carbonizzazione: **e.** e ogni cosa

textCapWords = Tastiera: alfabeto / predefinito. Tasto Invio: Invia / Avanti. Emozione: sì **Caso:**

custodia cammello . Suggerimento: sì. Inserisci. carbonizzazione: **e.** e ogni cosa

textCapSentences = Tastiera: alfabeto / predefinito. Tasto Invio: Invia / Avanti. Emozione: sì **Caso: caso di sentenza** . Suggerimento: sì. Inserisci. carbonizzazione: **e.** e ogni cosa

tempo = tastiera: numerico. Tasto Invio: Invia / Avanti. Emozione: no. Astuccio: -. **Suggerimento: no** . Inserisci. caratteri:

textMultiLine = Tastiera: alfabeto / predefinito. **Tasto Invio: accanto** . Emozione: sì Caso: minuscolo. Suggerimento: sì. Inserisci. carbonizzazione: **e.** e ogni cosa

numero = Tastiera: numerico . Tasto Invio: Invia / Avanti. Emozione: no. Astuccio: -. Suggerimento: no. **Inserisci. chars: niente**

textEmailAddress = Tastiera: alfabeto / predefinito. Tasto Invio: Invia / Avanti. **Emozione: no** . Caso: minuscolo. **Suggerimento: no** . Inserisci. caratteri: @ e **.** e ogni cosa

(Nessun tipo) = Tastiera: alfabeto / predefinito. **Tasto Invio: accanto** . Emozione: sì Caso: minuscolo. Suggerimento: sì. Inserisci. carbonizzazione: **e.** e ogni cosa

textPassword = Tastiera: alfabeto / predefinito. Tasto Invio: Invia / Avanti. Emozione: no. Caso: minuscolo. **Suggerimento: no** . Inserisci. carbonizzazione: **e.** e ogni cosa

text = Tastiera: Tastiera: alfabeto / predefinito. Tasto Invio: Invia / Avanti. Emozione: sì Caso: minuscolo. Suggerimento: sì. Inserisci. carbonizzazione: **e.** e ogni cosa

textShortMessage = Tastiera: alfabeto / predefinito. **Tasto Invio: emozione** . Emozione: sì Caso: minuscolo. Suggerimento: sì. Inserisci. carbonizzazione: **e.** e ogni cosa

textUri = Tastiera: alfabeto / predefinito. Tasto Invio: Invia / Avanti. Emozione: no. Caso: minuscolo. **Suggerimento: no** . Inserisci. caratteri: **/** e **.** e ogni cosa

textCapCharacters = Tastiera: alfabeto / predefinito. Tasto Invio: Invia / Avanti. Emozione: sì **Caso: MAIUSCOLE** . Suggerimento: sì. Inserisci. carbonizzazione: **e.** e ogni cosa

telefono = Tastiera: numerico . Tasto Invio: Invia / Avanti. Emozione: no. Astuccio: -. **Suggerimento: no** . Inserisci. caratteri: *** #. - / () WPN, + **

textPersonName = Tastiera: alfabeto / predefinito. Tasto Invio: Invia / Avanti. Emozione: sì Caso: minuscolo. Suggerimento: sì. Inserisci. carbonizzazione: **e.** e ogni cosa

Nota: l'impostazione di `Auto-capitalization` cambierà il comportamento predefinito.

Nota 2: nella `Numeric keyboard` , tutti i numeri sono in inglese 1234567890.

Nota 3: l'impostazione `Correction/Suggestion` cambierà il comportamento predefinito.

Nascondere SoftKeyboard

Nascondere Softkeyboard è un **requisito di base di** solito quando si lavora con EditText. Il softkey per *impostazione predefinita* può essere chiuso solo premendo il pulsante Indietro e così la maggior parte degli sviluppatori utilizza InputMethodManager per forzare Android a nascondere la tastiera virtuale chiamante hideSoftInputFromWindow e passando il token della finestra contenente la vista focalizzata. Il codice per eseguire quanto segue:

```
public void hideSoftKeyboard()
{
        InputMethodManager inputMethodManager = (InputMethodManager)
getSystemService(Activity.INPUT_METHOD_SERVICE);
        inputMethodManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(), 0);
}
```

Il codice è diretto, ma un altro problema importante che si presenta è che la funzione hide deve essere chiamata quando si verifica qualche evento. Cosa fare quando hai bisogno della Softkeyboard nascosta premendo su un punto diverso dal tuo EditText? Il seguente codice fornisce una funzione ordinata che deve essere chiamata nel tuo metodo onCreate () solo una volta.

```
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);
            }
        }
}
```

Icona o pulsante all'interno di Custom Edit Text e la sua azione e click listers.

Questo esempio aiuterà ad avere il testo Modifica con l'icona sul lato destro.

> Nota: In questo sto usando setCompoundDrawablesWithIntrinsicBounds, quindi se vuoi cambiare la posizione dell'icona puoi ottenerlo usando setCompoundDrawablesWithIntrinsicBounds in 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);
    }
}
```

Se si desidera modificare l'area di tocco, è possibile modificare i valori predefiniti di EXTRA_TOUCH_AREA come 50.

E per abilitare il pulsante e fare clic sul listener puoi chiamare dalla tua attività o frammento come questo,

```
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.

            }
        });
```

Leggi Modifica il testo online: https://riptutorial.com/it/android/topic/5843/modifica-il-testo

Capitolo 161: Modifiche all'orientamento

Osservazioni

Riferimento: https://guides.codepath.com/android/Handling-Configuration-Changes#references

Examples

Salvataggio e ripristino dello stato delle attività

Quando l'attività inizia a fermarsi, il sistema chiama `onSaveInstanceState()` modo che l'attività possa salvare le informazioni di stato con un insieme di coppie chiave-valore. L'implementazione predefinita di questo metodo salva automaticamente le informazioni sullo stato della gerarchia di visualizzazione dell'attività, ad esempio il testo in un widget `EditText` o la posizione di scorrimento di un controllo `ListView` .

Per salvare informazioni di stato aggiuntive per la tua attività, devi implementare `onSaveInstanceState()` e aggiungere coppie chiave-valore all'oggetto Bundle. Per esempio:

```
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);
    }
}
```

Il sistema chiamerà quel metodo prima che un'Attività venga distrutta. Successivamente il sistema chiamerà `onRestoreInstanceState` dove possiamo ripristinare lo stato dal bundle:

```
@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);
}
```

Lo stato di istanza può anche essere ripristinato nel metodo di attività # onCreate standard, ma è conveniente farlo in `onRestoreInstanceState` che assicura che tutta l'inizializzazione sia stata eseguita e consente alle sottoclassi di decidere se utilizzare l'implementazione predefinita. Leggi questo post StackOverflow per i dettagli.

Notare che `onSaveInstanceState` e `onRestoreInstanceState` non sono garantiti per essere chiamati insieme. Android invoca `onSaveInstanceState()` quando esiste la possibilità che l'attività possa essere distrutta. Tuttavia, ci sono casi in cui viene chiamato `onSaveInstanceState` ma l'attività non viene distrutta e come risultato non viene richiamato `onRestoreInstanceState` .

Salvataggio e ripristino dello stato dei frammenti

I frammenti hanno anche un metodo `onSaveInstanceState()` che viene chiamato quando il loro stato deve essere salvato:

```
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);
    }
}
```

Quindi possiamo estrarre i dati da questo stato salvato in `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;
   }
}
```

Affinché lo stato dei frammenti sia salvato correttamente, dobbiamo essere sicuri che non stiamo ricreando inutilmente il frammento sulle modifiche di configurazione. Ciò significa fare attenzione a non reinizializzare i frammenti esistenti quando già esistono. Eventuali frammenti inizializzati in un'attività devono essere ricercati per tag dopo una modifica alla configurazione:

```
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();
        }
    }
}
```

Questo ci impone di fare attenzione a includere un tag per la ricerca ogni volta che si inserisce un frammento nell'attività all'interno di una transazione:

```
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 questo semplice schema, possiamo riutilizzare correttamente i frammenti e ripristinare il loro stato attraverso le modifiche alla configurazione.

Frammenti di sostegno

In molti casi, possiamo evitare problemi quando un'attività viene ricreata semplicemente usando i frammenti. Se le viste e lo stato sono all'interno di un frammento, possiamo facilmente conservare il frammento quando l'attività viene ricreata:

```
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;
    }
```

```
}
```

Questo approccio impedisce che il frammento venga distrutto durante il ciclo di vita dell'attività. Sono invece trattenuti all'interno di Fragment Manager. Vedi i documenti ufficiali di Android per ulteriori informazioni .

Ora puoi verificare se il frammento esiste già per tag prima di crearne uno e il frammento manterrà lo stato tra le modifiche di configurazione. Per ulteriori dettagli, consultare la guida Gestione delle modifiche di runtime.

Orientamento dello schermo di blocco

Se vuoi bloccare il cambio di orientamento dello schermo di qualsiasi schermo (attività) della tua applicazione Android devi solo impostare la proprietà `android:screenOrientation` di una `<activity>` all'interno di **AndroidManifest.xml** :

```
<activity
    android:name="com.techblogon.screenorientationexample.MainActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
    <!-- ... -->
</activity>
```

Ora quell'attività è obbligata a essere sempre visualizzata in modalità " **ritratto** ".

Gestire manualmente le modifiche di configurazione

Se l'applicazione non ha bisogno di aggiornare le risorse durante una specifica modifica alla configurazione e si dispone di una limitazione delle prestazioni che richiede di evitare il riavvio delle attività, è possibile dichiarare che la propria attività gestisce la stessa modifica della configurazione, impedendo al sistema di riavviare il proprio attività.

Tuttavia, questa tecnica deve essere considerata l'ultima risorsa quando è necessario evitare riavvii a causa di una modifica della configurazione e non è consigliata per la maggior parte delle applicazioni. Per adottare questo approccio, dobbiamo aggiungere il nodo `android:configChanges` all'attività all'interno di **AndroidManifest.xml** :

```
<activity android:name=".MyActivity"
          android:configChanges="orientation|screenSize|keyboardHidden"
          android:label="@string/app_name">
```

Ora, quando una di queste configurazioni cambia, l'attività non si riavvia ma riceve una chiamata 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();
    }
}
```

Vedi la gestione dei documenti di modifica . Per ulteriori informazioni su quali modifiche alla configurazione è possibile gestire nella propria attività, consultare la documentazione di android: configChanges e la classe Configuration .

Gestire AsyncTask

Problema:

- Se dopo l'avvio di `AsyncTask` c'è una rotazione dello schermo, l'attività proprietaria viene distrutta e ricreata.
- Quando termina `AsyncTask` , vuole aggiornare l'interfaccia utente che potrebbe non essere più valida.

Soluzione:

Usando i caricatori , si può facilmente superare l'attività di distruzione / ricreazione.

Esempio:

Attività principale:

```
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:

È importante utilizzare o meno la libreria di compatibilità v4, ma non utilizzare parte di una parte e una parte dell'altro, poiché ciò porterà a errori di compilazione. Per controllare puoi guardare le importazioni per `android.support.v4.content` e `android.content` (non dovresti averle entrambe).

Blocca la rotazione dello schermo a livello di programmazione

È molto comune che durante lo sviluppo, si possa trovare **molto utile per bloccare / sbloccare lo schermo del dispositivo durante specifiche parti del codice** .

*Ad esempio, mentre mostra una finestra di dialogo con informazioni, lo sviluppatore potrebbe voler **bloccare** la rotazione dello schermo per impedire che la finestra di dialogo venga **chiusa** e l'attività corrente venga ricostruita per **sbloccarla di** nuovo quando la finestra di dialogo viene chiusa.*

Anche se possiamo ottenere il blocco della rotazione dal manifest facendo:

```
<activity
    android:name=".TheActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
</activity>
```

Si può farlo anche programmaticamente facendo quanto segue:

```
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);
        }
    }
}
```

E poi chiamando il seguente, rispettivamente per bloccare e sbloccare la rotazione del dispositivo

```
lockDeviceRotation(true)
```

e

```
lockDeviceRotation(false)
```

Leggi Modifiche all'orientamento online: https://riptutorial.com/it/android/topic/4621/modifiche-all-orientamento

Capitolo 162: Modo rapido per impostare Retrolambda su un progetto Android.

introduzione

Retrolambda è una libreria che consente di utilizzare espressioni lambda Java 8, riferimenti al metodo e istruzioni try-with-resources su Java 7, 6 o 5.

Il plug-in Gradle Retrolambda consente di integrare Retrolambda in una build basata su Gradle. Ciò consente ad esempio di utilizzare questi costrutti in un'applicazione Android, poiché attualmente lo sviluppo Android standard non supporta ancora Java 8.

Examples

Installazione ed esempio come usare:

Passaggi di installazione:

1. Scarica e installa jdk8.
2. Aggiungi quanto segue al file build.gradle principale del tuo progetto

```
buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
        classpath 'me.tatarka:gradle-retrolambda:3.2.3'
    }
}
```

3. Ora aggiungi questo al build.gradle del modulo dell'applicazione

```
apply plugin: 'com.android.application' // or apply plugin: 'java'
apply plugin: 'me.tatarka.retrolambda'
```

4. Aggiungi queste righe al build.gradle del modulo dell'applicazione per informare l'IDE del livello linguistico:

```
android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

Esempio:

Quindi cose come questa:

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        log("Clicked");
    }
});
```

Diventa questo:

```
button.setOnClickListener(v -> log("Clicked"));
```

Leggi Modo rapido per impostare Retrolambda su un progetto Android. online: https://riptutorial.com/it/android/topic/8822/modo-rapido-per-impostare-retrolambda-su-un-progetto-android-

Capitolo 163: Moshi

introduzione

Moshi è una moderna libreria JSON per Android e Java. Semplifica l'analisi di JSON in oggetti Java e Java in JSON.

Osservazioni

Non dimenticare, leggi sempre il README !

Examples

JSON in 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);
```

serializzare oggetti Java come 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);
```

Adattatori di tipo integrato

Moshi ha il supporto integrato per leggere e scrivere i principali tipi di dati di Java:

- Primitive (int, float, char ...) e le loro controparti in scatola (Integer, Float, Character ...).
- Array
- collezioni
- elenchi
- Imposta
- Maps Strings Enums

Supporta le classi del modello scrivendole campo per campo. Nell'esempio sopra Moshi usa

queste classi:

```
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"
    }
  ]
}
```

Leggi Moshi online: https://riptutorial.com/it/android/topic/8744/moshi

Capitolo 164: Multidex e il limite del metodo Dex

introduzione

DEX indica i file bytecode eseguibili dell'app (APK) di Android sotto forma di file Dalvik Executable (DEX), che contengono il codice compilato utilizzato per eseguire l'app.

La specifica eseguibile Dalvik limita il numero totale di metodi che possono essere referenziati all'interno di un singolo file DEX a 65.536 (64K), inclusi i metodi di framework Android, i metodi di libreria ei metodi nel proprio codice.

Per superare questo limite è necessario configurare il processo di creazione dell'app per generare più di un file DEX, noto come Multidex.

Osservazioni

Cos'è il dex?

Dex è il nome del formato file e della codifica a cui è stato compilato il codice Java di Android. Le versioni precedenti di Android caricavano ed eseguivano i binari `dex` direttamente in una macchina virtuale chiamata Dalvik. Le versioni più recenti di Android utilizzano Android Runtime (ART), che tratta i file `dex` come una rappresentazione intermedia ed esegue ulteriori compilazioni su di esso prima di eseguire l'applicazione.

Dex è un formato di file molto vecchio, in termini di durata di vita degli smartphone, ed è stato progettato per dispositivi la cui memoria principale è stata misurata in decine di megabyte. I limiti di progettazione di quei giorni sono rimasti con noi fino ad oggi.

Il problema:

Il formato del file `dex` codifica un limite al numero di metodi a cui è possibile fare riferimento in un singolo binario. Poiché la porzione del formato file che memorizza il numero di riferimenti è lunga due byte, il numero massimo di riferimenti al metodo è `0xFFFF` o 65535. Se un'applicazione contiene più di quel numero di riferimenti al metodo, non riuscirà a compilare.

Cosa fare al riguardo:

Google ha fornito un modo per aggirare questo problema, chiamato Multidex. Ha componenti in fase di compilazione e in fase di esecuzione. Come suggerisce il nome, in fase di compilazione dividerà il codice tra uno o più file `dex` . Al runtime, insegnerà al `ClassLoader` predefinito come cercare le classi da questi file.

Questo approccio funziona bene su dispositivi più recenti, ma presenta alcuni inconvenienti sostanziali. Può aumentare drasticamente il tempo di avvio delle applicazioni e sui dispositivi più vecchi può causare errori di `Application Not Responding` .

Multidex, se efficace, dovrebbe essere evitato se possibile.

Come evitare il limite:

Prima di configurare la tua app per abilitare l'uso di 64K o più riferimenti al metodo, dovresti prendere provvedimenti per ridurre il numero totale di riferimenti chiamati dal tuo codice app, compresi i metodi definiti dal codice dell'app o dalle librerie incluse. Le seguenti strategie possono aiutarti a evitare di colpire il limite di riferimento di dex:

- Verifica le **dipendenze dirette e transitive della tua app** : assicurati che qualsiasi dipendenza della libreria di grandi dimensioni inclusa nella tua app sia utilizzata in un modo che superi la quantità di codice che viene aggiunta all'applicazione. Un anti-pattern comune consiste nell'includere una libreria molto grande perché alcuni metodi di utilità erano utili. La riduzione delle dipendenze del codice dell'app può spesso aiutare a evitare il limite di riferimento di dex.
- **Rimuovi il codice inutilizzato con ProGuard** : configura le impostazioni di ProGuard per la tua app per eseguire ProGuard e assicurati di avere il restringimento abilitato per i build di rilascio. L'attivazione del restringimento garantisce che non vengano spediti codici inutilizzati con gli APK.

Il primo punto richiede diligenza e disciplina da parte dello sviluppatore. Quando si incorporano librerie di terze parti, è necessario considerare la dimensione della libreria. Ad esempio, due famose librerie JSON sono Jackson e Gson. Funzionalmente sono abbastanza simili, ma Gson tende a vedere un maggiore uso in Android. Una delle ragioni è che Jackson pesa circa 9.000 metodi, mentre Gson contribuisce con 1.900.

Sono disponibili diversi strumenti per aiutare gli sviluppatori a tenere traccia delle dimensioni della loro applicazione:

- dexcount-gradle-plugin riporta il numero di riferimenti al metodo nel tuo APK o AAR su ogni build
- dex-method-counts è uno strumento della riga di comando che conta il numero di riferimenti al metodo in un APK
- www.methodscount.com è un servizio web che conteggia i riferimenti al metodo in qualsiasi APK che carichi.

Examples

Multidex utilizzando direttamente MultiDexApplication

Usa questa opzione se non hai bisogno di una sottoclasse `Application` .

Questa è l'opzione più semplice, ma in questo modo non è possibile fornire la propria sottoclasse

`Application` . Se è necessaria una sottoclasse di `Application` , sarà necessario passare a una delle altre opzioni per farlo.

Per questa opzione, è sufficiente specificare il nome di classe completo `android.support.multidex.MultiDexApplication` per `android:name` proprietà `android:name` del tag `application` in 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 estendendo l'applicazione

Utilizzare questa opzione se il progetto richiede una sottoclasse `Application` .

Specificare questa sottoclasse `Application` utilizzando la proprietà `android:name` nel file manifest all'interno del tag `application` .

Nella sottoclasse `Application` , aggiungi la sovrascrittura del metodo `attachBaseContext()` , e in tale metodo chiama `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);
    }
}
```

Assicurati che la sottoclasse `Application` sia specificata nel tag `application` di AndroidManifest.xml:

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

Abilitazione di Multidex

Per abilitare una configurazione multidex è necessario:

- per cambiare la configurazione di build di Gradle
- utilizzare una `MultiDexApplication` o abilitare il MultiDex nella classe `Application`

Configurazione gradle

In `app/build.gradle` aggiungi queste parti:

```
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'
}
```

Abilita MultiDex nella tua applicazione

Quindi procedere con una delle tre opzioni:

- Multidex estendendo l'applicazione
- Multidex estendendo MultiDexApplication
- Multidex utilizzando direttamente MultiDexApplication

Quando queste impostazioni di configurazione vengono aggiunte a un'app, gli strumenti di build di Android costruiscono un dex primario (classes.dex) e supportano (classes2.dex, classes3.dex) secondo necessità.
Il sistema di compilazione li impacchetterà quindi in un file APK per la distribuzione.

Riferimenti del metodo di conteggio su ogni build (plug-in Dexcount Gradle)

Il plugin dexcount conta i metodi e il conteggio delle risorse di classe dopo una build di successo.

Aggiungi il plugin `app/build.gradle` :

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

```
buildscript {
    repositories {
        mavenCentral() // or jcenter()
    }

    dependencies {
        classpath 'com.getkeepsafe.dexcount:dexcount-gradle-plugin:0.5.5'
    }
}
```

Applicare il plug-in nel file `app/build.gradle` :

```
apply plugin: 'com.getkeepsafe.dexcount'
```

Cerca i dati di output generati dal plugin in:

../app/build/outputs/dexcount

Particolarmente utile è il grafico .html in:

../app/build/outputs/dexcount/debugChart/index.html

Multidex estendendo MultiDexApplication

Questo è molto simile all'utilizzo di una sottoclasse `Application` e l'override del metodo `attachBaseContext()` .

Tuttavia, utilizzando questo metodo, non è necessario eseguire l'override di `attachBaseContext()` poiché ciò è già stato fatto nella superclasse `MultiDexApplication` .

Estendi `MultiDexApplication` anziché `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()

     //.........
}
```

Aggiungi questa classe al tuo AndroidManifest.xml esattamente come se stessi estendendo l'applicazione:

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
```

```
    android:label="@string/app_name">
</application>
```

Leggi Multidex e il limite del metodo Dex online:
https://riptutorial.com/it/android/topic/1887/multidex-e-il-limite-del-metodo-dex

Capitolo 165: MVVM (Architettura)

Osservazioni

Sintassi stranezze con DataBinding

Quando si associa una funzione viewModel a una proprietà in xml, determinati prefissi di funzione come `get` o `is` vengono rilasciati. Per esempio. `ViewModel::getFormattedText` sul ViewModel diventerà `@{viewModel.formattedText}` quando lo si lega a una proprietà in xml. Analogamente con `ViewModel::isContentVisible` -> `@{viewModel.contentVisible}` (notazione Java Bean)

Le classi di bind generate come `ActivityMainBinding` prendono il nome dal xml per cui stanno creando i collegamenti per, non la classe java.

Binding personalizzati

In activity_main.xml ho impostato l'attributo textColor `app` e non lo spazio dei nomi di `android` . Perché? Poiché esiste un setter personalizzato definito per l'attributo `textColor` che risolve un ID risorsa ColorRes inviato da ViewModel a un colore effettivo.

```
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);
    }

}
```

Per i dettagli su come funziona controlla DataBinding Library: Custom Setters

Aspetta ... c'è una logica nel tuo xml !!!?

Si potrebbe argomentare che le cose che faccio in xml per `android:visibility` e `app:textColor` sono errate / anti-pattern nel contesto MVVM perché c'è una logica di visualizzazione nella mia vista. Tuttavia, direi che per me è più importante mantenere le dipendenze Android dal mio ViewModel per motivi di test.

Inoltre, cosa fa realmente `app:textColor` ? Risolve solo un puntatore di risorse per il colore effettivo

associato ad esso. Quindi il ViewModel decide ancora quale colore viene mostrato in base ad alcune condizioni.

Per quanto riguarda l' `android:visibility` che sento a causa di come viene chiamato il metodo, in realtà è ok per usare qui l'operatore ternario. Poiché il nome `isLoadingVisible` e `isContentVisible` non vi è alcun dubbio su cosa dovrebbe risolvere ogni risultato nella vista. Quindi ritengo che sia piuttosto l'esecuzione di un comando dato da ViewModel piuttosto che fare una logica di visualizzazione.

D'altra parte sono d'accordo che l'uso di `viewModel.isLoading ? View.VISIBLE : View.GONE` sarebbe una brutta cosa da fare perché stai facendo delle supposizioni nella vista che cosa significa questo stato per la vista.

Materiale utile

Le seguenti risorse mi hanno aiutato molto nel cercare di capire questo concetto:

- Jeremy Likness - Model-View-ViewModel (MVVM) Explained (C #) (08.2010)
- Shamlia Shukkur - Capire le basi del modello di progettazione MVVM (C #) (03.2013)
- Frode Nilsen - Android Databinding: addio presentatore, Ciao ViewModel! (07.2015)
- Joe Birch - Avvicinarsi ad Android con MVVM (09.2015)
- Florina Muntenescu - Modelli di architettura Android Parte 3: Model-View-ViewModel (10.2016)

Examples

Esempio MVVM utilizzando la libreria DataBinding

L'intero punto di MVVM consiste nel separare i livelli contenenti la logica dal livello di vista.

Su Android possiamo usare la libreria DataBinding per aiutarci con questo e rendere la maggior parte della nostra unità logica testabile senza preoccuparci delle dipendenze di Android.

In questo esempio mostrerò i componenti centrali per un'app semplice e stupida che fa quanto segue:

- All'avvio simula una chiamata di rete e mostra uno spinner di caricamento
- Mostra una vista con un contatore dei clic TextView, un messaggio TextView e un pulsante per incrementare il contatore
- Sul contatore dei clic aggiorna contatore e aggiorna il colore del contatore e il testo del messaggio se il contatore raggiunge un numero

Iniziamo con il livello vista:

`activity_main.xml` :

Se non hai familiarità con il funzionamento di DataBinding dovresti probabilmente impiegare 10 minuti per familiarizzarti con esso. Come puoi vedere, tutti i campi che dovresti aggiornare con i setter sono legati alle funzioni della variabile viewModel.

Se hai una domanda su `android:visibility` o `app:textColor` proprietà `app:textColor` controlla la sezione "Note".

```
<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>
```

Successivo il livello del modello. Qui ho:

- due campi che rappresentano lo stato dell'app
- getter per leggere il numero di clic e lo stato di eccitazione
- un metodo per incrementare il conteggio dei clic

- un metodo per ripristinare alcuni stati precedenti (importante per le modifiche di orientamento)

Inoltre definisco qui uno 'stato di eccitazione' che dipende dal numero di clic. Questo verrà in seguito utilizzato per aggiornare il colore e il messaggio sulla vista.

È importante notare che non ci sono ipotesi fatte nel modello su come lo stato potrebbe essere visualizzato all'utente!

`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;
        }
    }
}
```

Avanti il ViewModel.

Ciò attiverà le modifiche sul modello e formatterà i dati dal modello per mostrarli sulla vista. Si noti che è qui dove si valuta quale rappresentazione della GUI è appropriata per lo stato dato dal modello (`resolveCounterColor` e `resolveLabelText`). Quindi, per esempio, potremmo facilmente implementare un `UnderachieverClickerModel` che ha soglie più basse per lo stato di eccitazione senza toccare alcun codice nel viewModel o nella vista.

Si noti inoltre che ViewModel non contiene alcun riferimento per visualizzare oggetti. Tutte le proprietà sono associate tramite le annotazioni `@Bindable` e aggiornate quando `notifyChange()` (segnala che tutte le proprietà devono essere aggiornate) o `notifyPropertyChanged(BR.propertyName)` (segnala che queste proprietà devono essere aggiornate).

`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;
        }
    }

}
```

Legando tutto insieme nell'attività!

Qui vediamo la vista di inizializzare viewModel con tutte le dipendenze di cui potrebbe aver bisogno, che devono essere istanziate da un contesto Android.

Dopo che viewModel è inizializzato, è associato al layout xml tramite DataBindingUtil (si prega di controllare la sezione 'Sintassi' per la denominazione delle classi generate).

Gli abbonamenti alle note sono sottoscritti su questo livello perché dobbiamo gestire l'annullamento dell'iscrizione quando l'attività viene messa in pausa o distrutta per evitare perdite di memoria e NPE. Qui viene attivato anche il persistere e il ricaricamento della viewState su 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;
    }
}
```

Per vedere tutto in azione controlla questo esempio di progetto .

Leggi MVVM (Architettura) online: https://riptutorial.com/it/android/topic/7549/mvvm--architettura-

Capitolo 166: NavigationView

Osservazioni

NavigationView rappresenta un menu di navigazione standard per l'applicazione. Il contenuto del menu può essere popolato da un file di risorse del menu.

Prima di utilizzare `NavigationView` è necessario aggiungere la dipendenza della libreria di supporto alla progettazione nel file build.gradle:

```
 dependencies {
    compile 'com.android.support:design:24.2.0'
}
```

Documentazione ufficiale:

https://developer.android.com/reference/android/support/design/widget/NavigationView.html

Specifiche di progettazione materiale:

https://material.google.com/patterns/navigation-drawer.html#navigation-drawer-content

Examples

Come aggiungere NavigationView

Per utilizzare NavigationView basta aggiungere la dipendenza nel file `build.gradle` come descritto nella sezione commenti

Quindi aggiungi `NavigationView` nel layout

```
<?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 che verrà visualizzata nella parte superiore del cassetto

```
<?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 Un livello di astrazione per la barra degli strumenti per separarlo dal contenuto:

```
<?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 Il vero contenuto dell'attività solo per la demo, qui puoi inserire il tuo normale layout xml:

```
<?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>
```

Definisci il tuo file di menu come *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
```

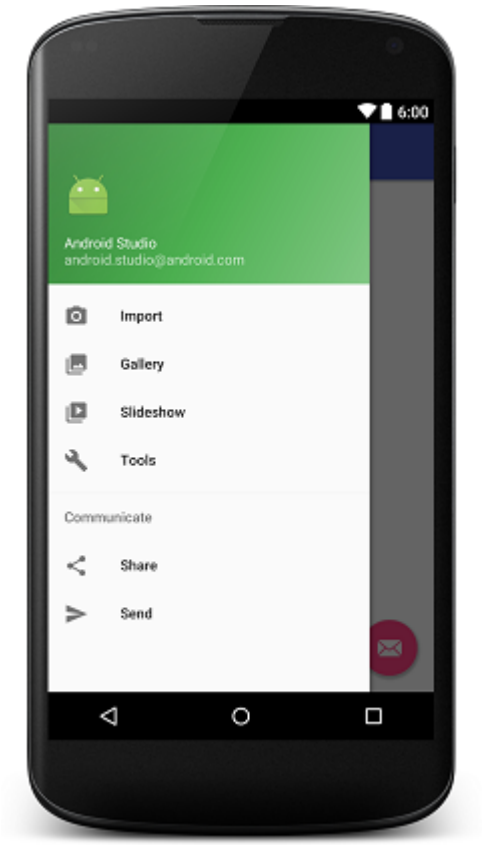
```
            android:id="@+id/nav_camera"
            android:icon="@drawable/ic_menu_camera"
            android:title="Import" />
        <item
            android:id="@+id/nav_gallery"
            android:icon="@drawable/ic_menu_gallery"
            android:title="Gallery" />
        <item
            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>
```

E infine `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;
    }
}
```

Sembrerà così:

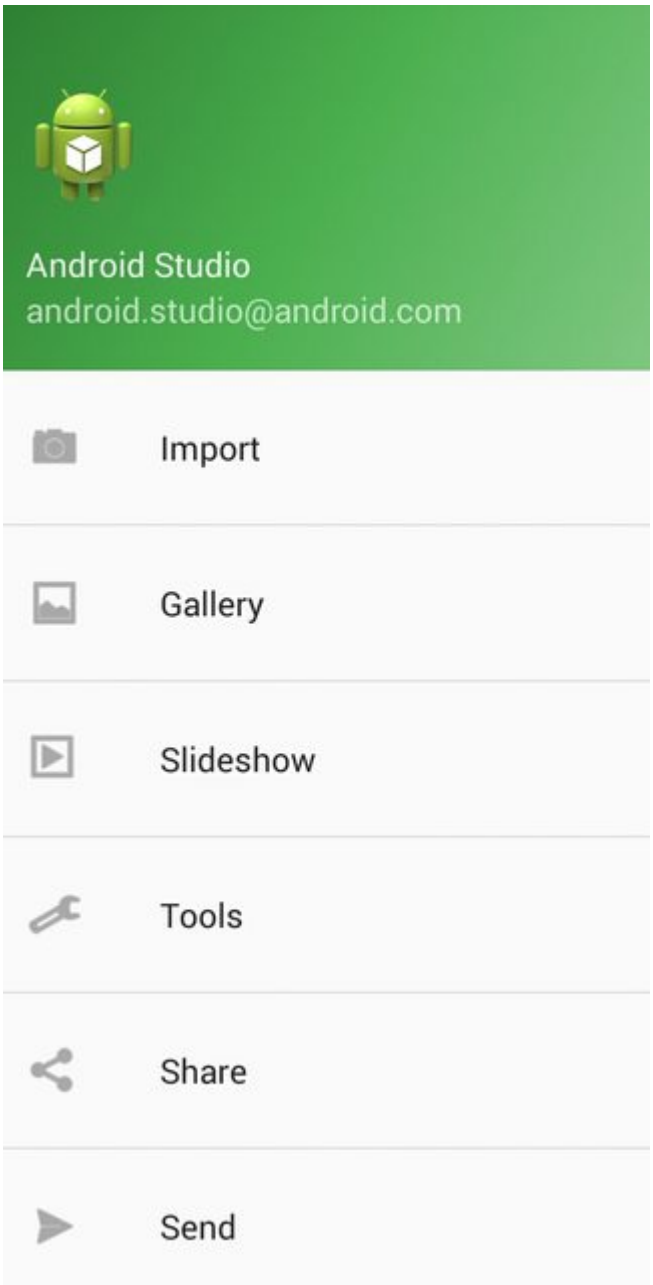


Aggiungi sottolineatura negli elementi del menu

Ogni gruppo termina con un separatore di riga. Se ogni voce del tuo menu ha il suo gruppo, otterrai l'output grafico desiderato. Funzionerà solo se i tuoi diversi gruppi hanno `android:id` diverso `android:id` . Inoltre, nel `menu.xml` ricordarsi di menzionare `android:checkable="true"` per singolo elemento e `android:checkableBehavior="single"` per un gruppo di elementi.

```
<?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>
```



Aggiungi separatori al menu

Accedi a RecyclerView in NavigationView e aggiungi ItemDecoration ad esso.

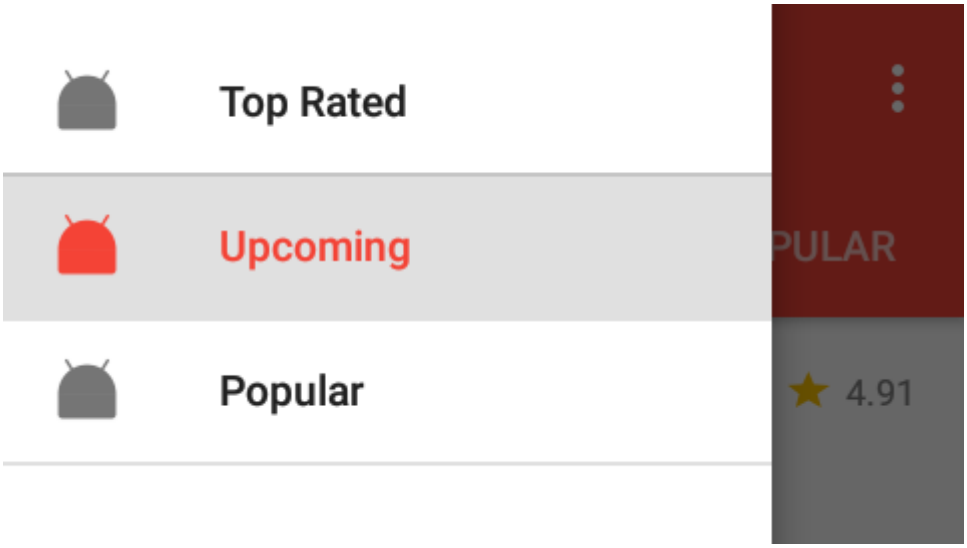
```
NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new DividerItemDecoration(this));
```

Codice per 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);
        }
    }
}
```

Anteprima:



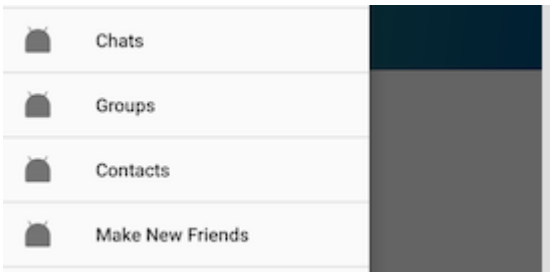
Aggiungi menu Divider usando DividerItemDecoration predefinito.

Basta usare la classe DividerItemDecoration predefinita:

```
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new
```

```
DividerItemDecoration(context,DividerItemDecoration.VERTICAL));
```

Anteprima :



Leggi NavigationView online: https://riptutorial.com/it/android/topic/97/navigationview

Capitolo 167: NDK Android

Examples

Creazione di eseguibili nativi per Android

progetto / jni / main.c

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

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

progetto / jni / Android.mk

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

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

progetto / jni / Application.mk

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

Se vuoi supportare dispositivi con versioni Android inferiori a 5.0 (API 21), devi compilare il tuo binario con `APP_PLATFORM` impostato su un'API precedente, ad esempio `android-8` . Questa è una conseguenza di Android 5.0 che applica i *BIN indipendenti dalla posizione* (PIE), mentre i dispositivi più vecchi non necessariamente supportano i PIE. Pertanto, è necessario utilizzare il PIE o il non-PIE, a seconda della versione del dispositivo. Se si desidera utilizzare il binario dall'applicazione Android, è necessario controllare il livello API ed estrarre il binario corretto.

`APP_ABI` può essere modificato su piattaforme specifiche come `armeabi` per costruire il binario solo per quelle architetture.

Nel peggiore dei casi, avrai sia un PIE che un binario non-PIE per ogni architettura (circa 14 binari diversi usando ndk-r10e).

Per costruire l'eseguibile:

```
cd project
ndk-build
```

Troverete i binari a `project/libs/<architecture>/hello_world` . Puoi utilizzarli tramite `ADB` (`push` e `chmod` con permesso eseguibile) o dalla tua applicazione (estrai e `chmod` con permesso eseguibile).

Per determinare l'architettura della CPU, recuperare la proprietà build `ro.product.cpu.abi` per l'architettura principale o `ro.product.cpu.abilist` (sui dispositivi più recenti) per un elenco completo delle architetture supportate. Puoi farlo usando la classe `android.os.Build` dall'interno dell'applicazione o usando `getprop <name>` tramite ADB.

Come pulire la build

Se hai bisogno di pulire la build:

```
ndk-build clean
```

Come utilizzare un makefile diverso da Android.mk

ndk-build NDK_PROJECT_PATH = PROJECT_PATH APP_BUILD_SCRIPT = MyAndroid.mk

Come accedere a ndk

Per prima cosa assicurati di collegarti alla libreria di logging nel tuo file `Android.mk` :

```
LOCAL_LDLIBS := -llog
```

Quindi utilizzare una delle seguenti chiamate `__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)
```

Oppure usa quelli in un modo più conveniente definendo le macro corrispondenti:

```
#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__)
```

Esempio :

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

Leggi NDK Android online: https://riptutorial.com/it/android/topic/492/ndk-android

Capitolo 168: notifiche

Examples

Creazione di una notifica semplice

Questo esempio mostra come creare una semplice notifica che avvia un'applicazione quando l'utente fa clic su di essa.

Specificare il contenuto della notifica:

```
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 l'intento di sparare al clic:

```
Intent intent = new Intent(this, MainActivity.class);
PendingIntent pi = PendingIntent.getActivity(this, 0, intent, Intent.FLAG_ACTIVITY_NEW_TASK);
mBuilder.setContentIntent(pi);
```

Infine, crea la notifica e mostrala

```
NotificationManager mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(0, mBuilder.build());
```

Avvisa la notifica con Ticker per i dispositivi più vecchi

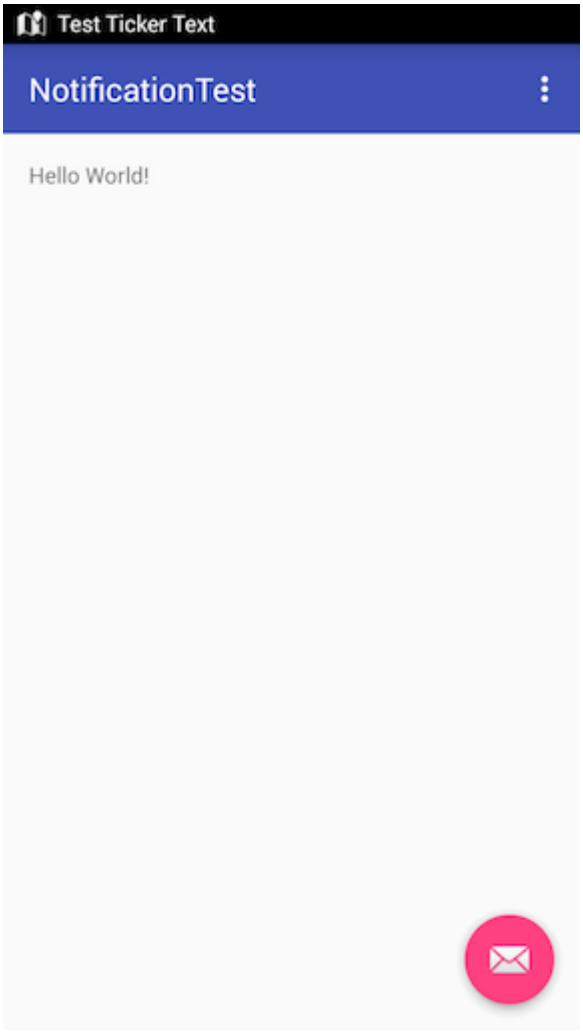
Ecco come effettuare una notifica di avviso per dispositivi abilitati e utilizzare un segno di spunta per i dispositivi più vecchi.

```
// 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());
```

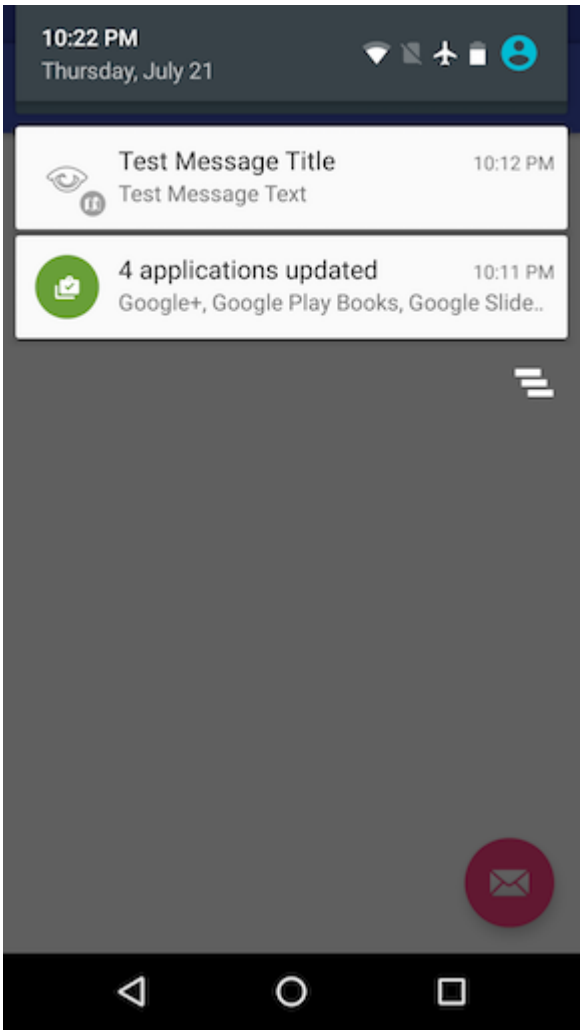
Ecco come appare su Android Marshmallow con la notifica Heads Up:

Ecco come appare su Android KitKat con il Ticker:

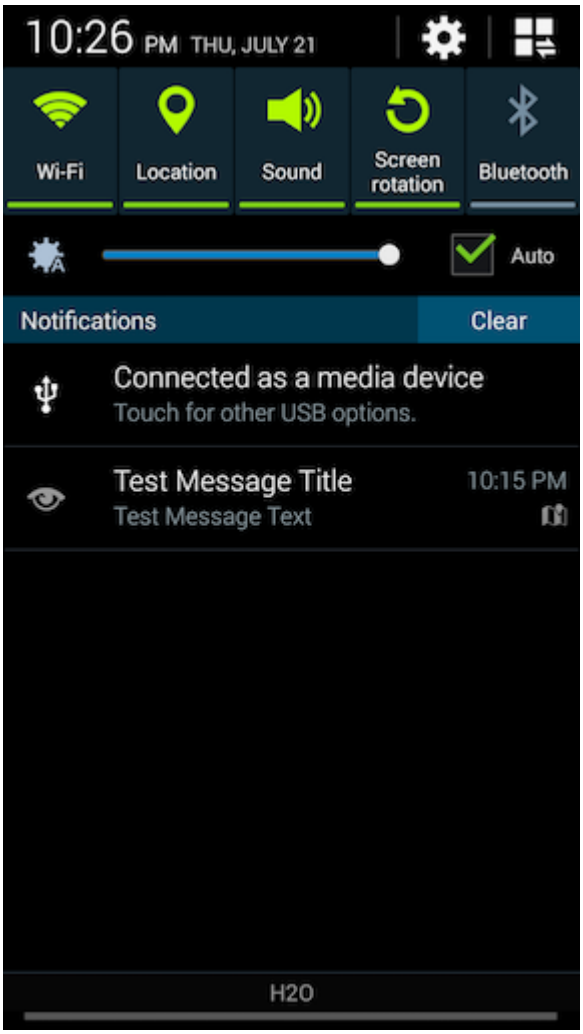


Su tutte le versioni di Android, la `Notification` viene visualizzata nel cassetto delle notifiche.

Android 6.0 Marshmallow:



Android 4.4.x KitKat:



Impostazione di priorità diverse nella notifica

```
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);
```

> Quando la notifica contiene un'immagine e si desidera espandere automaticamente l'immagine quando la notifica ricevuta utilizza "PRIORITY_MAX", è possibile utilizzare altri livelli di priorità come da requisiti

Informazioni sui livelli di priorità diversi:

PRIORITY_MAX : utilizzare per notifiche critiche e urgenti che avvisano l'utente di una condizione che è cruciale in termini di tempo o che deve essere risolta prima di poter continuare con una determinata attività.

PRIORITY_HIGH : utilizza principalmente comunicazioni importanti, come messaggi o eventi di chat con contenuti particolarmente interessanti per l'utente. Le notifiche ad alta priorità attivano la visualizzazione delle notifiche heads-up.

PRIORITY_DEFAULT - Utilizza per tutte le notifiche che non rientrano in nessuna delle altre priorità descritte qui.

PRIORITY_LOW - Utilizzare per le notifiche di cui si desidera informare l'utente, ma che sono meno urgenti. Le notifiche a bassa priorità tendono a comparire in fondo all'elenco, il che le rende una buona scelta per cose come aggiornamenti sociali pubblici o non orientati: l'utente ha chiesto di essere informato su di loro, ma queste notifiche non dovrebbero mai avere la precedenza su quelle urgenti o comunicazione diretta.

PRIORITY_MIN : utilizzare per informazioni contestuali o di background quali informazioni meteo o informazioni sulla posizione contestuale. Le notifiche con priorità minima non vengono visualizzate nella barra di stato. L'utente li scopre espandendo la tonalità di notifica.

Riferimenti: linee guida sulla progettazione dei materiali - notifiche

Pianificazione delle notifiche

A volte è necessario visualizzare una notifica in un momento specifico, un'attività che sfortunatamente non è banale sul sistema Android, in quanto non esiste alcun metodo `setTime()` o simile per le notifiche. Questo esempio delinea i passaggi necessari per pianificare le notifiche utilizzando `AlarmManager` :

1. **Aggiungi un `BroadcastReceiver`** che ascolti i messaggi di `Intent` trasmessi da Android `AlarmManager` .

Questo è il posto dove si costruisce la notifica in base agli extra forniti con l' `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. **Registrare il `BroadcastReceiver`** nel file `AndroidManifest.xml` (altrimenti il ricevitore non riceverà alcun `Intent` da `AlarmManager`):

```
<receiver
    android:name=".NotificationReceiver"
    android:enabled="true" />
```

3. **Pianificare una notifica** passando un `PendingIntent` per `BroadcastReceiver` con gli extra `Intent` necessari al sistema `AlarmManager` . Il tuo `BroadcastReceiver` riceverà l' `Intent` una volta

che l'ora è arrivata e mostrerà la notifica. Il seguente metodo pianifica una notifica:

```
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);
}
```

Si prega di notare che il `42` sopra deve essere univoco per ogni notifica programmata, altrimenti i `PendingIntent` si sostituiranno a vicenda causando effetti indesiderati!

4. **Annullare una notifica** ricostruendo il `PendingIntent` associato e cancellandolo sul sistema `AlarmManager` . Il seguente metodo annulla una notifica:

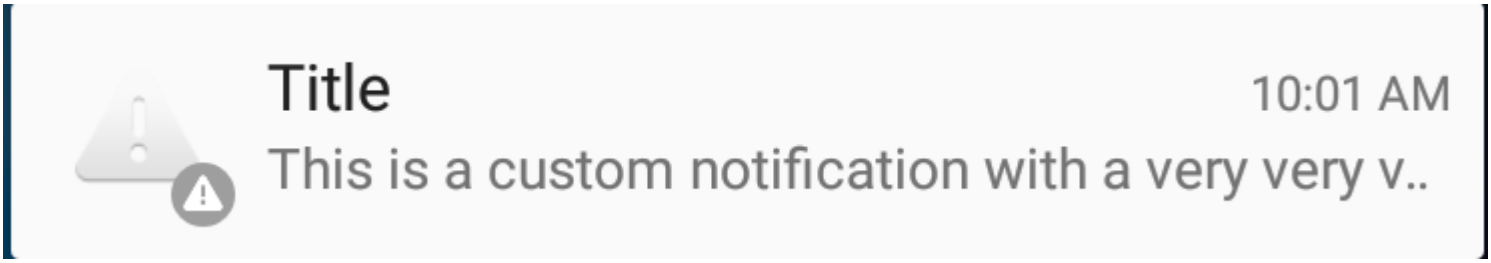
```
public static void cancelNotification(Context context, String title, String text) {
    Intent intent = new Intent(context, NotificationReceiver.class);
    intent.putExtra("title", title);
    intent.putExtra("text", text);
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
    // Cancel notification
    AlarmManager manager = (AlarmManager)
context.getSystemService(Context.ALARM_SERVICE);
    manager.cancel(pending);
}
```

Si noti che il `42` sopra deve corrispondere al numero del passaggio 3!

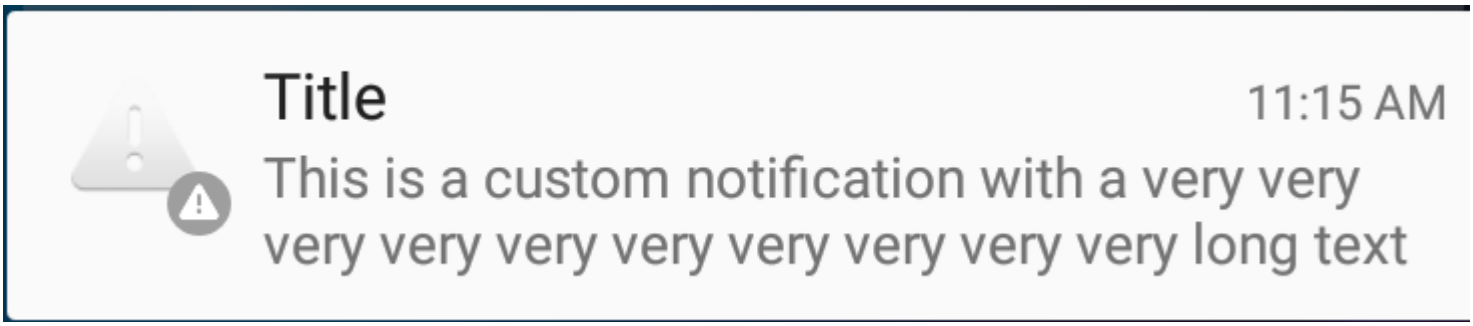
Imposta notifica personalizzata - mostra il testo del contenuto completo

Se si desidera visualizzare un testo lungo nel contesto, è necessario impostare un contenuto personalizzato.

Ad esempio, hai questo:



Ma desideri che il tuo testo sia mostrato integralmente:



Tutto quello che devi fare è **aggiungere uno stile** ai tuoi contenuti come di seguito:

```
  private void generateNotification(Context context) {
        String message = "This is a custom notification with a 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);
    }
```

Imposta l'icona di notifica personalizzata usando la libreria `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);
```

E quindi definire un layout all'interno della cartella dei layout:

```
<?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>
```

Ottenere in modo dinamico la dimensione dei pixel corretta per l'icona grande

Se stai creando un'immagine, decodificando un'immagine o ridimensionando un'immagine per adattarla all'ampia area dell'immagine di notifica, puoi ottenere le dimensioni corrette dei pixel in questo modo:

```
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);
```

Notifica in corso con il pulsante Azione

```
// 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);
```

Registrare un ricevitore di trasmissione per la stessa azione per gestire l'evento di clic del pulsante di azione.

Leggi notifiche online: https://riptutorial.com/it/android/topic/1347/notifiche

Capitolo 169: OkHttp

Examples

Registrazione dell'intercettore

`Interceptors` vengono utilizzati per intercettare `OkHttp` chiamate di `OkHttp` . La ragione per intercettare potrebbe essere quella di monitorare, riscrivere e riprovare le chiamate. Può essere utilizzato sia per la richiesta in uscita sia per la risposta in arrivo.

```
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;
  }
}
```

Risposte di riscrittura

```
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();
  }
};
```

Esempio di utilizzo di base

Mi piace avvolgere il mio `OkHttp` in una classe chiamata `HttpClient` per esempio, e in questa classe ho metodi per ciascuno dei verbi HTTP principali, `post` , `get` , `put` ed `delete` , più comunemente. (Di solito includo un'interfaccia, al fine di mantenerla per implementarla, in modo da poter facilmente passare a una diversa implementazione, se necessario):

```
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 sintassi è la stessa per `put` , `get` ed `delete` eccezione di 1 word (`.put(body)`), quindi potrebbe essere odioso pubblicare anche quel codice. L'utilizzo è piuttosto semplice, basta chiamare il metodo appropriato su qualche `url` con qualche payload `json` e il metodo restituirà una stringa come risultato che è possibile utilizzare e analizzare in un secondo momento. Supponiamo che la risposta sia un `json` , possiamo creare facilmente un `JSONObject` da esso:

```
String response = httpClient.post(MY_URL, JSON_PAYLOAD);
JSONObject json = new JSONObject(response);
// continue to parse the response according to it's structure
```

Ricevi Chiama

```
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());
  }
```

Ottieni chiamata asincrona

```
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());
      }
    });
  }
```

Registrazione dei parametri del modulo

```
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());
  }
```

Pubblicazione di una richiesta multipart

```
 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());
  }
```

Configurazione di OkHttp

Afferra via 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'
```

Leggi OkHttp online: https://riptutorial.com/it/android/topic/3625/okhttp

Capitolo 170: Okio

Examples

Scarica / Implementa

Scarica l'ultimo JAR o acquisisci tramite 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'
```

Decodificatore PNG

Decodificare i blocchi di un file PNG dimostra in pratica Okio.

```
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 e Buffers

ByteStrings e Buffers

Okio è costruito attorno a due tipi che racchiudono molte funzionalità in un'API semplice:

ByteString è una sequenza immutabile di byte. Per i dati dei personaggi, String è fondamentale. ByteString è il fratello perduto di String, rendendo facile trattare i dati binari come un valore. Questa classe è ergonomica: sa come codificare e decodificare se stessa come hex, base64 e UTF-8.

Il buffer è una sequenza mutabile di byte. Come ArrayList, non è necessario dimensionare il buffer in anticipo. Si leggono e scrivono i buffer come coda: scrivere i dati fino alla fine e leggerli dal lato anteriore. Non vi è alcun obbligo di gestire posizioni, limiti o capacità.

Internamente, `ByteString` e `Buffer` fanno cose intelligenti per risparmiare CPU e memoria. Se codifichi una stringa UTF-8 come ByteString, memorizza nella cache un riferimento a quella stringa in modo che se decodifichi in seguito, non c'è lavoro da fare.

`Buffer` è implementato come un elenco di segmenti collegati. Quando si spostano i dati da un buffer a un altro, viene riassegnata la proprietà dei segmenti anziché la copia dei dati. Questo approccio è particolarmente utile per i programmi multithread: un thread che comunica con la rete può scambiare dati con un thread di lavoro senza alcuna copia o cerimonia.

Leggi Okio online: https://riptutorial.com/it/android/topic/9952/okio

Capitolo 171: ORMLite in Android

Examples

Esempio di Android OrmLite su SQLite

ORMLite è un pacchetto di mappatura relazionale degli oggetti che fornisce funzionalità semplici e leggere per gli oggetti Java persistenti ai database SQL, evitando la complessità e il sovraccarico di pacchetti ORM standard.

Parlando per Android, OrmLite è implementato sul database supportato pronto all'uso, SQLite. Effettua chiamate dirette all'API per accedere a SQLite.

Configurazione gradle

Per iniziare dovresti includere il pacchetto nel gradle di build.

```
 // https://mvnrepository.com/artifact/com.j256.ormlite/ormlite-android
compile group: 'com.j256.ormlite', name: 'ormlite-android', version: '5.0'
POJO configuration
```

Quindi è necessario configurare un POJO da mantenere nel database. Qui bisogna fare attenzione alle annotazioni:

- Aggiungi l'annotazione @DatabaseTable all'inizio di ogni classe. Puoi anche usare @Entity.
- Aggiungi l'annotazione @DatabaseField prima di ogni campo da mantenere. Puoi anche usare @Column e altri.
- Aggiungi un costruttore senza argomento a ogni classe con almeno visibilità del pacchetto.

```
@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;
    }
}
```

Nell'esempio sopra c'è una tabella (form_model) con 4 campi.

il campo ID è l'indice generato automaticamente.

username è un indice del database.

Ulteriori informazioni sull'annotazione sono disponibili nella documentazione ufficiale .

Database Helper

Per continuare, è necessario creare una classe helper del database che dovrebbe estendere la classe OrmLiteSqliteOpenHelper.

Questa classe crea e aggiorna il database quando viene installata l'applicazione e può anche fornire le classi DAO utilizzate dalle altre classi.

DAO è l'acronimo di Data Access Object e fornisce tutte le funzionalità di Scrum e si specializza nella gestione di una singola classe persistente.

La classe helper deve implementare i seguenti due metodi:

- onCreate (SQLiteDatabase sqliteDatabase, ConnectionSource connectionSource);

  onCreate crea il database quando la tua app viene installata per la prima volta

- onUpgrade (database SQLiteDatabase, ConnectionSource connectionSource, int oldVersion, int newVersion);

  onUpgrade gestisce l'aggiornamento delle tabelle del database quando si aggiorna l'app a una nuova versione

Esempio di classe Database Helper:

```
  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;
        }
    }
```

Oggetto persistente in SQLite

Infine, la classe che persiste l'oggetto nel database.

```
     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;
    }
}
```

L'accessor del DOA presso il costruttore della suddetta classe è definito come:

```
 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;
}
```

Leggi ORMLite in Android online: https://riptutorial.com/it/android/topic/7571/ormlite-in-android

Capitolo 172: Ottenere calcolato Visualizzare le dimensioni

Osservazioni

Si noti che un'istanza `ViewTreeObserver` associata a un'istanza `View` può diventare non valida mentre quella `View` è ancora attiva. Dal `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()}.
```

Pertanto, se in precedenza è stato aggiunto un listener a un'istanza `ViewTreeObserver` e ora si desidera rimuoverlo, è più semplice chiamare `getViewTreeObserver` nell'istanza `View` corrispondente nuovamente per ricevere una nuova istanza `ViewTreeObserver` . (Il controllo di `isAlive` su un'istanza esistente è più utile per un piccolo vantaggio: se `ViewTreeObserver` non è più `ViewTreeObserver` , comunque stai recuperando quella nuova referenza!)

Examples

Calcolo delle dimensioni della vista iniziale in un'attività

```
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);
    }

}
```

Leggi Ottenere calcolato Visualizzare le dimensioni online:
https://riptutorial.com/it/android/topic/115/ottenere-calcolato-visualizzare-le-dimensioni

Capitolo 173: Ottenere i nomi dei font di sistema e usare i font

introduzione

I seguenti esempi mostrano come recuperare i nomi predefiniti dei font di sistema che sono memorizzati nella directory */ system / fonts /* e come usare un font di sistema per impostare il carattere di un elemento `TextView` .

Examples

Ottenere i nomi dei font di 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());
    }
}
```

Applicazione di un carattere di sistema a TextView

Nel seguente codice è necessario sostituire `fontsname` con il nome del carattere che si desidera utilizzare:

```
TextView lblexample = (TextView) findViewById(R.id.lblexample);
lblexample.setTypeface(Typeface.createFromFile("/system/fonts/" + "fontsname" + ".ttf"));
```

Leggi Ottenere i nomi dei font di sistema e usare i font online:
https://riptutorial.com/it/android/topic/10930/ottenere-i-nomi-dei-font-di-sistema-e-usare-i-font

Capitolo 174: Ottimizzazione del kernel Android

Examples

Configurazione RAM insufficiente

Android ora supporta dispositivi con 512 MB di RAM. Questa documentazione ha lo scopo di aiutare gli OEM a ottimizzare e configurare Android 4.4 per dispositivi con poca memoria. Molte di queste ottimizzazioni sono abbastanza generiche da poter essere applicate anche alle versioni precedenti.

Abilita il flag Dispositivo basso Ram

Stiamo introducendo una nuova API chiamata ActivityManager.isLowRamDevice () per le applicazioni per determinare se devono disattivare funzionalità specifiche che richiedono molta memoria e che funzionano male su dispositivi a bassa memoria.

Per i dispositivi da 512 MB, questa API dovrebbe restituire: "true" Può essere abilitata dalla seguente proprietà di sistema nel makefile del dispositivo.

```
PRODUCT_PROPERTY_OVERRIDES += ro.config.low_ram=true
```

Disabilita JIT

L'utilizzo della memoria JIT a livello di sistema dipende dal numero di applicazioni in esecuzione e dall'impronta del codice di tali applicazioni. Il JIT stabilisce una dimensione massima della cache del codice tradotto e tocca le pagine al suo interno secondo necessità. JIT costa tra 3M e 6M su un tipico sistema in esecuzione.

Le app di grandi dimensioni tendono a massimizzare la cache del codice abbastanza rapidamente (che per impostazione predefinita è stata 1M). In media, l'utilizzo della cache JIT funziona da qualche parte tra 100K e 200K byte per app. Ridurre la dimensione massima della cache può aiutare un po 'con l'utilizzo della memoria, ma se impostato troppo basso invierà il JIT in modalità thrashing. Per i dispositivi veramente a bassa memoria, si consiglia di disabilitare completamente JIT.

Questo può essere ottenuto aggiungendo la seguente riga al makefile del prodotto:

```
PRODUCT_PROPERTY_OVERRIDES += dalvik.vm.jit.codecachesize=0
```

Come aggiungere un CPU Governor

Il governor CPU stesso è solo un file C, che si trova in kernel_source / drivers / cpufreq /, ad esempio: cpufreq_smartass2.c. Sei responsabile di te stesso per trovare il governatore (guarda in

un repository del kernel esistente per il tuo dispositivo) Ma per chiamare e compilare con successo questo file nel tuo kernel dovrai apportare le seguenti modifiche:

1. Copia il tuo file governor (cpufreq_govname.c) e vai a kernel_source / drivers / cpufreq, ora incollalo.
2. e apri Kconfig (questa è l'interfaccia del layout del menu di configurazione) quando aggiungi un kernel, vuoi che venga visualizzato nella tua configurazione. Puoi farlo aggiungendo la scelta del governatore.

```
config CPU_FREQ_GOV_GOVNAMEHERE
tristate "'gov_name_lowercase' cpufreq governor"
depends on CPU_FREQ
help
governor' - a custom governor!
```

ad esempio, per smartassV2.

```
config CPU_FREQ_GOV_SMARTASS2
 tristate "'smartassV2' cpufreq governor"
 depends on CPU_FREQ
 help
 'smartassV2' - a "smart" optimized governor!
```

accanto all'aggiunta della scelta, devi anche dichiarare la possibilità che il governatore venga scelto come governatore predefinito.

```
 config CPU_FREQ_DEFAULT_GOV_GOVNAMEHERE
 bool "gov_name_lowercase"
 select CPU_FREQ_GOV_GOVNAMEHERE
 help
 Use the CPUFreq governor 'govname' as default.
```

ad esempio, per smartassV2.

```
config CPU_FREQ_DEFAULT_GOV_SMARTASS2
 bool "smartass2"
 select CPU_FREQ_GOV_SMARTASS2
 help
 Use the CPUFreq governor 'smartassV2' as default.
```

- Non riesci a trovare il posto giusto per metterlo? Cerca `"CPU_FREQ_GOV_CONSERVATIVE"` e inserisci il codice sotto, la stessa cosa conta per `"CPU_FREQ_DEFAULT_GOV_CONSERVATIVE"`

Ora che Kconfig è finito puoi salvare e chiudere il file.

3. Mentre `/drivers/cpufreq` ancora nella cartella `/drivers/cpufreq` , apri Makefile. In Makefile, aggiungi la riga corrispondente al tuo CPU Governor. per esempio:

```
obj-$(CONFIG_CPU_FREQ_GOV_SMARTASS2)    += cpufreq_smartass2.o
```

Be ware che non chiami il file C nativo, ma il file O! che è il file C compilato. Salva il file.

4. Sposta in: `kernel_source/includes/linux` . Ora apri `cpufreq.h` Scorri verso il basso finché non vedi qualcosa del tipo:

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_ONDEMAND)
 extern struct cpufreq_governor cpufreq_gov_ondemand;
 #define CPUFREQ_DEFAULT_GOVERNOR    (&amp;cpufreq_gov_ondemand)
```

(sono elencati anche altri governatori della CPU)

Ora aggiungi la tua voce con la CPU Governor selezionata, ad esempio:

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_SMARTASS2)
 extern struct cpufreq_governor cpufreq_gov_smartass2;
 #define CPUFREQ_DEFAULT_GOVERNOR (&amp;cpufreq_gov_smartass2)
```

Salva il file e chiudilo.

La configurazione iniziale di CPU Governor è ora completa. quando hai eseguito tutti i passaggi correttamente, dovresti essere in grado di scegliere il tuo governatore dal menu (`menuconfig` , `xconfig` , `gconfig` , `nconfig`). Una volta selezionato nel menu, verrà incluso nel kernel.

Commit che è quasi uguale alle istruzioni di cui sopra: Aggiungi smartassV2 e commit del governatore lulzactive

Schedulatori I / O

È possibile migliorare il kernel aggiungendo nuovi scheduler I / O, se necessario. Globalmente, governatori e scheduler sono uguali; entrambi forniscono un modo in cui il sistema dovrebbe funzionare. Tuttavia, per gli scheduler si tratta esclusivamente del flusso di dati di input / output, ad eccezione delle impostazioni della CPU. Gli scheduler I / O decidono come verrà programmata un'attività imminente di I / O. Gli scheduler standard come *noop* o *cfq* si *comportano* in modo molto ragionevole.

Gli schedulatori I / O possono essere trovati in *kernel_source / block* .

1. Copia il tuo schedulatore di I / O (per esempio, *sio-iosched.c*) e *vai* a *kernel_source / block* . Incolla il file dello schedulatore qui.

2. Ora apri *Kconfig.iosched* e aggiungi la tua scelta a *Kconfig* , ad esempio per *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. Quindi imposta l'opzione di scelta predefinita come segue:

```
default "sio" if DEFAULT_SIO
```

Salva il file.

4. Apri il *Makefile* in *kernel_source* / *block* / e aggiungi semplicemente la seguente riga per *SIO* :

```
obj-$(CONFIG_IOSCHED_SIO)    += sio-iosched.o
```

Salva il file e il gioco è fatto! Gli scheduler I / O dovrebbero ora apparire nella finestra di configurazione del menu.

Commit simile su GitHub: aggiunto lo scheduler I / O semplice .

Leggi Ottimizzazione del kernel Android online:
https://riptutorial.com/it/android/topic/9106/ottimizzazione-del-kernel-android

Capitolo 175: Ottimizzazione delle prestazioni

introduzione

Le prestazioni delle tue app sono un elemento cruciale dell'esperienza utente. Cerca di evitare schemi di cattiva esecuzione come lavorare sul thread dell'interfaccia utente e imparare come scrivere applicazioni veloci e reattive.

Examples

Salvare le ricerche View con il pattern ViewHolder

Soprattutto in un `ListView` , è possibile incorrere in problemi di prestazioni facendo troppe chiamate `findViewById()` durante lo scorrimento. Utilizzando il pattern `ViewHolder` , è possibile salvare queste ricerche e migliorare le prestazioni di `ListView` .

Se la tua voce di elenco contiene un singolo `TextView` , crea una classe `ViewHolder` per memorizzare l'istanza:

```
static class ViewHolder {
    TextView myTextView;
}
```

Durante la creazione dell'elemento della lista, allegare un oggetto `ViewHolder` alla voce dell'elenco:

```
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;
}
```

Utilizzando questo modello, `findViewById()` verrà chiamato solo quando viene creata una nuova `View` e `ListView` può riciclare le visualizzazioni in modo molto più efficiente.

Leggi Ottimizzazione delle prestazioni online:
https://riptutorial.com/it/android/topic/8711/ottimizzazione-delle-prestazioni

Capitolo 176: Otto Event Bus

Osservazioni

Otto è deprecato a favore di `RxJava` e `RxAndroid` . Questi progetti consentono lo stesso modello di programmazione event-driven di Otto, ma sono più capaci e offrono un migliore controllo del threading.

Examples

Passare un evento

Questo esempio descrive il passaggio di un evento utilizzando il bus eventi Otto .

Per utilizzare il bus eventi Otto in **Android Studio** devi inserire la seguente dichiarazione nel file gradle dei tuoi moduli:

```
dependencies {
    compile 'com.squareup:otto:1.3.8'
}
```

L'evento che vorremmo passare è un semplice oggetto Java:

```
public class DatabaseContentChangedEvent {
    public String message;

    public DatabaseContentChangedEvent(String message) {
        this.message = message;
    }
}
```

Abbiamo bisogno di un autobus per inviare eventi. Questo è in genere 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() {
    }
}
```

Per inviare un evento abbiamo solo bisogno del nostro BusProvider e del suo metodo `post` . Qui inviamo un evento se l'azione di un AsyncTask è completata:

```
public abstract class ContentChangingTask extends AsyncTask<Object, Void, Void> {

    ...

    @Override
    protected void onPostExecute(Void param) {
        BusProvider.getInstance().post(
            new DatabaseContentChangedEvent("Content changed")
        );
    }
}
```

Ricevere un evento

Per ricevere un evento è necessario implementare un metodo con il tipo di evento come parametro e annotarlo usando `@Subscribe` . Inoltre devi registrare / annullare la registrazione dell'istanza del tuo oggetto su `BusProvider` (vedi esempio *Invio di 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: per ricevere quell'evento deve esistere un'istanza della classe. Questo di solito non è il caso quando si desidera inviare un risultato da un'attività a un'altra attività. Quindi controlla il tuo caso d'uso per il bus eventi.

Leggi Otto Event Bus online: https://riptutorial.com/it/android/topic/6068/otto-event-bus

Capitolo 177: PackageManager

Examples

Recupera la versione dell'applicazione

```
public String getAppVersion() throws PackageManager.NameNotFoundException {
    PackageManager manager = getApplicationContext().getPackageManager();
    PackageInfo info = manager.getPackageInfo(
                            getApplicationContext().getPackageName(),
                            0);

    return info.versionName;
}
```

Nome versione e codice versione

Per ottenere `versionName` e `versionCode` della build corrente della tua applicazione devi `versionName` il gestore di pacchetti di 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
}
```

Tempo di installazione e tempo di aggiornamento

Per ottenere il momento in cui è stata installata o aggiornata la tua app, dovresti consultare il gestore di pacchetti di 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
}
```

Metodo di utilità che utilizza PackageManager

Qui possiamo trovare qualche metodo utile usando PackageManager,

Di seguito il metodo aiuterà a ottenere il nome dell'app usando il nome del pacchetto

```
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;
}
```

Di seguito il metodo aiuterà a ottenere l'icona dell'app usando il nome del pacchetto,

```
private Drawable getAppIcon(String packageName, Context context) {
    Drawable appIcon = null;
    try {
        appIcon = context.getPackageManager().getApplicationIcon(packageName);
    } catch (PackageManager.NameNotFoundException e) {
    }

    return appIcon;
}
```

Sotto il metodo aiuterà a ottenere l'elenco delle applicazioni installate.

```
public static List<ApplicationInfo> getLaunchIntent(PackageManager packageManager) {

    List<ApplicationInfo> list =
packageManager.getInstalledApplications(PackageManager.GET_META_DATA);

    return list;
}
```

Nota: il metodo sopra fornirà anche l'applicazione di avvio.

Sotto il metodo aiuterà a nascondere l'icona dell'app dal programma di avvio.

```
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: dopo aver spento il dispositivo e acceso questa icona tornerà nel programma di avvio.

Leggi PackageManager online: https://riptutorial.com/it/android/topic/4670/packagemanager

Capitolo 178: Parcelable

introduzione

Parcelable è un'interfaccia specifica per Android in cui implementare personalmente la serializzazione. È stato creato per essere molto più efficiente di Serializable e per aggirare alcuni problemi con lo schema di serializzazione Java predefinito.

Osservazioni

È importante ricordare che l'ordine in cui si scrivono i campi in un pacco DEVE ESSERE LO STESSO ORDINE che li si legge fuori dal pacco quando si costruisce il proprio oggetto personalizzato.

L'interfaccia parcelable ha un limite di dimensioni di 1 MB. Ciò significa che qualsiasi oggetto o combinazione di oggetti, inseriti in un pacco che occupano più di 1 MB di spazio, sarà danneggiato dall'altra parte. Questo può essere difficile da scoprire, quindi tieni a mente che tipo di oggetti hai intenzione di rendere parcelable. Se hanno grandi alberi di dipendenza, considera un altro modo per trasmettere i dati.

Examples

Rendere un oggetto personalizzato 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];
        }
    };
}
```

Oggetto parcelable contenente un altro oggetto Parcelable

Un esempio di una classe che contiene una classe parcelable all'interno:

```
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;
    }
}
```

Il proprietario è solo una normale classe parcelable.

Usare 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
}
```

Questo è preferibile (per esempio) usando un ordinale, perché l'inserimento di nuovi valori nell'enumerazione non influisce sui valori memorizzati in precedenza

Leggi Parcelable online: https://riptutorial.com/it/android/topic/1849/parcelable

Capitolo 179: Perdite di memoria

Examples

Perdite di memoria comuni e come risolverli

1. Risolvi i tuoi contesti:

Prova ad usare il contesto appropriato: ad esempio poiché un Toast può essere visto in molte attività invece che in una sola, usa `getApplicationContext()` per toast e dato che i servizi possono continuare a funzionare anche se un'attività è terminata, avvia un servizio con:

```
Intent myService = new Intent(getApplicationContext(), MyService.class);
```

Utilizza questa tabella come guida rapida per quale contesto è appropriato:

	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	

Articolo originale sul contesto qui .

2. Riferimento statico al contesto

Un errore di perdita di memoria serio sta mantenendo un riferimento statico a `View` . Ogni `View` ha un riferimento interno al `Context` . Il che significa che una vecchia attività con la sua intera gerarchia di viste non sarà raccolta di rifiuti fino a quando l'app non viene terminata. Avrai la tua app due volte in memoria ruotando lo schermo.

Assicurati che non ci sia assolutamente alcun riferimento statico a View, Context o ai loro discendenti.

3. Controlla che stai effettivamente finendo i tuoi servizi.

Ad esempio, ho un intentService che utilizza l'API del servizio di localizzazione di Google. E ho dimenticato di chiamare `googleApiClient.disconnect();` :

```
//Disconnect from API onDestroy()
if (googleApiClient.isConnected()) {
    LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient,
GoogleLocationService.this);
    googleApiClient.disconnect();
}
```

4. Verifica l'utilizzo di immagini e bitmap:

Se stai usando la **libreria di Picasso di** Square, ho scoperto che stavo perdendo memoria non usando il `.fit()` , che ha ridotto drasticamente il mio footprint di memoria da 50 MB in media a meno di 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. Se si utilizzano ricevitori di trasmissione, annullarne la registrazione.

6. Se si utilizza `java.util.Observer` (pattern Observer):

Assicurati di usare `deleteObserver(observer);`

Evitare le perdite di attività con AsyncTask

> **Una parola di cautela** : AsyncTask ha molti segreti a parte la perdita di memoria descritta qui. Quindi, fai attenzione con questa API o evita del tutto se non comprendi pienamente le implicazioni. Ci sono molte alternative (Thread, EventBus, RxAndroid, ecc.).

Un errore comune con `AsyncTask` è quello di acquisire un riferimento forte `Activity` (o `Fragment`) dell'host:

```
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!
  }
}
```

Questo è un problema perché `AsyncTask` può facilmente sopravvivere `Activity` padre, ad esempio se una modifica alla configurazione avviene mentre l'attività è in esecuzione.

Il modo giusto per farlo è quello di rendere il vostro compito di una `static` di classe, che non cattura il genitore, e in possesso di un debole, di riferimento per l'host `Activity` :

```
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
    }
  }
}
```

Richiamata anonima in attività

Ogni volta che crei una classe anonima, mantiene un riferimento implicito alla sua classe genitore. Quindi quando scrivi:

```
public class LeakyActivity extends Activity
{

...

    foo.registerCallback(new BarCallback()
    {
        @Override
        public void onBar()
        {
            // do something
        }
    });
}
```

In effetti, stai inviando un riferimento alla tua istanza LeakyActivity per cedere. Quando l'utente si allontana da LeakyActivity, questo riferimento può impedire che l'istanza LeakyActivity venga trucchiata. Si tratta di una grave perdita poiché le attività mantengono un riferimento all'intera

gerarchia della vista e sono quindi oggetti di grandi dimensioni nella memoria.

Come evitare questa perdita:

Ovviamente puoi evitare di utilizzare interamente le callback anonime nelle attività. Puoi anche annullare la registrazione di tutte le tue richiamate rispetto al ciclo di vita delle attività. così:

```
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);
    }
}
```

Contesto di attività in classi statiche

Spesso vorrai avvolgere alcune delle classi di Android in classi di utilità più facili da usare. Queste classi di utilità spesso richiedono un contesto per accedere al sistema operativo Android o alle risorse delle tue app. Un esempio comune di questo è un wrapper per la classe SharedPreferences. Per accedere alle preferenze condivise di Androids è necessario scrivere:

```
context.getSharedPreferences(prefsName, mode);
```

E quindi si può essere tentati di creare la seguente classe:

```
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);
    }
}
```

ora, se chiami `init()` con il tuo contesto di attività, LeakySharedPrefsWrapper manterrà un riferimento alla tua attività, impedendole di essere triturata.

Come evitare:

Quando si chiamano le funzioni di supporto statico, è possibile inviare il contesto dell'applicazione utilizzando `context.getApplicationContext();`

Quando si creano funzioni di supporto statiche, è possibile estrarre il contesto dell'applicazione dal contesto in cui viene fornito (Chiamare getApplicationContext () sul contesto dell'applicazione restituisce il contesto dell'applicazione). Quindi la correzione del nostro wrapper è semplice:

```
public static void init(Context context)
{
    sContext = context.getApplicationContext();
}
```

Se il contesto dell'applicazione non è appropriato per il tuo caso d'uso, puoi includere un parametro di contesto in ciascuna funzione di utilità, dovresti evitare di mantenere i riferimenti a questi parametri di contesto. In questo caso la soluzione sarebbe così:

```
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);
}
```

Rileva le perdite di memoria con la libreria LeakCanary

LeakCanary è una libreria Java Open Source per rilevare perdite di memoria nelle build di debug.

Basta aggiungere le dipendenze in `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'
 }
```

Quindi nella classe `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);
  }
}
```

Ora LeakCanary mostrerà automaticamente una notifica quando viene rilevata una perdita di memoria dell'attività nella build di **debug** .

NOTA: il codice di rilascio non conterrà alcun riferimento a LeakCanary oltre alle due classi vuote esistenti nella `leakcanary-android-no-op` .

Evitare le perdite di attività con gli ascoltatori

Se si implementa o si crea un listener in un'attività, prestare sempre attenzione al ciclo di vita dell'oggetto che ha registrato il listener.

Considera un'applicazione in cui abbiamo diverse attività / frammenti interessati quando un utente ha effettuato il login o l'uscita. Un modo per farlo sarebbe quello di avere un'istanza singleton di un `UserController` che può essere sottoscritto per ricevere una notifica quando lo stato dell'utente cambia:

```
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();
    }
}
```

Quindi ci sono due attività, `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();
    }
}
```

E `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();
    }
}
```

Quello che succede con questo esempio è che ogni volta che l'utente esegue il login e poi si disconnette di nuovo, viene fuoriuscita un'istanza `MainActivity` . La perdita si verifica perché vi è un riferimento all'attività in `UserController#listeners` .

Nota: anche se usassimo una classe interna anonima come ascoltatrice, l'attività verrebbe comunque trapelata:

```
...
this.userController.registerUserStateChangeListener(new UserController.StateListener() {
    @Override
    public void userLoggedIn() {
        showUserAccount();
    }

    @Override
    public void userLoggedOut() {
        finish();
    }
});
...
```

L'attività sarebbe ancora in perdita, perché la classe interna anonima ha un riferimento implicito alla classe esterna (in questo caso l'attività). Questo è il motivo per cui è possibile chiamare i metodi di istanza nella classe esterna dalla classe interna. In effetti, l'unico tipo di classi interne che non hanno un riferimento alla classe esterna sono classi interne **statiche** .

In breve, tutte le istanze di classi interne non statiche contengono un riferimento implicito all'istanza della classe esterna che le ha create.

Esistono due approcci principali per risolvere questo problema, aggiungendo un metodo per rimuovere un listener dagli ascoltatori di `UserController#listeners` o utilizzando un `WeakReference` per mantenere il riferimento degli ascoltatori.

Alternativa 1: rimozione degli ascoltatori

Iniziamo con la creazione di un nuovo metodo `removeUserStateChangeListener(StateListener listener)` :

```
public class UserController {

    ...

    public void registerUserStateChangeListener(StateListener listener) {
        listeners.add(listener);
    }
```

```
    public void removeUserStateChangeListener(StateListener listener) {
        listeners.remove(listener);
    }

    ...
}
```

Chiamiamo quindi questo metodo nel metodo `onDestroy` dell'attività:

```
public class MainActivity extends Activity implements UserController.StateListener{
    ...

    @Override
    protected void onDestroy() {
        super.onDestroy();
        userController.removeUserStateChangeListener(this);
    }
}
```

Con questa modifica le istanze di `MainActivity` non vengono più trapelate quando l'utente esegue il login e l'uscita. Tuttavia, se la documentazione non è chiara, è probabile che il prossimo sviluppatore che inizia a utilizzare `UserController` possa perdere la necessità di annullare la registrazione del listener quando l'attività viene distrutta, il che ci porta al secondo metodo per evitare questi tipi di perdite.

Alternativa 2: utilizzo di riferimenti deboli

Innanzitutto, iniziamo spiegando che cos'è un riferimento debole. Un riferimento debole, come suggerisce il nome, contiene un debole riferimento a un oggetto. Rispetto a un campo di istanza normale, che è un riferimento forte, un riferimento debole non impedisce al GC, il programma di raccolta dati obsoleti, di rimuovere gli oggetti. Nell'esempio sopra questo permetterebbe a `MainActivity` di essere garbage-collected dopo che è stato distrutto se `UserController` usato `WeakReference` al riferimento degli ascoltatori.

In breve, un riferimento debole indica al GC che se nessun altro ha un forte riferimento a questo oggetto, andare avanti e rimuoverlo.

Cerchiamo di modificare `UserController` per utilizzare un elenco di `WeakReference` per tenere traccia dei suoi ascoltatori:

```
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 questa modifica non importa se gli ascoltatori vengono rimossi o meno, dal momento che `UserController` non ha riferimenti forti a nessuno degli ascoltatori. Tuttavia, scrivere questo codice boilerplate ogni volta è ingombrante. Pertanto, creiamo una classe generica chiamata `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);
        }
    }
}
```

Ora riscriviamo `UserController` per utilizzare `WeakCollection<T>` :

```
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();
        }
    }

    ...
}
```

Come mostrato nell'esempio di codice sopra, la `WeakCollection<T>` rimuove tutto il codice boilerplate necessario per usare `WeakReference` invece di una normale lista. Per `UserController#removeUserStateChangeListener(StateListener)` tutto: se `UserController#removeUserStateChangeListener(StateListener)` una chiamata a `UserController#removeUserStateChangeListener(StateListener)` , il listener e tutti gli oggetti a cui fa riferimento non si diffonderanno.

Evita perdite di memoria con la classe anonima, il gestore, il compito del timer, il thread

In Android, ogni sviluppatore utilizza la `Anonymous Class` (eseguibile) almeno una volta in un progetto. Qualsiasi `Anonymous Class` ha un riferimento al suo genitore (attività). Se eseguiamo un'attività di lunga durata, l'attività principale non verrà distrutta fino al termine dell'attività. Esempio utilizza il gestore e la classe `Runnable` anonima. La memoria si perde quando abbandoniamo l'attività prima che il `Runnable` sia finito.

```
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...
```

Come lo risolviamo?

1. Non eseguire operazioni a lungo con la `Anonymous Class` o abbiamo bisogno di una `Static class` per esso e passiamo a `WeakReference` (come attività, vista ...). `Thread` è lo stesso con la `Anonymous Class` .
2. Annulla il `Handler` , `Timer` quando l'attività viene distrutta.

Leggi Perdite di memoria online: https://riptutorial.com/it/android/topic/2687/perdite-di-memoria

Capitolo 180: Picasso

introduzione

Picasso è una libreria di immagini per Android. È stato creato e gestito da Square . Semplifica il processo di visualizzazione delle immagini da posizioni esterne. La libreria gestisce ogni fase del processo, dalla richiesta HTTP iniziale alla memorizzazione nella cache dell'immagine. In molti casi, sono necessarie solo poche righe di codice per implementare questa libreria ordinata.

Osservazioni

Picasso è una potente libreria di download e memorizzazione di immagini per Android.
Segui questo esempio per aggiungere la libreria al tuo progetto.

Siti internet:

- fonte
- Doc
- Registro delle modifiche

Examples

Aggiunta della Libreria Picasso al tuo progetto Android

Dalla documentazione ufficiale :

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>
```

Segnaposto e gestione degli errori

Picasso supporta sia segnaposti di download che di errore come funzionalità opzionali. Inoltre,

fornisce i callback per la gestione del risultato del download.

```
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() {}
   });
```

Una richiesta verrà ritentata tre volte prima che venga visualizzato il segnaposto dell'errore.

Ridimensionamento e rotazione

```
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);
```

Avatar circolari con Picasso

Ecco un esempio di classe di trasformazione Circle di Picasso basata sull'originale , con l'aggiunta di un bordo sottile e include anche la funzionalità per un separatore opzionale per l'impilamento:

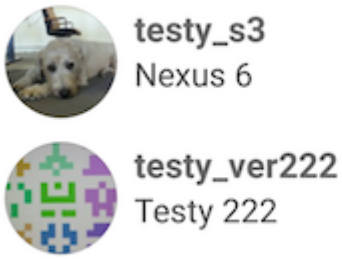
```
import android.graphics.Bitmap;
import android.graphics.BitmapShader;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Style;

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";
    }
}
```

Ecco come usarlo quando si carica un'immagine (supponendo che `this` tratti di un contesto di attività, e `url` è una stringa con l'url dell'immagine da caricare):

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform())
    .into(ivAvatar);
```

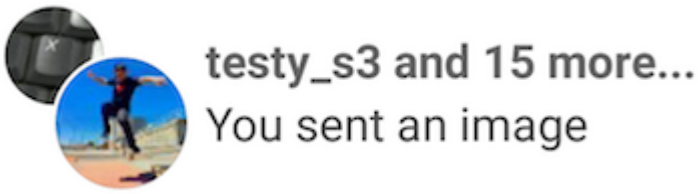
Risultato:



Per l'uso con il separatore, dare `true` al costruttore per l'immagine in alto:

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform(true))
    .into(ivAvatar);
```

Risultato (due ImageViews in un FrameLayout):



Disattiva la cache in Picasso

```
Picasso.with(context)
.load(uri)
.networkPolicy(NetworkPolicy.NO_CACHE)
.memoryPolicy(MemoryPolicy.NO_CACHE)
.placeholder(R.drawable.placeholder)
.into(imageView);
```

Caricamento immagine da spazio esterno

```
String filename = "image.png";
String imagePath = getExternalFilesDir() + "/" + filename;

Picasso.with(context)
    .load(new File(imagePath))
    .into(imageView);
```

Download dell'immagine come bitmap usando Picasso

Se vuoi scaricare l'immagine come `Bitmap` usando il codice seguente di `Picasso` ti aiuterà:

```
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) {
            }
        });
```

Annullamento di richieste di immagini tramite Picasso

In alcuni casi è necessario cancellare una richiesta di download di immagini in Picasso prima che il download sia completato.

Ciò potrebbe accadere per vari motivi, ad esempio se la visualizzazione genitore è passata ad un'altra vista prima che il download dell'immagine possa essere completato.

In questo caso, puoi annullare la richiesta di download dell'immagine utilizzando il metodo `cancelRequest()` :

```
ImageView imageView;

//......

Picasso.with(imageView.getContext()).cancelRequest(imageView);
```

Utilizzo di Picasso come ImageGetter per Html.fromHtml

Utilizzo di Picasso come ImageGetter per 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);
    }

    //------------------------------------------------------------------//

}
}
```

L'utilizzo è semplice:

```
Html.fromHtml(textToParse, new PicassoImageGetter(picasso, textViewTarget), null);
```

Prova prima la cache del disco offline, poi vai online e recupera l'immagine

per prima cosa aggiungi OkHttp al file gradle build del modulo dell'app

```
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'
```

Quindi crea una classe che estende l'applicazione

```
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);

    }
}
```

aggiungilo al file Manifest come segue:

```
<application
        android:name=".Global"
        .. >

</application>
```

Uso normale

```
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");
            }
        });
    }
});
```

Link alla risposta originale

Leggi Picasso online: https://riptutorial.com/it/android/topic/2172/picasso

Capitolo 181: Ping ICMP

introduzione

La richiesta Ping ICMP può essere eseguita in Android creando un nuovo processo per eseguire la richiesta ping. L'esito della richiesta può essere valutato al completamento della richiesta ping dal suo processo.

Examples

Esegue un singolo Ping

Questo esempio tenta una singola richiesta Ping. Il comando ping all'interno della chiamata al metodo `runtime.exec` può essere modificato su qualsiasi comando ping valido che potresti eseguire tu stesso nella riga di comando.

```
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();
}
```

Leggi Ping ICMP online: https://riptutorial.com/it/android/topic/9434/ping-icmp

Capitolo 182: planata

introduzione

**** AVVERTENZA Questa documentazione non è mantenuta e spesso non accurata ****

La documentazione ufficiale di Glide è una fonte molto migliore:

Per Glide v4, vedi http://bumptech.github.io/glide/ . Per Glide v3, vedere https://github.com/bumptech/glide/wiki .

Osservazioni

Glide è una veloce ed efficiente gestione dei media open source e framework di caricamento delle immagini per Android che avvolge la decodifica dei media, la memoria e la cache del disco e il pooling delle risorse in un'interfaccia semplice e facile da usare.

Glide supporta il recupero, la decodifica e la visualizzazione di fermi immagine, immagini e GIF animate. Glide include un'API flessibile che consente agli sviluppatori di collegarsi a quasi tutti gli stack di rete.
Per impostazione predefinita, Glide utilizza uno stack basato su HttpUrlConnection personalizzato, ma include anche librerie di utilità plug-in nel progetto Volley di Google o nella libreria OkHttp di Square .

L'obiettivo principale di Glide è quello di far scorrere qualsiasi tipo di elenco di immagini il più agevole e veloce possibile, ma Glide è efficace anche in quasi tutti i casi in cui è necessario recuperare, ridimensionare e visualizzare un'immagine remota.

Codice sorgente e ulteriore documentazione sono disponibili su GitHub: https://github.com/bumptech/glide

Examples

Aggiungi Glide al tuo progetto

Dalla documentazione ufficiale :

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>
```

A seconda della configurazione e dell'utilizzo di ProGuard (DexGuard), potrebbe essere necessario includere le seguenti righe nel proguard.cfg (per ulteriori informazioni, vedere la wiki di Glide):

```
-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
```

Caricamento di un'immagine

ImageView

Per caricare un'immagine da un URL, Uri, ID risorsa o qualsiasi altro modello in un `ImageView` :

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
String yourUrl = "http://www.yoururl.com/image.png";

Glide.with(context)
    .load(yourUrl)
    .into(imageView);
```

Per Uris, sostituire `yourUrl` con il tuo Uri (`content://media/external/images/1`). Per i Drawable, sostituire `yourUrl` con il proprio ID risorsa (`R.drawable.image`).

RecyclerView e ListView

In ListView o RecyclerView, puoi utilizzare esattamente le stesse linee:

```
@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);
}
```

Se non si desidera avviare un caricamento in `onBindViewHolder` , assicurarsi di `clear()` qualsiasi `ImageView` Glide possa essere gestito prima di modificare `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);
    }
}
```

Trasformazione del cerchio di slittamento (carica l'immagine in un ImageView circolare)

Crea un'immagine circolare con la planata.

```
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);
```

Trasformazioni predefinite

Glide include due trasformazioni predefinite, adatte al centro e al centro del ritaglio.

Fit center:

```
Glide.with(context)
    .load(yourUrl)
    .fitCenter()
    .into(yourView);
```

Il centro Fit esegue la stessa trasformazione di ScaleType.FIT_CENTER di Android.

Coltura centrale:

```
Glide.with(context)
    .load(yourUrl)
    .centerCrop()
    .into(yourView);
```

Il ritaglio centrale esegue la stessa trasformazione di ScaleType.CENTER_CROP di Android.

Per ulteriori informazioni, vedere la wiki di Glide .

Scorri le immagini degli angoli arrotondati con il bersaglio Glide personalizzato

Per prima cosa rendi una classe di utilità o usa questo metodo nella classe necessaria

```
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);
        }
    };
}
```

Caricamento immagine:

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Poiché si utilizza asBitmap (), tuttavia, le animazioni verranno rimosse. Puoi usare la tua animazione in questo posto usando il metodo animate ().

Esempio con dissolvenza simile all'animazione Glide predefinita.

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .animate(R.anim.abc_fade_in)
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Si prega di notare che questa animazione è una risorsa privata della libreria di supporto - non è raccomandabile da utilizzare in quanto può cambiare o addirittura essere rimossa.

Si noti inoltre che è necessario disporre della libreria di supporto per utilizzare RoundedBitmapDrawableFactory

Precaricamento delle immagini

Per precaricare le immagini remote e assicurarsi che l'immagine venga scaricata solo una volta:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE)
    .preload();
```

Poi:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE) // ALL works here too
    .into(imageView);
```

Per precaricare le immagini locali e accertarsi che una copia trasformata si trovi nella cache del disco (e magari nella cache della memoria):

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // Or whatever transformation you want
    .preload(200, 200); // Or whatever width and height you want
```

Poi:

```
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);
```

Segnaposto e gestione degli errori

Se vuoi aggiungere un Drawable da mostrare durante il caricamento, puoi aggiungere un segnaposto:

```
Glide.with(context)
    .load(yourUrl)
    .placeholder(R.drawable.placeholder)
    .into(imageView);
```

Se vuoi che venga mostrato un Drawable se il caricamento fallisce per qualsiasi motivo:

```
Glide.with(context)
    .load(yourUrl)
    .error(R.drawable.error)
    .into(imageView);
```

Se vuoi che venga mostrato un Drawable se fornisci un modello null (URL, Uri, percorso file, ecc.):

```
Glide.with(context)
    .load(maybeNullUrl)
    .fallback(R.drawable.fallback)
    .into(imageView);
```

Carica l'immagine in un ImageView circolare senza trasformazioni personalizzate

Crea un BitmapImageViewTarget personalizzato per caricare l'immagine in:

```
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));
```

Gestione del caricamento dell'immagine Glide non riuscita

```
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);
    }
});
```

Qui al `SET_YOUR_DEFAULT_IMAGE` posto puoi impostare qualsiasi `Drawable` predefinito. Questa immagine verrà mostrata se il caricamento dell'immagine non è riuscito.

Leggi planata online: https://riptutorial.com/it/android/topic/1091/planata

Capitolo 183: Port Mapping utilizzando la libreria Cling in Android

Examples

Aggiunta del supporto Cling al tuo progetto Android

build.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'

}
```

Mappatura di una porta 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;
}
```

Leggi Port Mapping utilizzando la libreria Cling in Android online:
https://riptutorial.com/it/android/topic/6208/port-mapping-utilizzando-la-libreria-cling-in-android

Capitolo 184: Posizione

introduzione

Le API di Android Location sono utilizzate in un'ampia varietà di app per scopi diversi, come trovare la posizione dell'utente, notificare quando un utente ha lasciato un'area generale (Geofencing) e aiutare a interpretare l'attività dell'utente (camminare, correre, guidare, ecc.).

Tuttavia, le API di posizione Android non sono l'unico mezzo per acquisire la posizione dell'utente. Quanto segue fornirà esempi su come utilizzare il `LocationManager` di Android e altre librerie di posizioni comuni.

Osservazioni

Per creare app che supportano la localizzazione in Android, esistono due percorsi:

- `LocationManager` open source nativo di Android
- `FusedLocationProviderApi` di Google, che fa parte di Google Play Services

LocationManager

Professionisti

- Controllo più granulare
- Disponibile in tutti i dispositivi
- Parte del framework Android

Contro

- Lo scarico della batteria è un problema, se non gestito correttamente
- Richiede la logica per cambiare provider di posizione, se il dispositivo non è in grado di trovare una posizione (ad esempio, un GPS scadente all'interno di un edificio)

Caratteristiche

- Listener NMEA
- Listener di stato GPS
- Ascolta le modifiche allo stato del provider (ad esempio, il GPS è disattivato dall'utente)
- Elenco dei fornitori per scegliere l'origine della posizione da

provider

GPS

- Autorizzazioni richieste:
    - `ACCESS_FINE_LOCATION`

- Precisione: 10m - 100m
- Requisiti di potenza: ALTA
- Disponibilità: in tutto il mondo (con una visione chiara del cielo)
- **NOTE** :
    - Gli aggiornamenti delle posizioni di solito arrivano una volta al secondo, ma in situazioni in cui il GPS non è stato utilizzato per un po 'di tempo e l' A-GPS non è disponibile, ci vuole qualche minuto per ricevere una posizione.
    - Nei casi in cui la visione chiara del cielo è ostruita, i punti GPS non si raggruppano molto bene (i punti posizione "salta") e la precisione può essere fuorviante in certe aree a causa dell'effetto " Urban Canyon ".

Rete

- Autorizzazioni richieste:
    - `ACCESS_COARSE_LOCATION` o `ACCESS_FINE_LOCATION`
- Precisione: 100m - 1000m +
- Requisiti di alimentazione: LOW - MEDIUM
- Disponibilità: entro il raggio della torre cellulare o del segnale wifi
- **GLI APPUNTI:**
    - Gli aggiornamenti delle posizioni si verificano meno frequentemente del GPS
    - Gli aggiornamenti di posizione in genere non si raggruppano bene (i punti di localizzazione "saltano") e la precisione può variare in base al numero di fattori diversi (numero di segnali wifi, intensità del segnale, tipo di torre cellulare, ecc.)

Passivo

- Autorizzazioni richieste:
    - `ACCESS_FINE_LOCATION`
- Precisione: 10m - 1000m +
- Requisiti di alimentazione: NESSUNO
- Disponibilità: solo quando un'altra app riceve una posizione dal GPS o dalla rete
- **GLI APPUNTI:**
    - Non fare affidamento su questo per darti aggiornamenti continui. Questo ascolta passivamente altre app che fanno richieste di posizione e le restituisce.
    - Non restituisce punti FusedLocationProviderApi, solo i punti di localizzazione sottostanti utilizzati per generarli.

FusedLocationProviderApi

Professionisti

- Offre meno batteria scarica "fuori dalla scatola"
- Gestisce male il GPS
- Ottiene aggiornamenti più regolarmente

Contro

- Controllo meno granulare sul GPS
- Potrebbe non essere disponibile su tutti i dispositivi o in alcuni Paesi
- Richiede dipendenza dalla libreria di terze parti

Caratteristiche

- Uso ben gestito dei fornitori di localizzazione per risparmi ottimali sulle batterie
- Generalmente genera punti più precisi di Network Location Provider
- Aggiornamenti più frequenti della libreria, consentendo ulteriori miglioramenti
- Non è necessario specificare il tipo di provider da utilizzare

PosizioneRichiedi livelli di priorità

PRIORITY_HIGH_ACCURACY

- Autorizzazioni richieste:
  - `ACCESS_FINE_LOCATION` per una posizione più precisa o `ACCESS_COARSE_LOCATION` per una posizione meno precisa
- Precisione: 10m - 100m
- Requisiti di potenza: ALTA
- Disponibilità: ovunque siano disponibili i servizi di Google Play.
- **GLI APPUNTI:**
  - Se `ACCESS_FINE_LOCATION` non viene utilizzato, questo non utilizzerà il GPS per generare aggiornamenti di posizione, ma troverà comunque un punto abbastanza accurato nelle giuste condizioni.
  - Se si utilizza `ACCESS_FINE_LOCATION` , può o non può utilizzare il GPS per generare punti di posizione, a seconda di quanto accurato possa attualmente tracciare il dispositivo in base alle condizioni ambientali.
  - Sebbene questo possa riportare aggiornamenti di posizione più accurati rispetto alle altre impostazioni, è ancora soggetto all'effetto " Urban Canyon ".

PRIORITY_BALANCED_POWER_ACCURACY

- Autorizzazioni richieste:
  - `ACCESS_FINE_LOCATION` per una posizione più precisa o `ACCESS_COARSE_LOCATION` per una posizione meno precisa
- Precisione: 100m - 1000m +
- Requisiti di alimentazione: MEDIO
- Disponibilità: ovunque siano disponibili i servizi di Google Play.
- **GLI APPUNTI:**
  - Stesse note di `PRIORITY_HIGH_ACCURACY`
  - Sebbene sia improbabile, questa impostazione potrebbe comunque utilizzare il GPS per generare una posizione.

PRIORITY_LOW_POWER

- Autorizzazioni richieste:
  - `ACCESS_FINE_LOCATION` o `ACCESS_COARSE_LOCATION`

- Precisione: 100m - 1000m +
- Requisiti di alimentazione: LOW
- Disponibilità: ovunque siano disponibili i servizi di Google Play.
- **GLI APPUNTI:**
  - Probabilmente non usa il GPS, ma finora non è stato testato.
  - Gli aggiornamenti in genere non sono molto accurati
  - Usato generalmente per rilevare cambiamenti significativi nella posizione

PRIORITY_NO_POWER

- Autorizzazioni richieste:
  - `ACCESS_FINE_LOCATION` o `ACCESS_COARSE_LOCATION`
- Precisione: 10m - 1000m +
- Requisiti di alimentazione: NESSUNO
- Disponibilità: ovunque siano disponibili i servizi di Google Play.
- **GLI APPUNTI:**
  - Funziona in modo quasi identico a `LocationManager PASSIVE_PROVIDER`
  - Riporta gli aggiornamenti dei servizi di Google Play quando ricevuti, dove `PASSIVE_PROVIDER` riporta gli aggiornamenti di posizione sottostanti utilizzati

Risoluzione dei problemi

OnLocationChanged () Mai chiamato

Poiché questo sembra essere un problema comune con il rilevamento di posizioni Android, inserirò una breve lista di controllo delle correzioni comuni:

---

1. Controlla il tuo manifest!

   Uno dei problemi più comuni è che le autorizzazioni corrette non sono mai state fornite. Se utilizzi il GPS (con o senza rete), usa `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>` , altrimenti usa `<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>` . FusedLocationApi di Google richiede `ACCESS_FINE_LOCATION` .

---

2. (Per Android 6+) Verifica le autorizzazioni di runtime !

   Verifica e richiedi le autorizzazioni! Se non ti vengono mai concesse autorizzazioni, finirai con arresti anomali, o peggio (se stai riscontrando tutte le eccezioni), finirai senza alcuna indicazione di nulla! Non importa se l'utente concede il permesso all'inizio dell'app, controlla sempre se hai le autorizzazioni per tutte le chiamate. L'utente può facilmente accedere alle proprie impostazioni e revocarle.

---

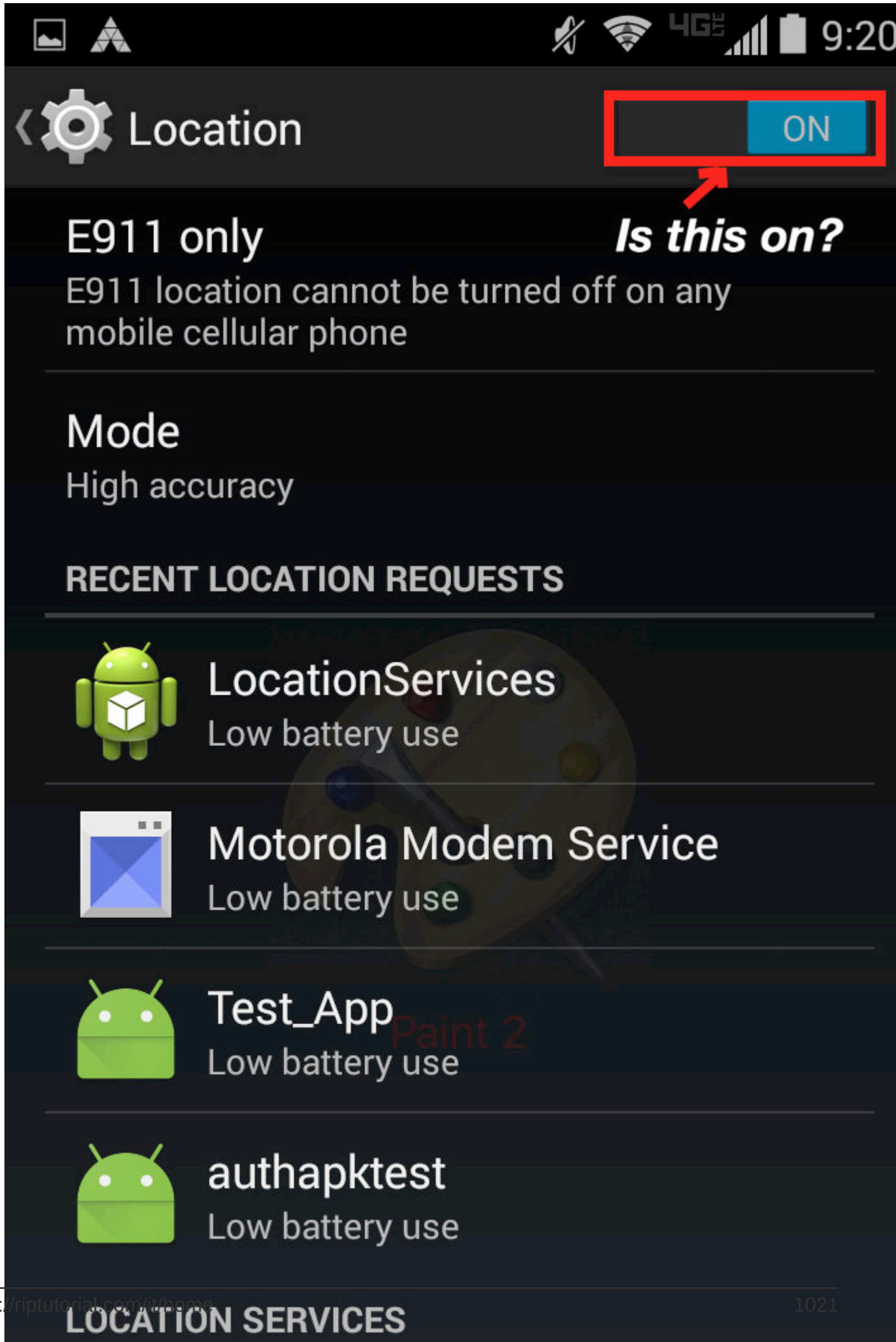
3. Controlla il tuo codice!

Sei sicuro di passare nell'ascoltatore giusto? Hai aggiunto `BroadcastReceiver` o `IntentService` al manifest? Stai utilizzando `PendingIntent.getService()` su una classe `BroadcastReceiver` o su `getBroadcast()` su una classe `IntentService` ? Sei sicuro di non annullare la registrazione del listener da qualche altra parte nel tuo codice subito dopo la richiesta?

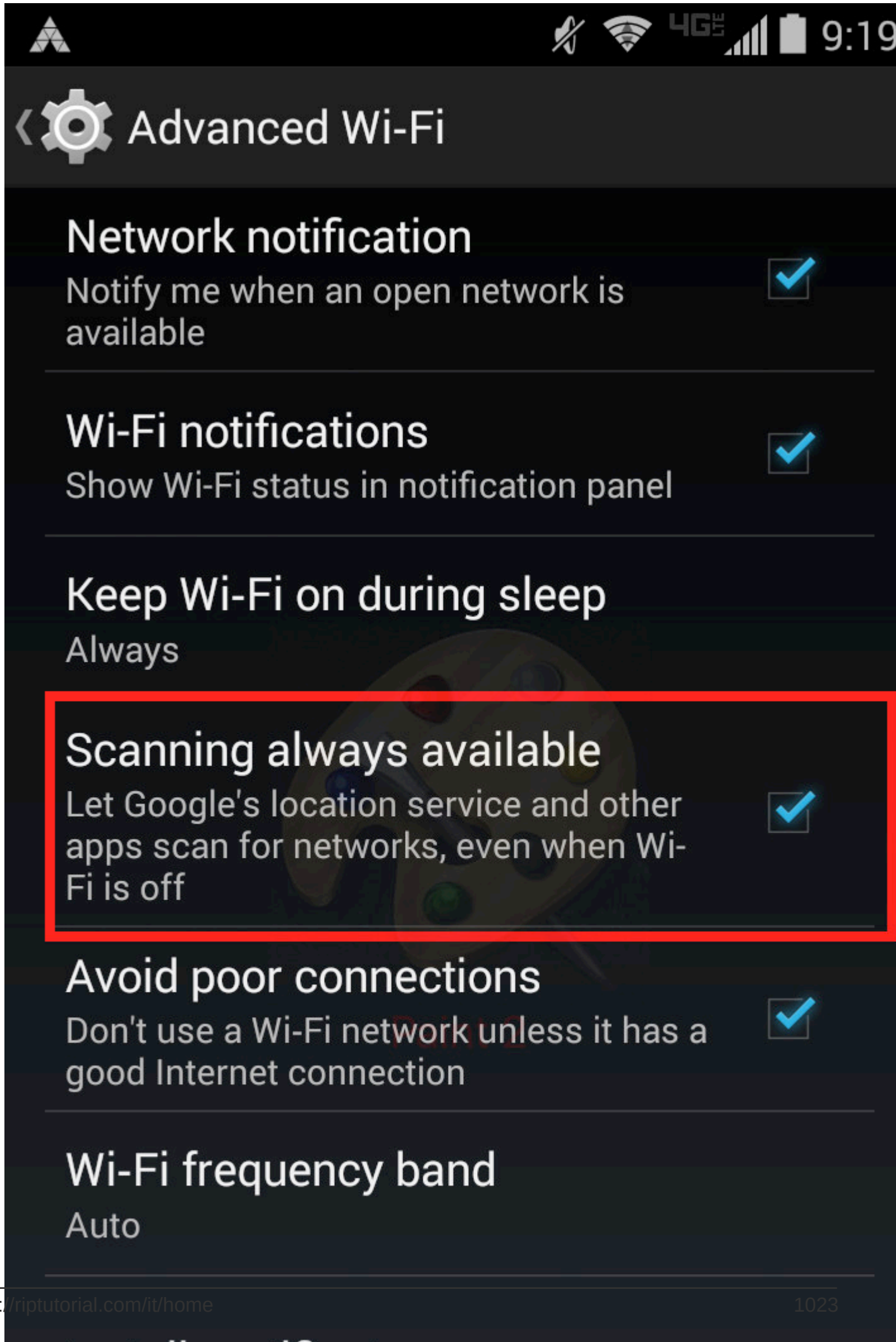
---

4. Controlla le impostazioni del dispositivo!

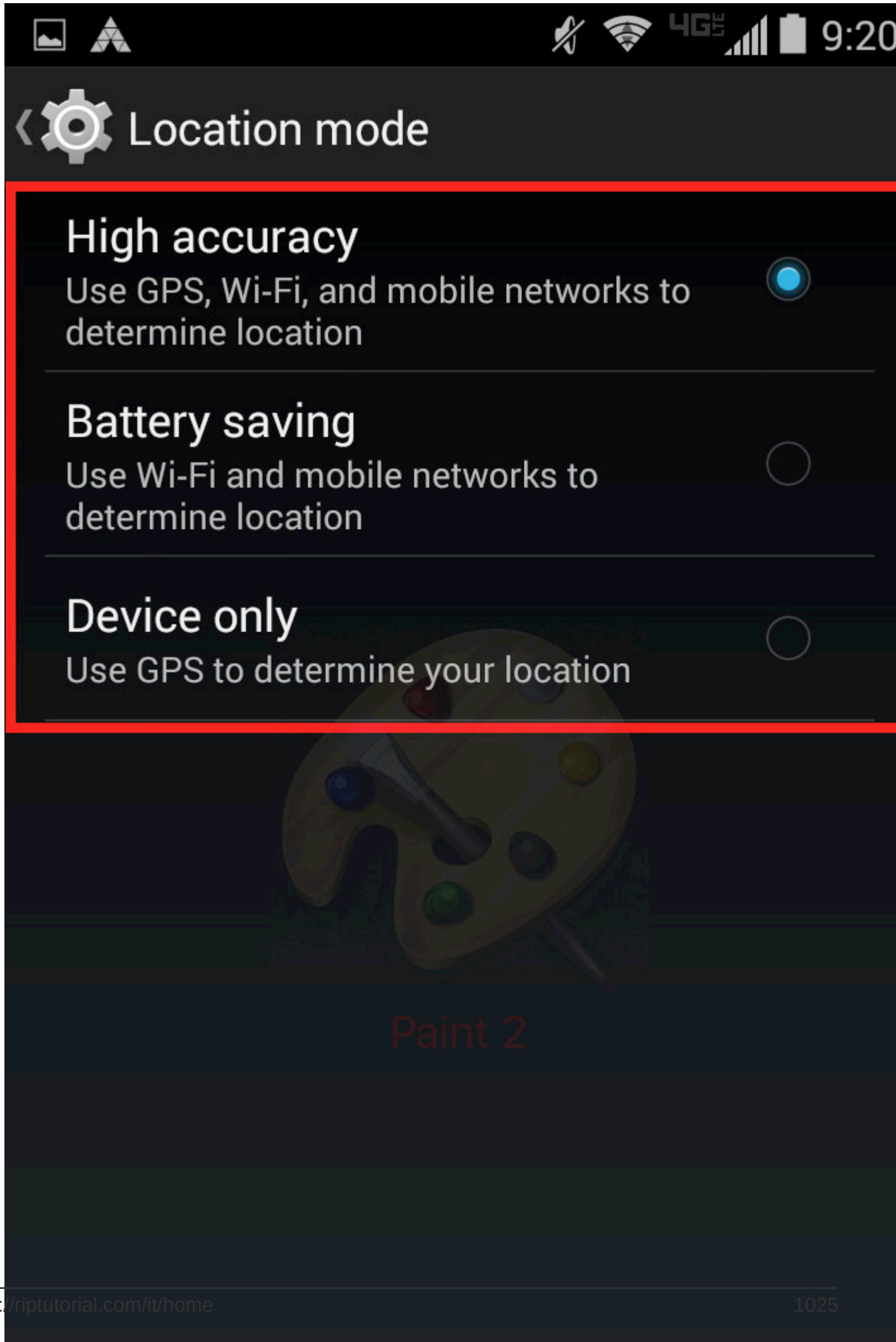
   Ovviamente, assicurati di aver attivato i servizi di localizzazione.


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

Se utilizzi i servizi di rete, hai attivato "Scansione sempre disponibile"? La modalità di localizzazione è impostata su "Migliore" ("Alta precisione") o "Risparmio batteria" ("Solo rete")?


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

Se utilizzi il GPS, hai attivato "Migliore" ("Alta precisione") o "Solo dispositivo" in modalità Posizione?


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

Sì, è qui due volte. Hai provato a utilizzare `LocationListener` invece di `PendingIntent` o viceversa per assicurarti di aver implementato correttamente `LocationManager` ? Sei sicuro che la richiesta di posizione non venga rimossa in una parte del ciclo di vita di attività o servizio che non ti aspettavi di accadere?

o viceversa per assicurarti di aver implementato correttamente `LocationManager` ? Sei sicuro che la richiesta di posizione non venga rimossa in una parte del ciclo di vita di attività o servizio che non ti aspettavi di accadere?

---

6. Controlla i tuoi dintorni!

   Stai testando il GPS al primo piano di un edificio nel centro di San Francisco? Stai testando le posizioni della rete nel bel mezzo del nulla? Lavori in un bunker sotterraneo segreto privo di tutti i segnali radio, chiedendoti perché il tuo dispositivo non ha la posizione? Controlla sempre i dintorni quando cerchi di risolvere i problemi di localizzazione!

Potrebbero esserci molte altre ragioni meno ovvie per cui la posizione non funziona, ma prima di cercare quelle correzioni esoteriche, è sufficiente scorrere questa lista di controllo rapida.

Examples

API di posizione fusa

Esempio di utilizzo dell'attività 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
    }
}
```

Esempio Utilizzo servizio w / PendingIntent e BroadcastReceiver

ExampleActivity

Lettura consigliata: 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
            }
        }
    }
}
```

Servizio di localizzazione

NOTA: non dimenticare di registrare questo servizio nel Manifesto!

```
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;
    }
}
```

LocationReceiver

NOTA: non dimenticare di registrare questo ricevitore nel Manifest!

```
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));
        }
    }
}
```

Richiesta di aggiornamenti di posizione tramite LocationManager

Come sempre, è necessario assicurarsi di disporre delle autorizzazioni necessarie.

```
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) {

    }
}
```

Richiesta di aggiornamenti di posizione su un thread separato utilizzando LocationManager

Come sempre, è necessario assicurarsi di disporre delle autorizzazioni necessarie.

```
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) {

    }
}
```

Registra geofence

Ho creato `GeoFenceObserversationService` classe **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,getGeofencePendingIntent()).setResultCa

        } 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);
        }
    }
```

Dove ho iniziato il servizio? Dalla classe dell'applicazione

- `startService(new Intent(getApplicationContext(),GeoFenceObserversationService.class));`

Come ho registrato Geofences?

- `GeoFenceObserversationService.getInstant().addGeofences();`

Ottieni indirizzo da posizione usando Geocoder

Dopo aver ottenuto il `Location` oggetto dal `FusedAPI` , si può facilmente acquisire `Address` informazioni da quell'oggetto.

```
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;
}
```

Ottenere gli aggiornamenti di posizione in un BroadcastReceiver

Per prima cosa creare una classe BroadcastReceiver per gestire gli aggiornamenti della posizione in arrivo:

```
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
***");
            }
        }
    }
```

```
}
```

Quindi, quando ti connetti a GoogleApiClient nel callback 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);
}
```

Non dimenticare di rimuovere l'intento di aggiornamento della posizione nella callback del ciclo di vita appropriato:

```
@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();
}
```

Leggi Posizione online: https://riptutorial.com/it/android/topic/1837/posizione

Capitolo 185: Processore di annotazione

introduzione

Il processore di annotazione è uno strumento creato in javac per la scansione e l'elaborazione delle annotazioni in fase di compilazione.

Le annotazioni sono una classe di metadati che possono essere associati a classi, metodi, campi e persino altre annotazioni. Esistono due modi per accedere a queste annotazioni in fase di runtime tramite riflessione e in fase di compilazione tramite i processori di annotazione.

Examples

@NonNull Annotazione

```
public class Foo {
    private String name;
    public Foo(@NonNull String name){...};
    ...
}
```

Qui @NonNull è un'annotazione che viene elaborata dal tempo di compilazione da parte di Android Studio per avvertirti che la particolare funzione ha bisogno di un parametro non nullo.

Tipi di annotazioni

Esistono tre tipi di annotazioni.

1. **Annotazione marcatore** : annotazione senza metodo

   ```
   @interface CustomAnnotation {}
   ```

2. **Annotazione a valore singolo** - annotazione con un metodo

   ```
   @interface CustomAnnotation {
       int value();
   }
   ```

3. **Annotazione a più valori** : annotazione con più di un metodo

   ```
   @interface CustomAnnotation{
       int value1();
       String value2();
       String value3();
   }
   ```

Creazione e utilizzo di annotazioni personalizzate

Per creare annotazioni personalizzate dobbiamo decidere

- Target - su cui queste annotazioni funzioneranno come il livello del campo, il livello del metodo, il livello del tipo ecc.
- Conservazione - a quale livello di annotazione sarà disponibile.

Per questo, abbiamo creato annotazioni personalizzate. Dai un'occhiata a quelli usati principalmente:

@Bersaglio

Element Types	Where the a
TYPE	class, interfac
FIELD	fields
METHOD	methods
CONSTRUCTOR	constructors
LOCAL_VARIABLE	local variable
ANNOTATION_TYPE	annotation ty
PARAMETER	parameter

@Ritenzione

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

Creazione di annotazioni personalizzate

```
@Retention(RetentionPolicy.SOURCE) // will not be available in compiled class
@Target(ElementType.METHOD) // can be applied to methods only
@interface CustomAnnotation{
      int value();
}
```

Utilizzo di annotazioni personalizzate

```
class Foo{
  @CustomAnnotation(value = 1)  // will be used by an annotation processor
  public void foo(){..}
}
```

il valore fornito in `@CustomAnnotation` verrà utilizzato da un processo di annotazione per generare codice in fase di compilazione, ecc.

Leggi Processore di annotazione online: https://riptutorial.com/it/android/topic/10726/processore-di-annotazione

Capitolo 186: Programmazione Android con Kotlin

introduzione

Usare Kotlin con Android Studio è un compito facile dato che Kotlin è sviluppato da JetBrains. È la stessa società che sta dietro IntelliJ IDEA - un IDE di base per Android Studio. Questo è il motivo per cui non ci sono quasi problemi con la compatibilità.

Osservazioni

Se vuoi saperne di più su Kotlin Programming Language, consulta la documentazione .

Examples

Installazione del plugin Kotlin

Innanzitutto, devi installare il plug-in di Kotlin.

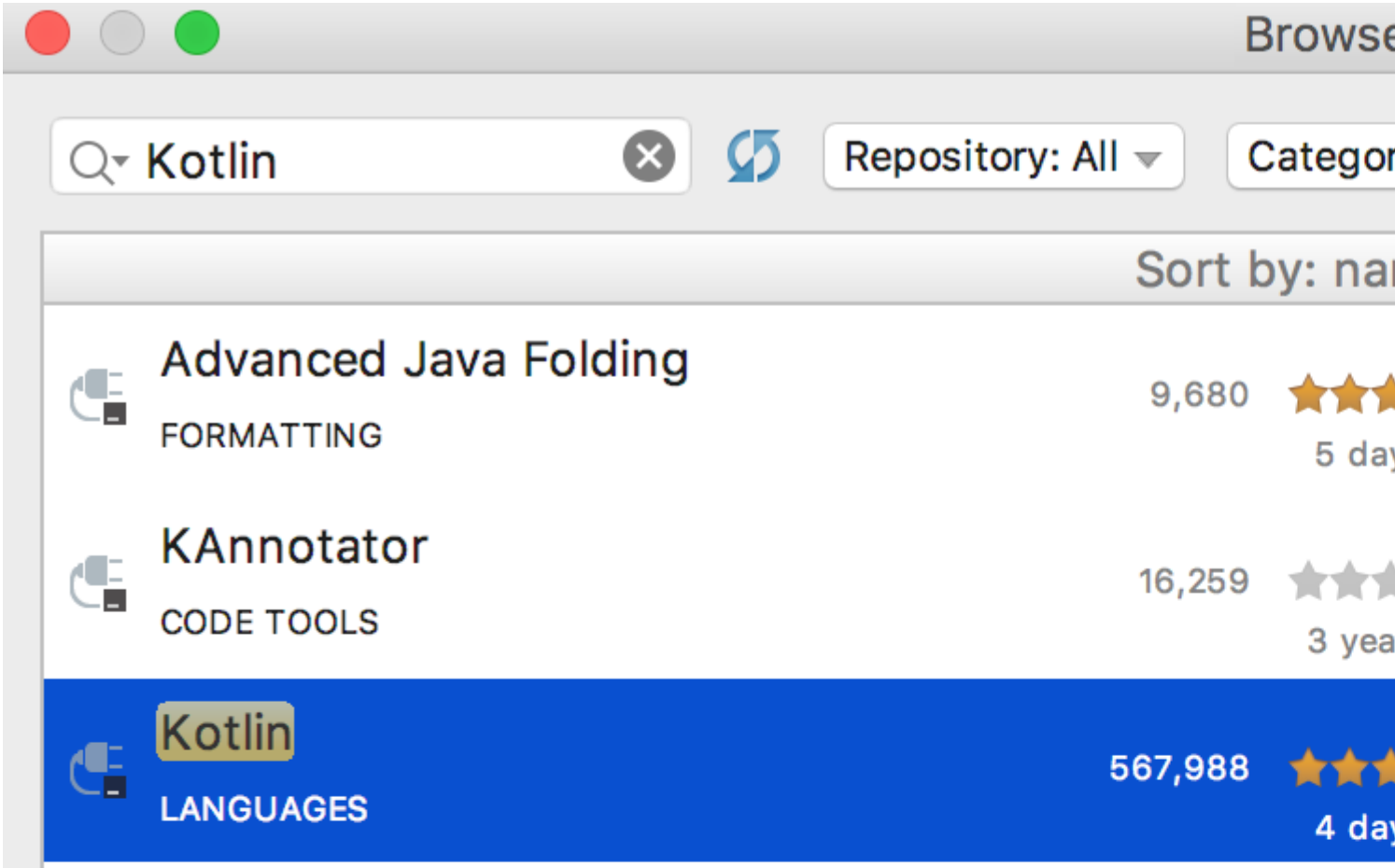
Per Windows:

- Passare a `File` → `Settings` → `Plugins` → `Install JetBrains plugin`

Per Mac:

- Passare ad `Android Studio` → `Preferences` → `Plugins` → `Install JetBrains plugin`

E poi cerca e installa Kotlin. È necessario riavviare l'IDE al termine.


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

e quindi aggiungere il supporto Kotlin ad esso o modificare il tuo progetto esistente. Per farlo, devi:

1. **Aggiungi dipendenza a un file gradle radice** - devi aggiungere la dipendenza per il plugin `kotlin-android` a un file root `build.gradle` .

```
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. **Applicare Kotlin Android Plugin** : è sufficiente aggiungere il `apply plugin: 'kotlin-android'` a un file `build.gradle` modulo.

3. **Aggiungi dipendenza a Kotlin stdlib** - aggiungi la dipendenza a `'org.jetbrains.kotlin:kotlin-stdlib:1.1.2'` alla sezione delle dipendenze in un file `build.gradle` modulo.

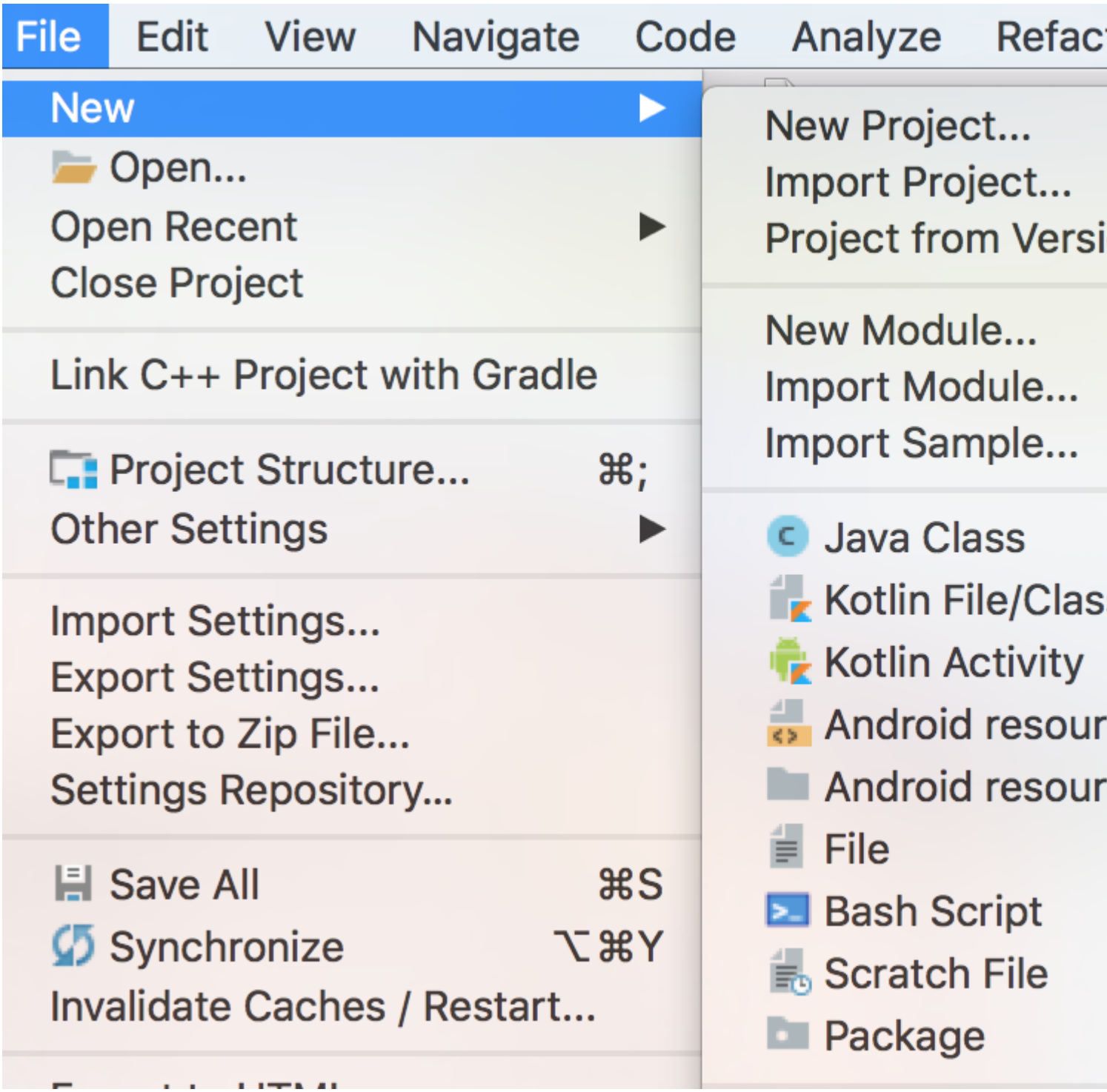
Per un nuovo progetto, il file `build.gradle` potrebbe avere il seguente aspetto:

```
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'
}
```

Creare una nuova attività di Kotlin

1. Fare clic su `File` → `New` → `Kotlin Activity` .
2. Scegli un tipo di attività.
3. Seleziona il nome e altri parametri per l'attività.
4. Finire.

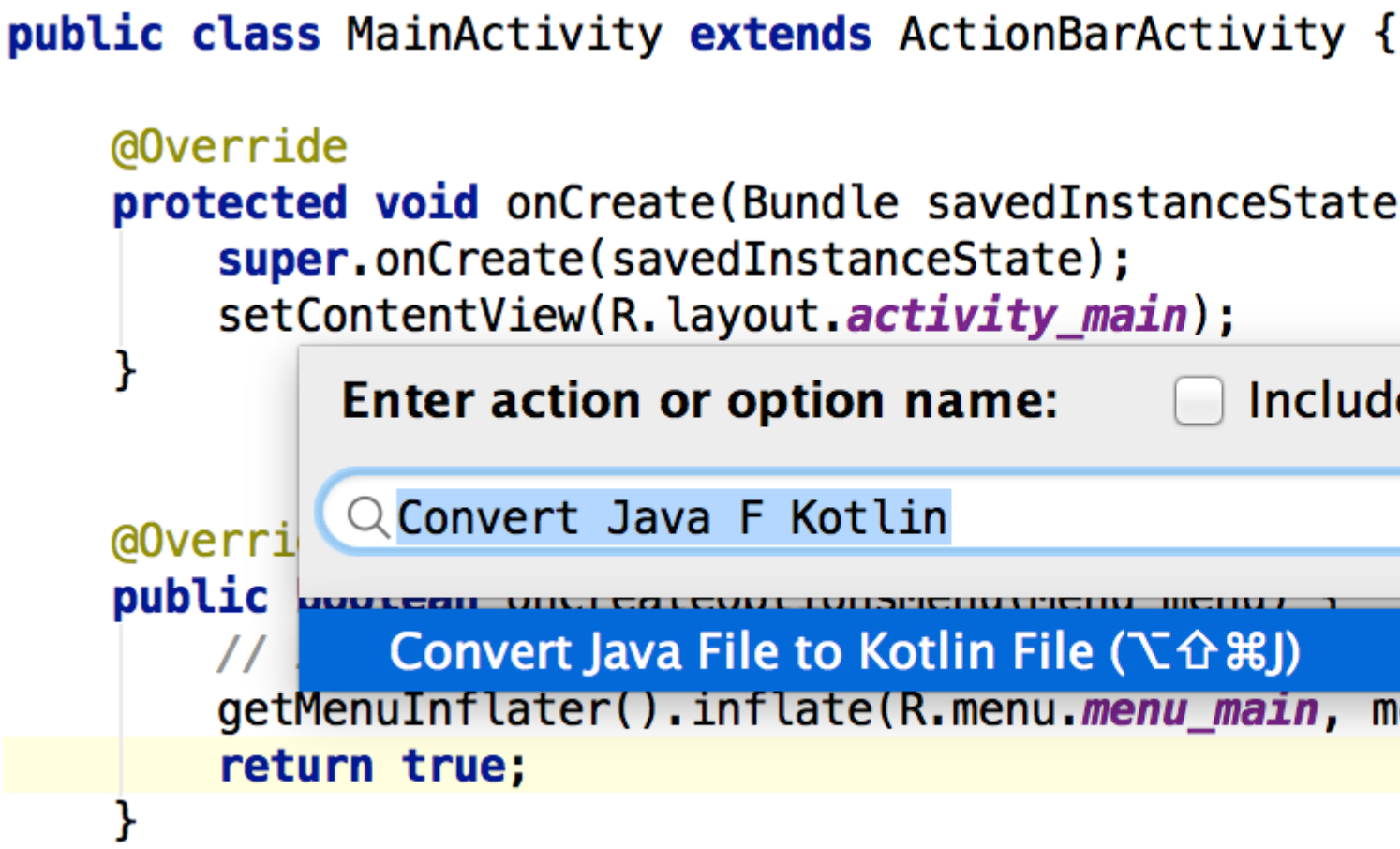


La classe finale potrebbe assomigliare a questa:

```
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)
    }
}
```

Conversione del codice Java esistente in Kotlin

Kotlin Plugin per Android Studio supporta la conversione di file Java esistenti in file Kotlin. Scegli un file Java e invoca l'azione Converti file Java in file Kotlin:



Avvio di una nuova attività

```
fun startNewActivity(){
    val intent: Intent = Intent(context, Activity::class.java)
    startActivity(intent)
}
```

È possibile aggiungere extra all'intento proprio come in Java.

```
fun startNewActivityWithIntents(){
    val intent: Intent = Intent(context, Activity::class.java)
    intent.putExtra(KEY_NAME, KEY_VALUE)
    startActivity(intent)
}
```

Leggi Programmazione Android con Kotlin online:
https://riptutorial.com/it/android/topic/9623/programmazione-android-con-kotlin

Capitolo 187: Programmazione del lavoro

Osservazioni

Attenzione ad eseguire un sacco di codice o fare lavori pesanti all'interno di `JobService` , ad esempio in `onStartJob()` . Il codice verrà eseguito sul thread **principale / dell'interfaccia utente** e pertanto può portare a un'interfaccia utente bloccata, all'app non più rispondente o addirittura a un arresto anomalo della tua app!

Per questo `AsyncTask` , è necessario scaricare il lavoro, ad esempio utilizzando un `Thread` o `AsyncTask` .

Examples

Utilizzo di base

Crea un nuovo JobService

Questo viene fatto estendendo la classe `JobService` e implementando / sovrascrivendo i metodi richiesti `onStartJob()` e `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;
    }
}
```

Aggiungi il nuovo JobService al tuo AndroidManifest.xml

Il seguente passaggio è *obbligatorio* , altrimenti non sarai in grado di eseguire il tuo lavoro:

Dichiara la tua classe `MyJobService` come un nuovo elemento `<service>` tra `<application> </application>` nel tuo *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>
```

Imposta ed esegui il lavoro

Dopo aver implementato un nuovo JobService e aggiunto al tuo *AndroidManifest.xml* , puoi continuare con i passaggi finali.

- `onButtonClick_startJob()` prepara ed esegue un lavoro periodico. Oltre ai lavori periodici, `JobInfo.Builder` consente di specificare molte altre impostazioni e vincoli. Ad esempio, è possibile definire che è necessario un *caricatore collegato* o una *connessione di rete* per eseguire il lavoro.
- `onButtonClick_stopJob()` annulla tutti i lavori in esecuzione

```
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
    }
}
```

Dopo aver chiamato `onButtonClick_startJob()` , il processo verrà eseguito approssimativamente a intervalli di 10 secondi, anche quando l'app è in stato di *pausa* (l'utente ha premuto il pulsante home e l'app non è più visibile).

Anziché annullare tutti i processi in esecuzione all'interno di `onButtonClick_stopJob()` , è anche possibile chiamare `jobScheduler.cancel()` per annullare un lavoro specifico in base all'ID lavoro.

Leggi Programmazione del lavoro online:
https://riptutorial.com/it/android/topic/6907/programmazione-del-lavoro

Capitolo 188: ProGuard: offuscamento e riduzione del codice

Examples

Regole per alcune delle librerie ampiamente utilizzate

Attualmente contiene regole per le seguenti librerie: -

1. Coltello da burro
2. RxJava
3. Libreria di supporto Android
4. Libreria di supporto alla progettazione Android
5. Retrofit
6. Gson e Jackson
7. Otto
8. Crashlitycs
9. Picasso
10. raffica
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;
}
```

Abilita ProGuard per la tua build

Per abilitare le configurazioni di `ProGuard` per la tua applicazione devi abilitarlo nel tuo file gradle di livello del modulo. è necessario impostare il valore di `minifyEnabled true` .

È inoltre possibile abilitare `shrinkResources true` che rimuoverà le risorse contrassegnate da `ProGuard` come non utilizzate.

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

Il codice sopra riportato applicherà le tue configurazioni ProGuard contenute in `proguard-rules.pro` ("proguard-project.txt" in Eclipse) al tuo apk rilasciato.

Per consentire di determinare in seguito la riga su cui si è verificata un'eccezione in una traccia stack, "proguard-rules.pro" deve contenere le seguenti righe:

```
-renamesourcefileattribute SourceFile
-keepattributes SourceFile,LineNumberTable
```

Per abilitare Proguard in Eclipse, aggiungere `proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt` a "project.properties"

Rimuovere le istruzioni di registrazione traccia (e altre) al momento della compilazione

Se vuoi rimuovere le chiamate a determinati metodi, supponendo che restituiscano nulla e che non abbiano effetti collaterali (come in, chiamarli non cambia alcun valore di sistema, argomenti di riferimento, statiche, ecc.) Allora puoi fare in modo che ProGuard li rimuova dal output dopo il completamento della compilazione.

Ad esempio, trovo questo utile nella rimozione delle istruzioni di registrazione debug / verbose utili nel debug, ma la produzione di stringhe per loro non è necessaria in produzione.

```
# 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: se `-dontoptimize` viene utilizzato in qualsiasi configurazione di ProGuard in modo che non stia minimizzando / rimuovendo il codice inutilizzato, questo non eliminerà le dichiarazioni. (Ma chi non vorrebbe rimuovere il codice inutilizzato, giusto?)

Nota 2: questa chiamata rimuoverà la chiamata da registrare, ma non proteggerà il tuo codice. Le stringhe rimarranno effettivamente nell'apk generato. Leggi di più in questo post .

Proteggi il tuo codice dagli hacker

L'offuscamento è spesso considerato come una soluzione magica per la protezione del codice, rendendo più difficile capire il codice se viene mai decompilato dagli hacker.

Ma se stai pensando che rimuovere `Log.x(..)` realtà rimuova le informazioni di cui hanno bisogno gli hacker, avrai una brutta sorpresa.

Rimozione di tutte le tue chiamate di registro con:

```
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    ...etc
}
```

rimuoverà effettivamente la chiamata Log stessa, ma di solito *non* le Stringhe che hai inserito.

Se, ad esempio, all'interno della tua registrazione, digiti un messaggio di log comune come: `Log.d(MyTag,"Score="+score);` , il compilatore converte il `+` in un 'nuovo StringBuilder ()' all'esterno della chiamata Log. ProGuard non cambia questo nuovo oggetto.

Il tuo codice de-compilato avrà ancora un `StringBuilder` sospeso per `"Score="` , aggiunto alla versione offuscata per la variabile `score` (diciamo che è stata convertita in `b`).
Ora l'hacker sa cosa è `b` e ha senso del tuo codice.

Una buona pratica per rimuovere effettivamente questi residui dal tuo codice è o non metterli lì al primo posto (usa invece String formattatore, con le regole proguard per rimuoverli), o per

avvolgere le tue chiamate di `Log` con:

```
    if (BuildConfig.DEBUG) {
        Log.d(TAG,".."+var);
    }
```

Mancia:

Verifica quanto è ben protetto il tuo codice offuscato decompilando te stesso!

1. dex2jar - converte l'apk in jar
2. jd - decompila il jar e lo apre in un editor di gui

Abilitazione di ProGuard con un file di configurazione di offuscamento personalizzato

ProGuard consente allo sviluppatore di offuscare, ridurre e ottimizzare il suo codice.

 # 1 Il primo passo della procedura è di abilitare proguard sulla build .

Questo può essere fatto **impostando il comando 'minifyEnabled' su true** sulla build desiderata

 # 2 Il secondo passo è specificare quali file proguard utilizziamo per la build data

Questo può essere fatto **impostando la linea 'proguardFiles' con i nomi file appropriati**

```
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 Lo sviluppatore può quindi modificare il suo file proguard con le regole che desidera.

Ciò può essere fatto modificando il file (ad esempio 'proguard-rules-tests.pro') e aggiungendo i vincoli desiderati. *Il seguente file serve come esempio di file 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
```

Infine, ogni volta che lo sviluppatore esegue e / o genera il suo nuovo file .APK, verranno applicate le configurazioni proguard personalizzate in modo da soddisfare i requisiti.

Leggi ProGuard: offuscamento e riduzione del codice online:
https://riptutorial.com/it/android/topic/4500/proguard--offuscamento-e-riduzione-del-codice

Capitolo 189: Pubblica il file .aar su Apache Archiva con Gradle

Examples

Semplice esempio di implementazione

```
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'
            }
        }
    }

}
```

Leggi Pubblica il file .aar su Apache Archiva con Gradle online:
https://riptutorial.com/it/android/topic/6453/pubblica-il-file--aar-su-apache-archiva-con-gradle

Capitolo 190: Pubblica su Play Store

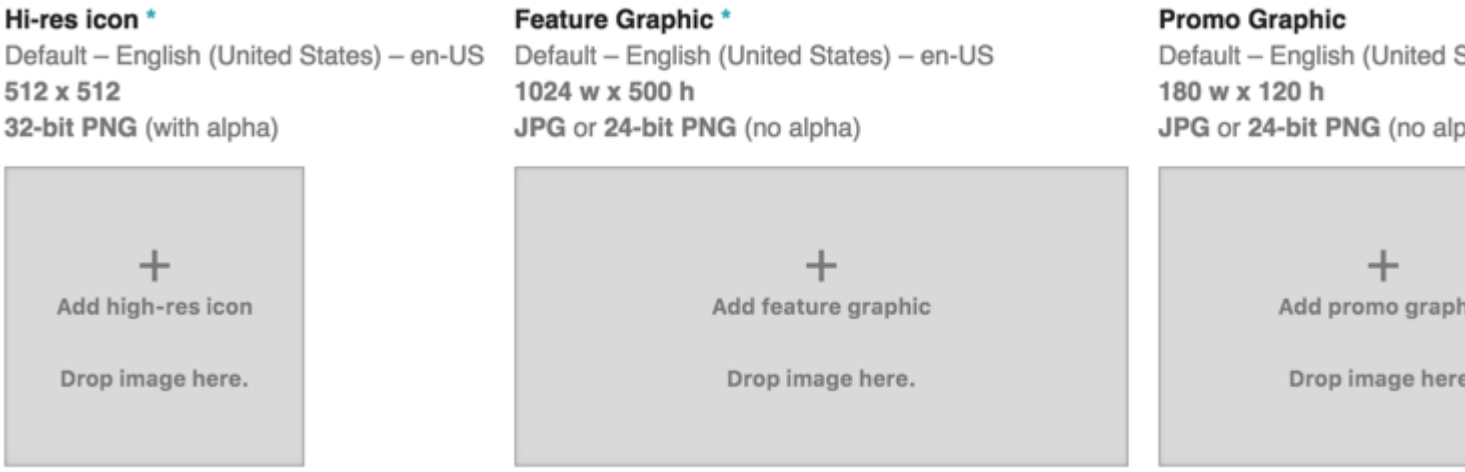
Examples

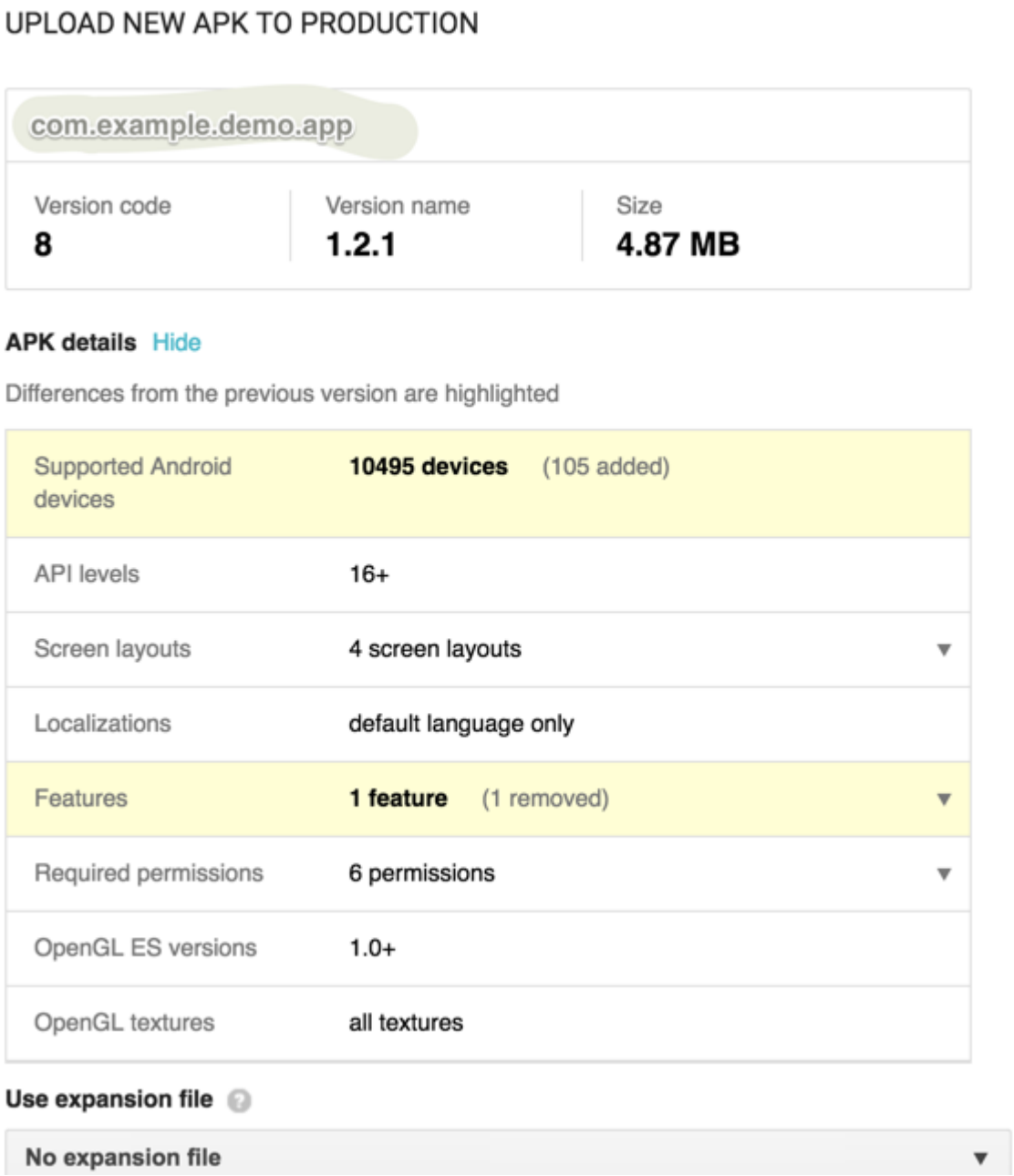
Guida minima alla presentazione delle app

Requisiti:

- Un account sviluppatore
- Un apk già costruito e firmato con una chiave non di debug
- Un'app gratuita che non ha fatturazione in-app
- no Firebase Cloud Messaging o servizi di gioco

1. Vai a https://play.google.com/apps/publish/
   1a) Crea il tuo account sviluppatore se non ne hai uno
2. Fare clic sul pulsante `Create new Application`
3. Fai clic su Invia APK
4. Compila tutti i campi obbligatori nel modulo, comprese alcune risorse che verranno visualizzate sul Play Store (vedi immagine sotto)
5. Quando è soddisfatto, `Publish app` clic sul pulsante `Publish app`





Ulteriori informazioni sull'accesso a Configura impostazioni di firma

Leggi Pubblica su Play Store online: https://riptutorial.com/it/android/topic/5369/pubblica-su-play-store

Capitolo 191: Pubblica una libreria per i repository di Maven

Examples

Pubblica il file .aar su Maven

Per poter pubblicare su un repository in formato Maven, è possibile utilizzare il plugin "maven-publish" per gradle.

Il plugin dovrebbe essere aggiunto al file `build.gradle` nel modulo della libreria.

```
apply plugin: 'maven-publish'
```

È necessario definire la pubblicazione e i relativi attributi di identità anche nel file `build.gradle` . Questi attributi di identità saranno mostrati nel file pom generato e in futuro per l'importazione di questa pubblicazione li userai. Devi anche definire quali artefatti vuoi pubblicare, per esempio voglio solo pubblicare il file .aar generato dopo aver costruito la libreria .

```
publishing {
    publications {
        myPulication(MavenPublication) {
            groupId 'com.example.project'
            version '1.0.2'
            artifactId 'myProject'
            artifact("$buildDir/outputs/aar/myProject.aar")
        }
    }
}
```

Dovrai anche definire l'URL del tuo repository

```
publishing{
    repositories {
        maven {
            url "http://www.myrepository.com"
        }
    }
}
```

Qui è il file `build.gradle` libreria `build.gradle`

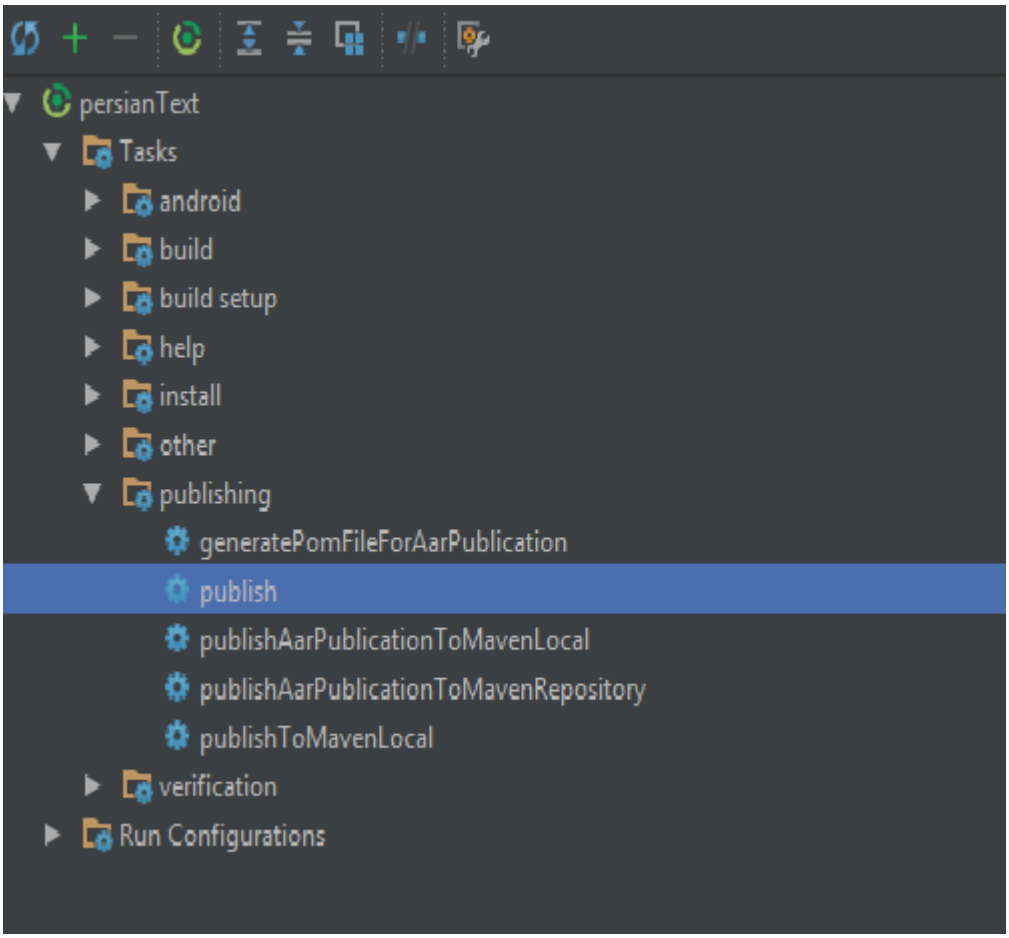
```
apply plugin: 'com.android.library'
apply plugin: 'maven-publish'

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"
        }
    }
}
```

Per la pubblicazione è possibile eseguire il comando gradle console

gradle pubblicare

oppure puoi eseguire dal pannello delle attività gradle



Leggi Pubblica una libreria per i repository di Maven online:
https://riptutorial.com/it/android/topic/9359/pubblica-una-libreria-per-i-repository-di-maven

Capitolo 192: Pugnale 2

Sintassi

- @Modulo
- @Component (dependencies = {OtherComponent.class}, modules = {ModuleA.class, ModuleB.class})
- DaggerMyComponent.create ()
- DaggerMyComponent.builder (). Mymodule (newMyModule ()). Create ()

Osservazioni

Non confondere con il pugnale di quadrato, il predecessore del pugnale 2.

Examples

Configurazione dei componenti per l'iniezione di applicazioni e attività

Un `AppComponent` base che dipende da un singolo `AppModule` per fornire oggetti singleton a livello di applicazione.

```
@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {

    void inject(App app);

    Context provideContext();

    Gson provideGson();
}
```

Un modulo da utilizzare insieme a `AppComponent` che fornirà i suoi oggetti singleton, ad esempio un'istanza di `Gson` da riutilizzare nell'intera applicazione.

```
@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;
    }
}
```

Un'applicazione sottoclassata per l'impostazione di dagger e del 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;
    }
}
```

Ora un componente con ambito attività che dipende da `AppComponent` per accedere agli oggetti Singleton.

```
@ActivityScope
@Component(dependencies = AppComponent.class, modules = ActivityModule.class)
public interface MainActivityComponent {

    void inject(MainActivity activity);
}
```

E un `ActivityModule` riutilizzabile che fornirà dipendenze di base, come 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();
```

```
    }
}
```

Mettendo tutto insieme siamo pronti e possiamo iniettare la nostra attività ed essere sicuri di usare lo stesso `Gson` tutta l'app!

```
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);
    }
}
```

Ambiti personalizzati

```
@Scope
@Documented
@Retention(RUNTIME)
public @interface ActivityScope {
}
```

Gli ambiti sono solo annotazioni e puoi crearne di tuoi laddove necessario.

Costruttore di iniezione

Le classi senza dipendenze possono essere facilmente create dal pugnale.

```
public class Engine {

    @Inject // <-- Annotate your constructor.
    public Engine() {
    }
}
```

Questa classe può essere fornita da *qualsiasi* componente. Non ha *dipendenze* e *non ha ambito* . Non è necessario alcun altro codice.

Le dipendenze sono dichiarate come parametri nel costruttore. Dagger chiamerà il costruttore e fornirà le dipendenze, purché tali dipendenze possano essere fornite.

```
public class Car {
```

```
    private Engine engine;

    @Inject
    public Car(Engine engine) {
        this.engine = engine;
    }
}
```

Questa classe può essere fornita da ogni componente *se* questo componente può anche fornire tutte le sue dipendenze - `Engine` in questo caso. Poiché `Engine` può anche essere iniettato dal costruttore, *qualsiasi* componente può fornire `Car` .

È possibile utilizzare l'iniezione del costruttore ogni volta che tutte le dipendenze possono essere fornite dal componente. Un componente può fornire una dipendenza, se

- può crearlo usando l'iniezione del costruttore
- un modulo del componente può fornirlo
- può essere fornito dal componente principale (se è un `@Subcomponent`)
- può usare un oggetto esposto da un componente da cui dipende (dipendenze dei componenti)

Utilizzo di @Subcomponent anziché di @Component (dependencies = {...})

```
@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);
    }
}
```

Come aggiungere Dagger 2 in build.gradle

Dal rilascio di Gradle 2.2, l'uso del plugin per Android-apt non è più utilizzato. Deve essere usato il seguente metodo di impostazione di Dagger 2. Per la versione precedente di Gradle, utilizzare il metodo precedente mostrato di seguito.

Per 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'
}
```

Per Gradle <2.2

Per utilizzare Dagger 2 è necessario aggiungere il plugin per `android-apt` , aggiungere questo alla root build.gradle:

```
buildscript {
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.0'
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}
```

Quindi il build.gradle del modulo dell'applicazione dovrebbe contenere:

```
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}"
}
```

Riferimento: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Creazione di un componente da più moduli

Dagger 2 supporta la creazione di un componente da più moduli. Puoi creare il tuo componente in questo modo:

```
@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);
}
```

I due moduli di riferimento `GeneralPurposeModule` e `SpecificModule` possono quindi essere implementati come segue:

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 di iniezione delle dipendenze, il componente prenderà gli oggetti da entrambi i

moduli in base alle esigenze.

Questo approccio è molto utile in termini di *modularità* . Nell'esempio, esiste un modulo generico utilizzato per istanziare componenti come l'oggetto `Retrofit` (utilizzato per gestire le comunicazioni di rete) e un `PropertiesReader` (incaricato di gestire i file di configurazione). Esiste anche un modulo specifico che gestisce l'istanziazione di specifici controller e classi di servizio in relazione a quel componente specifico dell'applicazione.

Leggi Pugnale 2 online: https://riptutorial.com/it/android/topic/3088/pugnale-2

Capitolo 193: Pulsante

Sintassi

- <Pulsante ... />
- Android: onClick = "methodname"
- button.setOnClickListener (new OnClickListener () {...});
- classname di classe public implementa View.OnLongClickListener

Examples

in linea onClickListener

Supponiamo di avere un pulsante (possiamo crearlo a livello di codice o collegarlo da una vista usando findViewbyId (), ecc ...)

```
Button btnOK = (...)
```

Ora crea una classe anonima e impostala in linea.

```
btnOk.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Do stuff here...
    }
});
```

Utilizzando il layout per definire un'azione di clic

Quando creiamo un pulsante nel layout, possiamo utilizzare l'attributo `android:onClick` per fare riferimento a un metodo nel codice per gestire i clic.

Pulsante

```
<Button
    android:width="120dp"
    android:height="wrap_content"
    android:text="Click me"
    android:onClick="handleClick" />
```

Quindi, nella tua attività, crea il metodo `handleClick` .

```
public void handleClick(View v) {
    // Do whatever.
}
```

Utilizzando lo stesso evento click per una o più viste nell'XML

Quando creiamo qualsiasi vista nel layout, possiamo utilizzare l'attributo android: onClick per fare riferimento a un metodo nell'attività associata o nel frammento per gestire gli eventi di clic.

Layout 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" />
```

Codice attività / frammento

Nel **codice** , crea il metodo che hai chiamato, dove v sarà la vista che è stata toccata e fai qualcosa per ogni vista che chiama questo metodo.

```
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;
    }
}
```

Se lo desideri, puoi anche utilizzare un metodo diverso per ciascuna vista (in questo caso, ovviamente, non è necessario verificare l'ID).

Ascoltando gli eventi di clic lungo

Per catturare un clic lungo e utilizzarlo è necessario fornire un listener appropriato al pulsante:

```
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);
```

Definizione di listener esterno

Quando dovrei usarlo

- Quando il codice all'interno di un listener in linea è troppo grande e il tuo metodo / classe diventa brutto e difficile da leggere
- Vuoi eseguire la stessa azione in vari elementi (vista) della tua app

---

Per ottenere ciò è necessario creare una classe che implementa uno degli ascoltatori nell'API di visualizzazione .

Ad esempio, dai aiuto quando fai un clic lungo su qualsiasi elemento:

```
public class HelpLongClickListener implements View.OnLongClickListener
{
    public HelpLongClickListener() {
    }

    @Override
    public void onLongClick(View v) {
        // show help toast or popup
    }
}
```

Quindi devi solo avere un attributo o una variabile nella tua `Activity` per usarlo:

```
HelpLongClickListener helpListener = new HelpLongClickListener(...);

button1.setOnClickListener(helpListener);
button2.setOnClickListener(helpListener);
label.setOnClickListener(helpListener);
button1.setOnClickListener(helpListener);
```

NOTA: la definizione dei listener in una classe separata presenta uno svantaggio, non è possibile accedere direttamente ai campi della classe, quindi è necessario passare dati (contesto, vista) tramite il costruttore a meno che non si rendano pubblici gli attributi o si definiscano i geters.

Listener di clic personalizzato per impedire più clic veloci

Per **evitare che un pulsante venga attivato più volte in un breve periodo di tempo** (diciamo 2 clic in 1 secondo, il che potrebbe causare seri problemi se il flusso non viene controllato), è possibile implementare un ***SingleClickListener*** personalizzato.

Questo ClickListener imposta un intervallo di tempo specifico come soglia (ad esempio, 1000 ms).

Quando si fa clic sul pulsante, verrà eseguito un controllo per verificare se il trigger è stato eseguito nell'ultimo periodo di tempo definito e, in caso contrario, verrà attivato.

```
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);

}
```

E nella classe, SingleClickListener è associato al pulsante in gioco

```
myButton = (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(new SingleClickListener() {
    @Override
    public void performClick(View view) {
        // do stuff
    }
});
```

Personalizzazione dello stile del pulsante

Ci sono molti modi possibili per personalizzare l'aspetto di un pulsante. Questo esempio presenta diverse opzioni:

Opzione 0: Usa ThemeOverlay (attualmente il modo più semplice / più veloce)

Crea un nuovo stile nel tuo file di stili:

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>
```

Quindi nel layout in cui si posiziona il pulsante (ad es. 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>
```

Opzione 1: crea il tuo stile di pulsante

In values / styles.xml, crea un nuovo stile per il tuo pulsante:

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>
```

Quindi nel layout in cui si posiziona il pulsante (ad es. In 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>
```

Opzione 2: assegna un drawable per ciascuno degli stati dei tuoi pulsanti

Creare un file xml in una cartella disegnabile denominata "mybuttondrawable.xml" per definire la risorsa estraibile di ciascuno degli stati dei pulsanti:

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>
```

Ciascuno di questi drawable possono essere immagini (ad es. Mybutton_disabled.png) o file xml definiti dall'utente e memorizzati nella cartella dei disegni. Per esempio:

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>
```

Quindi nel layout in cui si posiziona il pulsante (ad es. 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>
```

Opzione 3: aggiungi lo stile del tuo pulsante al tema dell'app

Puoi sovrascrivere lo stile del pulsante Android predefinito nella definizione del tema dell'app (in 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>
```

Opzione 4: sovrapponi un colore allo stile di pulsante predefinito in modo programmatico

Basta trovare il pulsante nella tua attività e applicare un filtro colore:

```
Button mybutton = (Button) findViewById(R.id.mybutton);
mybutton.getBackground().setColorFilter(anycolor, PorterDuff.Mode.MULTIPLY)
```

È possibile controllare diversi metodi di fusione qui ed esempi piacevoli qui .

Leggi Pulsante online: https://riptutorial.com/it/android/topic/5607/pulsante

Capitolo 194: Pulsante hardware Eventi / Intenti (PTT, LWP, ecc.)

introduzione

Diversi dispositivi Android hanno pulsanti personalizzati aggiunti dal produttore. Questo apre nuove possibilità per lo sviluppatore nella gestione di quei pulsanti, specialmente quando si rendono le app destinate ai dispositivi hardware.

Questo argomento documenta i pulsanti con gli intent che possono essere ascoltati tramite intent-receiver.

Examples

Dispositivi Sonim

I dispositivi Sonim hanno diversi modelli personalizzati per modello:

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
```

Registrazione dei pulsanti

Per ricevere tali intenti dovrai assegnare i pulsanti alla tua app in Impostazioni telefono. Sonim ha la possibilità di registrare automaticamente i pulsanti sull'app quando viene installata. Per farlo dovrai contattarli e ottenere una chiave specifica per il pacchetto da includere nel tuo Manifest in questo modo:

```
<meta-data
    android:name="app_key_green_data"
    android:value="your-key-here" />
```

Dispositivi RugGear

Pulsante PTT

```
android.intent.action.PTT.down
android.intent.action.PTT.up
```

Confermato su: RG730, RG740A

Leggi Pulsante hardware Eventi / Intenti (PTT, LWP, ecc.) online:
https://riptutorial.com/it/android/topic/10418/pulsante-hardware-eventi---intenti--ptt--lwp--ecc--

Capitolo 195: Raccoglitori di data e ora

Examples

DatePicker materiale

aggiungere le dipendenze sottostanti al file `build.gradle` nella sezione delle dipendenze. (questa è una libreria non ufficiale per il selezionatore di date)

```
  compile 'com.wdullaer:materialdatetimepicker:2.3.0'
```

Ora dobbiamo aprire `DatePicker` sull'evento Click del pulsante.

Quindi crea un pulsante sul file xml come di seguito.

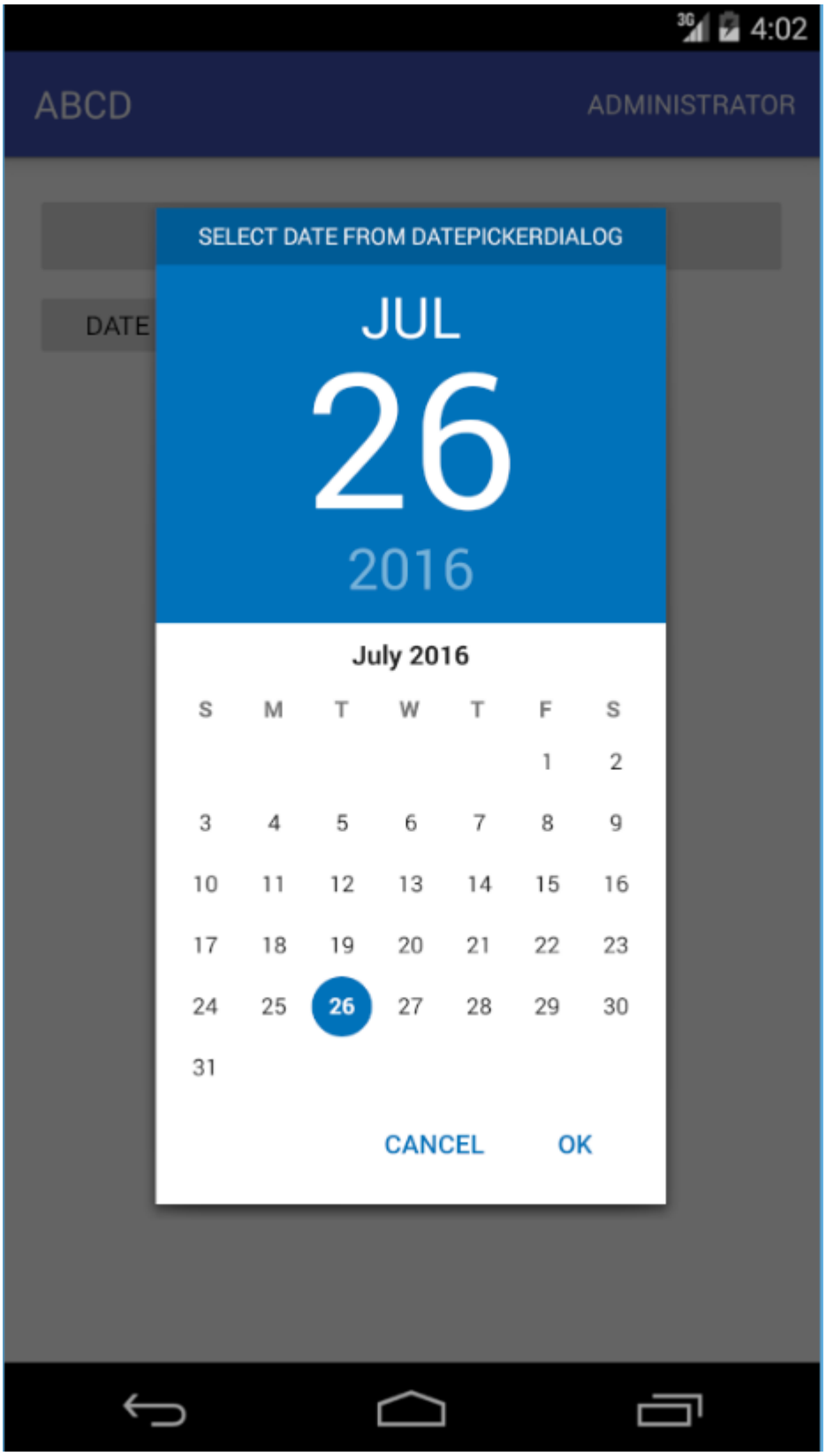
```
     <Button
            android:id="@+id/dialog_bt_date"
            android:layout_below="@+id/resetButton"
            android:layout_width="wrap_content"
            android:layout_height="40dp"
            android:textColor="#FF000000"
            android:gravity="center"
            android:text="DATE"/>
```

e in MainActivity usare in questo modo.

```
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;
    }

}
```

Produzione :



Finestra di selezione data

Si tratta di una finestra di dialogo che richiede all'utente di selezionare la data utilizzando `DatePicker` . La finestra di dialogo richiede il contesto, l'anno iniziale, il mese e il giorno per mostrare la finestra di dialogo con la data di inizio. Quando l'utente seleziona la data richiamata tramite `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();

    }
```

Si noti che il mese è un int che inizia da 0 per gennaio a 11 per dicembre

Leggi Raccoglitori di data e ora online: https://riptutorial.com/it/android/topic/2836/raccoglitori-di-data-e-ora

Capitolo 196: raffica

introduzione

Volley è una libreria HTTP Android che è stata introdotta da Google per rendere le chiamate in rete molto più semplici. Di default tutte le chiamate di rete di Volley sono eseguite in modo asincrono, gestendo tutto in un thread in background e restituendo i risultati in primo piano con l'uso di callback. Poiché il recupero dei dati su una rete è una delle attività più comuni eseguite in qualsiasi app, la libreria Volley è stata creata per facilitare lo sviluppo di app per Android.

Sintassi

- RequestQueue queue = Volley.newRequestQueue (context); // imposta la coda
- Richiesta richiesta = new SomeKindOfRequestClass (Request.Method, String url, Response.Listener, Response.ErrorListener); // imposta un qualche tipo di richiesta, il tipo esatto e gli argomenti cambiano per ogni tipo di richiesta
- queue.add (richiesta); // aggiungi la richiesta alla coda; il listener di risposta appropriato verrà chiamato una volta che la richiesta è terminata (o terminata per qualsiasi motivo)

Osservazioni

Installazione

Puoi costruire Volley dal codice sorgente ufficiale di Google . Per un po ', quella era l'unica opzione. O utilizzando una delle versioni pre-costruite di terze parti. Tuttavia, Google ha finalmente rilasciato un pacchetto Maven ufficiale su jcenter.

Nel file `build.gradle` livello di `build.gradle` , aggiungilo all'elenco delle dipendenze:

```
dependencies {
    ...
    compile 'com.android.volley:volley:1.0.0'
}
```

Assicurati che l'autorizzazione `INTERNET` sia impostata nel manifest dell'app:

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

Documentazione ufficiale

Google non ha fornito una documentazione molto ampia su questa libreria e non l'ha mai toccata da anni. Ma ciò che è disponibile può essere trovato a:

https://developer.android.com/training/volley/index.html

C'è una documentazione non ufficiale ospitata su GitHub, sebbene ci dovrebbe essere una posizione migliore per ospitare questo in futuro:

https://pablobaxter.github.io/volley-docs/

Examples

Basic StringRequest che utilizza il metodo 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);
```

Annulla una richiesta

```
// 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);
```

Aggiunta di attributi del tempo di progettazione personalizzati a NetworkImageView

Esistono diversi attributi aggiuntivi che Volley `NetworkImageView` aggiunge allo standard `ImageView` . Tuttavia, questi attributi possono essere impostati solo nel codice. Di seguito è riportato un esempio di come creare una classe di estensione che raccoglierà gli attributi dal file di layout XML e li `NetworkImageView` all'istanza `NetworkImageView` .

Nella tua directory `~/res/xml` , aggiungi un file chiamato `attrx.xml` :

```
<resources>
    <declare-styleable name="MoreNetworkImageView">
        <attr name="defaultImageResId" format="reference"/>
        <attr name="errorImageResId" format="reference"/>
    </declare-styleable>
</resources>
```

Aggiungi un nuovo file di classe al tuo progetto:

```
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 file di layout di esempio che mostra l'uso degli attributi personalizzati:

```
<?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>
```

Richiedi 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);
```

Aggiunta di intestazioni personalizzate alle tue richieste [ad es. Per l'autenticazione di base]

Se è necessario aggiungere intestazioni personalizzate alle richieste di volley, non è possibile farlo dopo l'inizializzazione, poiché le intestazioni vengono salvate in una variabile privata.

Invece, è necessario sovrascrivere il metodo `getHeaders()` di `Request.class` come tale:

```
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;
    }
};
```

Spiegazione dei parametri:

- `REQUEST_METHOD` : una delle costanti `Request.Method.*` .
- `REQUEST_URL` - L'URL completo per inviare la tua richiesta a.
- `REQUEST_BODY` - Un oggetto `JSONObject` contenente il corpo POST da inviare (o null).
- `RESP_LISTENER` - Un oggetto `Response.Listener<?>` , Il cui `onResponse(T data)` viene chiamato al completamento con esito positivo.
- `ERR_LISTENER` - Un oggetto `Response.ErrorListener` , il cui `onErrorResponse(VolleyError e)` viene chiamato su una richiesta non riuscita.

Se desideri creare una richiesta personalizzata, puoi aggiungere anche le intestazioni al suo interno:

```
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;
    }
    ...
}
```

Helper Class per la gestione degli errori di volley

```
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);
        }
```

Autenticazione del server remoto tramite StringRequest tramite il metodo POST

Per il gusto di questo esempio, supponiamo di avere un server per gestire le richieste POST che faremo dalla nostra app per 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();
}
```

Utilizzo di Volley per richieste HTTP

Aggiungi la dipendenza gradle in build.gradle a livello di app

```
compile 'com.android.volley:volley:1.0.0'
```

Inoltre, aggiungi l'autorizzazione android.permission.INTERNET al manifest della tua app.

** Crea istanza Singleton Volley RequestQueue nella tua Applicazione **

```
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;
}
}
```

Ora, è possibile utilizzare l'istanza volley utilizzando il metodo getInstance () e aggiungere una nuova richiesta nella coda utilizzando `InitApplication.getInstance().addToQueue(request);`

Un semplice esempio per richiedere JsonObject dal server è

```
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));
```

Per gestire i timeout di Volley è necessario utilizzare `RetryPolicy` . Un criterio di riprova viene utilizzato nel caso in cui una richiesta non possa essere completata a causa di un errore di rete o di altri casi.

Volley fornisce un modo semplice per implementare `RetryPolicy` per le tue richieste. Per impostazione predefinita, Volley imposta tutti i timeout di socket e connessione su 5 secondi per tutte le richieste. `RetryPolicy` è un'interfaccia in cui è necessario implementare la logica di come si desidera riprovare una particolare richiesta quando si verifica un timeout.

Il costruttore prende i seguenti tre parametri:

- `initialTimeoutMs` - Specifica il timeout del socket in millisecondi per ogni tentativo.
- `maxNumRetries` : viene tentato il numero di tentativi.
- `backoffMultiplier` - Un moltiplicatore che viene utilizzato per determinare il tempo esponenziale impostato su socket per ogni tentativo di tentativo.

Risposta variabile booleana dal server con richiesta json in volley

puoi scegliere una classe personalizzata al di sotto di 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 questo con la tua attività

```
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 come corpo della richiesta

Le richieste predefinite integrate in volley non consentono di passare un `JSONArray` come corpo della richiesta in una richiesta `POST` . Invece, puoi solo passare un oggetto `JSON` come parametro.

Tuttavia, invece di passare un oggetto `JSON` come parametro al costruttore della richiesta, è necessario sovrascrivere il metodo `getBody()` di `Request.class` . Dovresti passare `null` come terzo parametro:

```
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;
        }
    }
};
```

Spiegazione dei parametri:

- `REQUEST_URL` - L'URL completo per inviare la tua richiesta a.
- `RESP_LISTENER` - Un oggetto `Response.Listener<?>` , Il cui `onResponse(T data)` viene chiamato al completamento con esito positivo.
- `ERR_LISTENER` - Un oggetto `Response.ErrorListener` , il cui `onErrorResponse(VolleyError e)` viene chiamato su una richiesta non riuscita.

Leggi raffica online: https://riptutorial.com/it/android/topic/2800/raffica

Capitolo 197: RecyclerView

introduzione

RecyclerView è una versione più avanzata di List View con prestazioni migliorate e funzionalità aggiuntive.

Parametri

Parametro	Dettaglio
Adattatore	Una sottoclasse di RecyclerView.Adapter responsabile della fornitura di viste che rappresentano elementi in un set di dati
Posizione	La posizione di un elemento di dati all'interno di un adattatore
Indice	L'indice di una vista figlio allegata come utilizzata in una chiamata a getChildAt (int). Contrasto con posizione
Rilegatura	Il processo di preparazione di una vista secondaria per visualizzare i dati corrispondenti a una posizione all'interno dell'adattatore
Riciclare (visualizzare)	Una vista precedentemente utilizzata per visualizzare i dati per una specifica posizione dell'adattatore può essere collocata in una cache per riutilizzarla in un secondo momento per visualizzare nuovamente lo stesso tipo di dati in un secondo momento. Questo può migliorare drasticamente le prestazioni saltando l'inflazione iniziale del layout o la costruzione
Scrap (view)	Una vista secondaria che è entrata in uno stato temporaneamente staccato durante il layout. Le viste di scarto possono essere riutilizzate senza essere completamente distaccate dal dispositivo principale RecyclerView, non modificate se non è necessario eseguire nuovamente la rifilatura o modificare l'adattatore se la vista è stata considerata sporca
Sporco (vista)	Una vista secondaria che deve essere rimbalzata dall'adattatore prima di essere visualizzata

Osservazioni

`RecyclerView` è una visualizzazione flessibile per fornire una finestra limitata in un set di dati di grandi dimensioni.

Prima di utilizzare `RecyclerView` è necessario aggiungere la dipendenza della libreria di supporto nel file `build.gradle` :

```
dependencies {
    // Match the version of your support library dependency
    compile 'com.android.support:recyclerview-v7:25.3.1'
}
```

È possibile trovare l'ultimo numero di versione di recyclerview dal sito ufficiale.

Altri argomenti correlati:

Esistono altri argomenti che descrivono i componenti di `RecyclerView` :

- RecyclerView LayoutManagers
- RecyclerView ItemDecorations
- RecyclerView onClickListeners

Documentazione ufficiale

http://developer.android.com/reference/android/support/v7/widget/RecyclerView.html

Versioni precedenti:

```
  //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

Aggiunta di una vista Recycler

Aggiungi la dipendenza come descritto nella sezione Note, quindi aggiungi una `RecyclerView` al tuo layout:

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

Dopo aver aggiunto un widget `RecyclerView` al layout, ottenere un handle per l'oggetto, collegarlo a un gestore di layout e collegare un adattatore per i dati da visualizzare:

```
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 semplicemente imposta il layout manager da xml aggiungendo queste righe:

```
xmlns:app="http://schemas.android.com/apk/res-auto"
app:layoutManager="android.support.v7.widget.LinearLayoutManager"
```

Se si è certi che le modifiche apportate al contenuto di `RecyclerView` non modificheranno le dimensioni del layout di `RecyclerView` , utilizzare il codice seguente per migliorare le prestazioni del componente. Se RecyclerView ha una dimensione fissa, sa che RecyclerView non verrà ridimensionato a causa dei suoi figli, quindi non chiama affatto il layout della richiesta. Gestisce solo il cambiamento stesso. Se invalidante qualunque sia il genitore, il coordinatore, il layout o qualsiasi altra cosa. (puoi utilizzare questo metodo anche prima di impostare `LayoutManager` e `Adapter`):

```
mRecyclerView.setHasFixedSize(true);
```

`RecyclerView` fornisce questi gestori di layout integrati da utilizzare. Quindi puoi creare un elenco, una griglia e una griglia sfalsata utilizzando `RecyclerView` :

1. LinearLayoutManager mostra gli oggetti in una lista a scorrimento verticale o orizzontale.
2. GridLayoutManager mostra gli oggetti in una griglia.
3. StaggeredGridLayoutManager mostra gli oggetti in una griglia sfalsata.

Caricamento più agevole degli articoli

Se gli elementi nel tuo `RecyclerView` caricano i dati dalla rete (comunemente immagini) o eseguono altre elaborazioni, ciò può richiedere molto tempo e potresti finire con gli elementi sullo schermo ma non completamente caricati. Per evitare ciò, è possibile estendere il `LinearLayoutManager` esistente per precaricare un numero di elementi prima che diventino visibili sullo schermo:

```
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;
  }
}
```

Trascina e rilascia e scorri con RecyclerView

È possibile implementare le funzionalità di scorrimento a rilascio e trascinamento con RecyclerView senza utilizzare librerie di terze parti.
Basta usare la classe `ItemTouchHelper` inclusa nella libreria di supporto di RecyclerView.

`ItemTouchHelper` di `ItemTouchHelper` con il callback `SimpleCallback` e, in base alla funzionalità supportata, è necessario eseguire l'override di `onMove(RecyclerView, ViewHolder, ViewHolder)` e / o `onSwiped(ViewHolder, int)` e infine collegarlo a `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 ricordare che il costruttore `SimpleCallback` applica la stessa strategia di scorrimento a tutti gli elementi di `RecyclerView` . In ogni caso è possibile aggiornare la direzione di scorrimento predefinita per elementi specifici semplicemente sovrascrivendo il metodo `getSwipeDirs(RecyclerView, ViewHolder)` .

Supponiamo ad esempio che il nostro `RecyclerView` includa un `HeaderViewHolder` e che ovviamente non vogliamo applicare lo swiping ad esso. Sarà sufficiente ignorare `getSwipeDirs` come segue:

```
@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);
}
```

Aggiungi intestazione / piè di pagina a RecyclerView

Questo è un codice adattatore di esempio.

```
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
    }
}
}
```

Ecco una buona lettura sull'implementazione di `RecyclerView` con intestazione e piè di pagina.

Metodo alternativo:

Mentre la risposta sopra funzionerà, puoi utilizzare questo approccio anche usando una vista riciclatore usando un `NestedScrollView` . Puoi aggiungere un layout per l'intestazione usando il seguente approccio:

```
<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>
```

Oppure puoi anche usare un `LinearLayout` con allineamento verticale nel tuo `NestedScrollView` .

Nota: funziona solo con `RecyclerView` sopra **23.2.0**

```
compile 'com.android.support:recyclerview-v7:23.2.0'
```

Utilizzo di più ViewHolders con ItemViewType

A volte un RecyclerView dovrà utilizzare diversi tipi di viste per essere visualizzati nell'elenco mostrato nell'interfaccia utente, e ogni vista richiede un diverso layout xml da gonfiare.

Per questo problema, è possibile utilizzare diverse ViewHolders in single Adapter, utilizzando un metodo speciale in RecyclerView - `getItemViewType(int position)` .

Di seguito è riportato l'esempio dell'utilizzo di due ViewHolders:

1. Un ViewHolder per la visualizzazione delle voci dell'elenco
2. Un ViewHolder per la visualizzazione di più visualizzazioni di intestazione

```
@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.
    }
}
```

Filtra gli elementi all'interno di RecyclerView con SearchView

aggiungi il metodo di `filter` in `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` è inizializzato nel costruttore dell'adattatore come `itemsCopy.addAll(items)` .

In tal caso, chiama il `filter` da `OnQueryTextListener` da `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;
    }
});
```

Menu a comparsa con recyclerView

inserisci questo codice nel ViewHolder

NOTA: In questo codice che sto usando `btnExpand` click-evento, per tutto il `recyclerview` evento click è possibile impostare ascoltatore a itemView oggetto.

```
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);
```

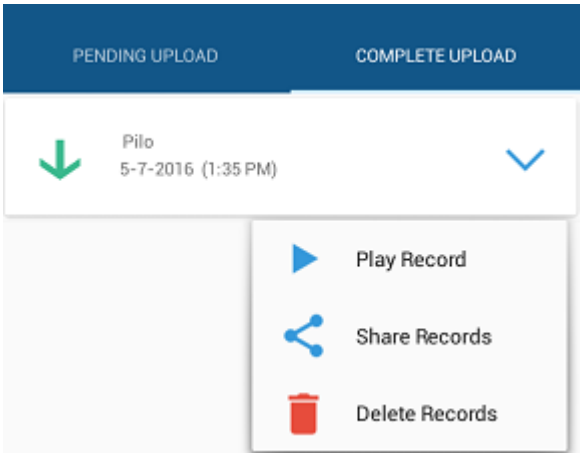
```
            time = (TextView)itemView.findViewById(R.id.time);
            btnIn = (Button)itemView.findViewById(R.id.btn_in_out);

            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();
                }
            });

        }
    }
```

modo alternativo per mostrare le icone nel menu

```
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) {

}
```

Ecco l'output:



Animare la modifica dei dati

`RecyclerView` eseguirà un'animazione pertinente se viene utilizzato uno qualsiasi dei metodi "notify" ad eccezione di `notifyDataSetChanged` ; questo include `notifyItemChanged` , `notifyItemInserted` , `notifyItemMoved` , `notifyItemRemoved` , etc.

L'adattatore dovrebbe estendere questa classe invece di `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);
            }
        }
    }
}
```

> **NON si** deve usare lo stesso `List` per `setModels` e `List` nell'adattatore.

Dichiarate i `models` come variabili globali. `DataModel` è solo una classe fittizia.

```
private List<DataModel> models;
private YourAdapter adapter;
```

Inizializza i `models` prima di passarli alla scheda. `YourAdapter` è l'implementazione di `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));
```

Chiamalo dopo aver aggiornato i tuoi `models` globali.

```
adapter.setModels(new ArrayList(models));
```

> Se non si esegue l'override degli `equals` , tutto il confronto viene confrontato per riferimento.

Esempio utilizzando SortedList

Android ha introdotto la classe `SortedList` subito dopo l'introduzione di `RecyclerView` . Questa classe gestisce tutte le chiamate del metodo 'notify' a `RecyclerView.Adapter` per garantire un'animazione corretta e consente anche il batching di più modifiche, quindi le animazioni non vanno in jitter.

```
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

Ecco una classe ViewHolder generica che è possibile utilizzare con qualsiasi layout di DataBinding. Qui viene creata un'istanza di una particolare classe `ViewDataBinding` utilizzando l'oggetto `View` inflazionato e la classe di utilità `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;
    }
}
```

Dopo aver creato questo corso puoi usare `<layout>` nel tuo file di layout per abilitare il databinding per quel layout come questo:

`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>
```

ed ecco il tuo esempio di datiModello:

```
public class ItemModel {
    public String itemLabel;
}
```

Per impostazione predefinita, la libreria Android Data Binding genera una classe `ViewDataBinding` basata sul nome del file di layout, convertendolo in caso Pascal e aggiungendo il suffisso "Binding" ad esso. Per questo esempio sarebbe `MyItemBinding` per il file di layout `my_item.xml` . Quella classe Binding avrebbe anche un metodo setter per impostare l'oggetto definito come dati nel file di layout (`ItemModel` per questo esempio).

Ora che abbiamo tutti i pezzi possiamo implementare il nostro adattatore in questo modo:

```
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();
    }
}
```

Scorrimento senza fine in Recycleview.

Qui ho condiviso uno snippet di codice per implementare lo scrolling infinito nella vista del riciclo.

Passo 1: Prima di tutto, crea un metodo astratto nell'adattatore Recycleview come di seguito.

```
public abstract class ViewAllCategoryAdapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    public abstract void load();
}
```

Passo 2: Ora sovrascrivi il metodo BindViewHolder e `getItemCount()` della classe ViewAllCategoryAdapter e chiama il metodo `Load()` come di seguito.

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, final int position) {
    if ((position >= getItemCount() - 1)) {
        load();
    }
}

@Override
public int getItemCount() {
    return YOURLIST.size();
}
```

Passo 3: Ora ogni logica di backend è completa ora è il momento di eseguire questa logica. È semplice sovrascrivere il metodo di caricamento in cui si crea l'oggetto dell'adattatore. Questo metodo viene chiamato automaticamente mentre l'utente raggiunge la fine dell'elenco.

```
  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);
```

Ora il metodo `load()` chiama automaticamente mentre l'utente scorre alla fine della lista.

Buona fortuna

Mostra la vista predefinita fino al caricamento degli elementi o quando i dati non sono disponibili

Immagine dello schermo

Classe adattatore

```
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>
```

Sto usando FontAwesome con Iconics Library per le immagini. Aggiungi questo al tuo file build.gradle a livello di app.

```
compile 'com.mikepenz:fontawesome-typeface:4.6.0.3@aar'
compile 'com.mikepenz:iconics-core:2.8.1@aar'
```

Aggiungi linee di divisione a oggetti RecyclerView

Basta aggiungere queste righe all'inizializzazione

```
RecyclerView mRecyclerView = (RecyclerView) view.findViewById(recyclerView);
mRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
mRecyclerView.addItemDecoration(new DividerItemDecoration(getActivity(),
DividerItemDecoration.VERTICAL));
```

Aggiungi un `adapter` e chiama `.notifyDataSetChanged();` come di solito !
Questa non è una funzionalità incorporata di Recyclerview ma aggiunta nelle librerie di supporto. Quindi non dimenticare di includere questo nel tuo file build.gradle a livello di app

```
compile "com.android.support:appcompat-v7:25.3.1"
compile "com.android.support:recyclerview-v7:25.3.1"
```

> Più ItemDecorations possono essere aggiunti a un singolo RecyclerView.

Modifica del colore del divisore :

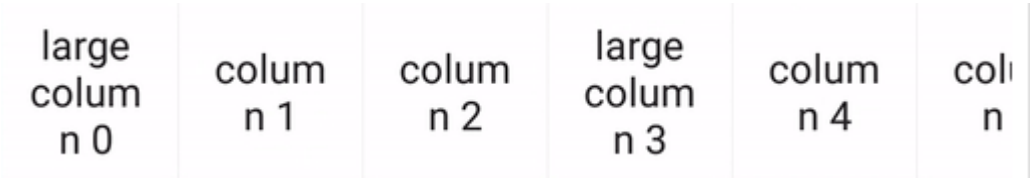
È abbastanza facile impostare un colore per un oggetto Decorazione.

1. Il passo è: creare un file `divider.xml` che si trova sulla cartella `drawable`

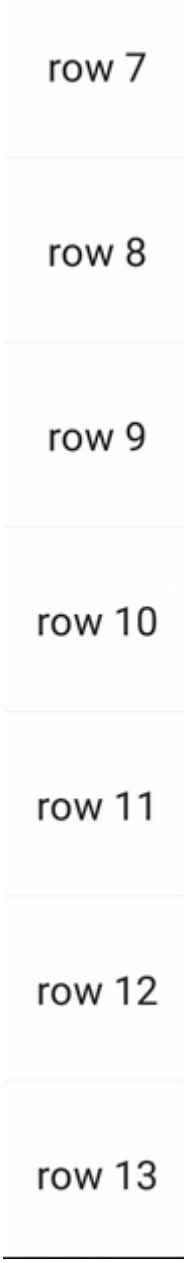
```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="line">
    <size
        android:width="1px"
        android:height="1px"/>
    <solid android:color="@color/divider_color"/>
</shape>
```

2. il passo è: impostazione drawable

```
    // 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);
```



Leggi RecyclerView online: https://riptutorial.com/it/android/topic/169/recyclerview

Capitolo 198: RecyclerView e LayoutManagers

Examples

GridLayoutManager con conteggio dinamico dello span

Quando si crea una panoramica di riciclo con un gestore di layout di griglia, è necessario specificare il conteggio di span nel costruttore. Il conteggio dello span si riferisce al numero di colonne. Questo è abbastanza goffo e non tiene conto delle dimensioni dello schermo o dell'orientamento dello schermo. Un approccio consiste nel creare più layout per le varie dimensioni dello schermo. Un altro approccio più dinamico può essere visto sotto.

Per prima cosa creiamo una classe RecyclerView personalizzata come segue:

```
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);
        }
    }
}
```

Questa classe determina il numero di colonne che possono essere inserite nel recyclerview. Per usarlo dovrai metterlo nel tuo layout.xml come segue:

```
<?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"
    />
```

Si noti che utilizziamo l'attributo columnWidth. Il recyclerview ne avrà bisogno per determinare quante colonne si inseriranno nello spazio disponibile.

Nella tua attività / frammento ottieni solo un riferimento al ricylerview e ne imposti un adattatore (e tutte le decorazioni o animazioni degli elementi che desideri aggiungere). **NON IMPOSTARE UN MANAGER LAYOUT**

```
RecyclerView recyclerView = (RecyclerView) findViewById(R.id.auto_fit_recycler_view);
recyclerView.setAdapter(new MyAdapter());
```

(dove MyAdapter è la tua classe dell'adattatore)

Ora hai una panoramica sul riciclaggio che regolerà lo spancount (es. Colonne) per adattarsi alle dimensioni dello schermo. Come aggiunta finale, potresti voler centrare le colonne nel recyclerview (di default sono allineate a layout_start). Puoi farlo modificando leggermente la classe AutofitRecyclerView. Inizia creando una classe interiore nel recyclerview. Questa sarà una classe che si estende da GridLayoutManager. Aggiungerà abbastanza padding a sinistra e a destra per centrare le righe:

```
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();
        }
    }
}
```

Quindi, quando si imposta il LayoutManager in AutofitRecyclerView, utilizzare CenteredGridLayoutManager come segue:

```
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);
}
```

E questo è tutto! Hai uno spancount dinamico, riciclabile basato sul gridlayoutmanager allineato al centro.

fonti:

- Blog di Chiu-Ki Chan's Square Island
- StackOverflow

Aggiunta della vista intestazione a recyclerview con gridlayout manager

Per aggiungere un'intestazione a un recyclerview con un gridlayout, è innanzitutto necessario comunicare all'adattatore che la vista dell'intestazione è la prima posizione, piuttosto che la cella standard utilizzata per il contenuto. Successivamente, il gestore di layout deve essere informato che la prima posizione deve avere una span uguale al conteggio * span dell'intera lista. *

Prendi una normale classe RecyclerView.Adapter e configurala come segue:

```
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
    }
}
```

Quindi nell'attività / frammento:

```
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);
```

Lo stesso approccio può essere utilizzato aggiungendo un footer in aggiunta o al posto di un'intestazione.

Fonte: blog Chiu-Ki Chan's Square Island

Elenco semplice con LinearLayoutManager

Questo esempio aggiunge un elenco di luoghi con l'immagine e il nome utilizzando un `ArrayList` di costume `Place` oggetti come set di dati.

Disposizione delle attività

Il layout dell'attività / frammento o in cui viene utilizzato solo RecyclerView deve contenere RecyclerView. Non c'è ScrollView o un layout specifico necessario.

```
<?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>
```

Definire il modello di dati

È possibile utilizzare qualsiasi tipo di classe o di dati primitivi come un modello, come `int` , `String` , `float[]` o `CustomObject` . RecyclerView farà riferimento a un `List` di questi oggetti / primitive.

Quando una voce di elenco si riferisce a diversi tipi di dati come testo, numeri, immagini (come in questo esempio con i luoghi), è spesso una buona idea utilizzare un oggetto personalizzato.

```
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;
    }
}
```

Elenca il layout dell'articolo

Devi specificare un file di layout xml che verrà utilizzato per ogni elemento dell'elenco. In questo esempio, viene utilizzato un `ImageView` per l'immagine e un `TextView` per il nome. Il `LinearLayout` posiziona l' `ImageView` a sinistra e il `TextView` direttamente all'immagine.

```
<?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>
```

Creare un adattatore RecyclerView e ViewHolder

Successivamente, è necessario ereditare `RecyclerView.Adapter` e `RecyclerView.ViewHolder` . Una solita struttura di classe sarebbe:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    public class ViewHolder extends RecyclerView.ViewHolder {
        // ...
    }
}
```

Innanzitutto, implementiamo `ViewHolder` . Assorbe solo il costruttore predefinito e salva le viste necessarie in alcuni campi:

```
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);
    }
}
```

Il costruttore dell'adattatore imposta il set di dati utilizzato:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    private List<Place> mPlaces;

    public PlaceListAdapter(List<Place> contacts) {
        mPlaces = contacts;
    }

    // ...
}
```

Per usare il nostro layout di elenco personalizzato, sostituiamo il metodo `onCreateViewHolder(...)` . In questo esempio, il file di layout si chiama `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);
    }

    // ...
}
```

In `onBindViewHolder(...)` , effettivamente impostiamo i contenuti delle viste. Otteniamo il modello usato trovandolo nella `List` nella posizione data e quindi impostando l'immagine e il nome nelle viste `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());
    }

    // ...
}
```

Dobbiamo inoltre implementare `getItemCount()` , che restituisce semplicemente la dimensione della `List` .

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public int getItemCount() {
        return mPlaces.size();
    }

    // ...
}
```

(Genera dati casuali)

Per questo esempio, genereremo alcuni posti casuali.

```
@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;
}
```

Collega RecyclerView con PlaceListAdapter e il set di dati

Connettere un `RecyclerView` con un adattatore è molto semplice. È necessario impostare `LinearLayoutManager` come gestore di layout per ottenere il layout dell'elenco.

```
@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));
}
```

Fatto!

StaggeredGridLayoutManager

1. Crea la tua RecyclerView nel tuo file xml di layout:

```
<android.support.v7.widget.RecyclerView
            android:id="@+id/recycleView"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
```

2. Crea la tua classe Model per contenere i tuoi dati:

```
     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. Creare un file di layout per conservare gli elementi di 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. Creare la classe dell'adattatore per 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. Crea un'istanza di RecyclerView nella tua attività o frammento:

```
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() );
```

Non dimenticare di aggiungere la dipendenza Picasso nel tuo file build.gradle:

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

Leggi RecyclerView e LayoutManagers online:
https://riptutorial.com/it/android/topic/6772/recyclerview-e-layoutmanagers

Capitolo 199: RecyclerView onClickListeners

Examples

Nuovo esempio

```
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);
    }
}
```

Esempio di Kotlin e RxJava

Primo esempio reimplementato in Kotlin e utilizzo di RxJava per un'interazione più pulita.

```
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
        }
    }
}
```

L'utilizzo è piuttosto semplice allora. È possibile iscriversi su thread separati utilizzando le strutture RxJava.

```
val adapter = SampleAdapter(arrayOf("Hello", "World"))
adapter.itemClickStream.subscribe { v ->
    if (v.id == R.id.textView) {
        // do something
    }
}
```

Facile esempio OnLongClick e OnClick

Innanzitutto, implementa il tuo titolare della vista:

```
implements View.OnClickListener, View.OnLongClickListener
```

Quindi, registra gli ascoltatori come segue:

```
itemView.setOnClickListener(this);
itemView.setOnLongClickListener(this);
```

Quindi, ignorare gli ascoltatori come segue:

```
@Override
public void onClick(View v) {
    onclicklistner.onItemClick(getAdapterPosition(), v);
}

@Override
public boolean onLongClick(View v) {
    onclicklistner.onItemLongClick(getAdapterPosition(), v);
    return true;
}
```

E infine, aggiungi il seguente codice:

```
public void setOnItemClickListener(onClickListner onclicklistner) {
    SampleAdapter.onclicklistner = onclicklistner;
}

public void setHeader(View v) {
    this.headerView = v;
}
```

```
public interface onClickListner {
    void onItemClick(int position, View v);
    void onItemLongClick(int position, View v);
}
```

Demo dell'adattatore

```
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));
        }
    }
}
```

Il codice di esempio sopra può essere chiamato con il seguente codice:

```
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();
    }
});
```

Elemento di selezione degli ascoltatori

Per implementare un listener di click di elementi e / o un listener di oggetti lunghi, è possibile creare un'interfaccia nell'adattatore:

```
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;
    }

}
```

Quindi puoi impostare i listener dei clic dopo aver creato un'istanza dell'adattatore:

```
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;
    }
});
```

Un altro modo per implementare Listener di clic articoli

Un altro modo per implementare l'elemento click listener consiste nell'utilizzare l'interfaccia con diversi metodi, il cui numero è uguale al numero di visualizzazioni selezionabili, e utilizzare gli ascoltatori di clic sovrascritti, come si può vedere di seguito. Questo metodo è più flessibile, perché puoi impostare i listener dei clic su visualizzazioni diverse e abbastanza facilmente controllare la logica dei clic separatamente per ciascuno.

```
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);
        }
    }
}
```

E quando hai un'istanza di adattatore, puoi impostare l'ascoltatore dei clic che ascolta facendo clic su ciascuna delle viste:

```
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);
    }
}
```

In 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) {


   }
  }));
```

Leggi RecyclerView onClickListeners online: https://riptutorial.com/it/android/topic/96/recyclerview-onclicklisteners

Capitolo 200: Registrazione e utilizzo di Logcat

Sintassi

- `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)`

Parametri

Opzione	Descrizione
-b (buffer)	Carica un buffer di registro alternativo per la visualizzazione, ad esempio eventi o radio. Il buffer principale è usato di default. Vedere Visualizzazione dei buffer di registro alternativi.
-c	Cancella (svuota) l'intero registro ed esce.
-d	Scarica il log sullo schermo ed esce.
-f (nome file)	Scrive l'output del messaggio di log su (nome file). L'impostazione predefinita è stdout.
-g	Stampa la dimensione del buffer di registro specificato ed esce.
-n (conta)	Imposta il numero massimo di registri ruotati su (conteggio). Il valore predefinito è 4. Richiede l'opzione -r.
-r (kbytes)	Ruota il file di registro ogni (kbyte) di output. Il valore predefinito è 16. Richiede l'opzione -f.
-S	Imposta le specifiche del filtro predefinito su Silenzioso.
-v (formato)	Imposta il formato di output per i messaggi di registro. L'impostazione predefinita è il breve formato.

Osservazioni

Definizione

Logcat è uno strumento da riga di comando che esegue il dump di un registro dei messaggi di sistema, incluse le tracce dello stack quando il dispositivo genera un errore e i messaggi che hai scritto dalla tua app con la classe Log .

Quando usare

Se stai pensando di utilizzare i metodi System.out di Java per la stampa sulla console invece di utilizzare uno dei metodi Log di Android, devi sapere che funzionano sostanzialmente allo stesso modo. Tuttavia, è meglio evitare di usare i metodi di Java perché le informazioni extra e la formattazione fornite dai metodi di log di Android sono più vantaggiose. Inoltre, i metodi di stampa System.out vengono reindirizzati al metodo `Log.i()` .

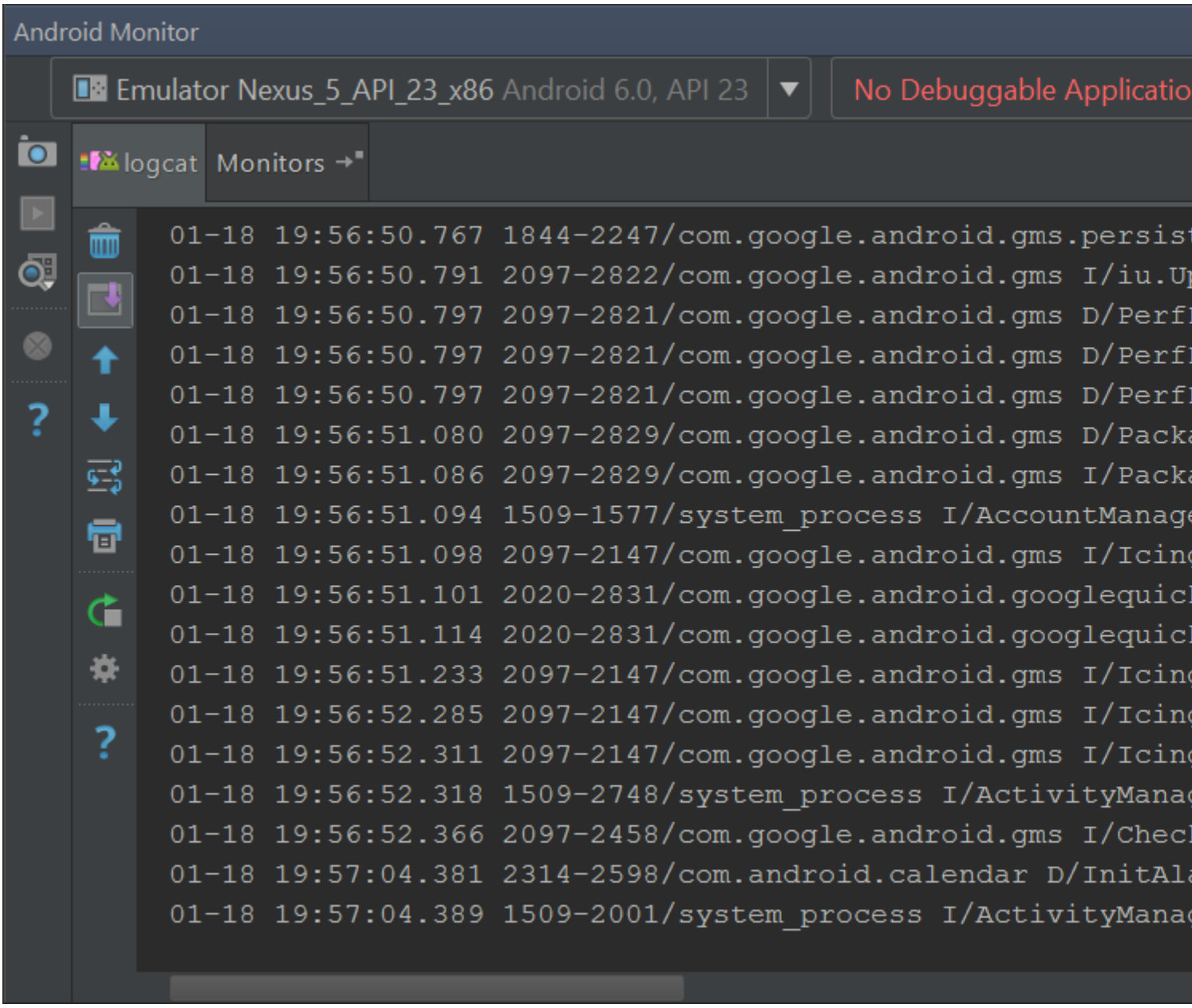
Link utili

- Documentazione ufficiale per sviluppatori Android per Log e logcat .
- Stackoveflow Domanda: Android Log.v (), Log.d (), Log.i (), Log.w (), Log.e () - Quando utilizzarli?

Examples

Filtro dell'output del logcat

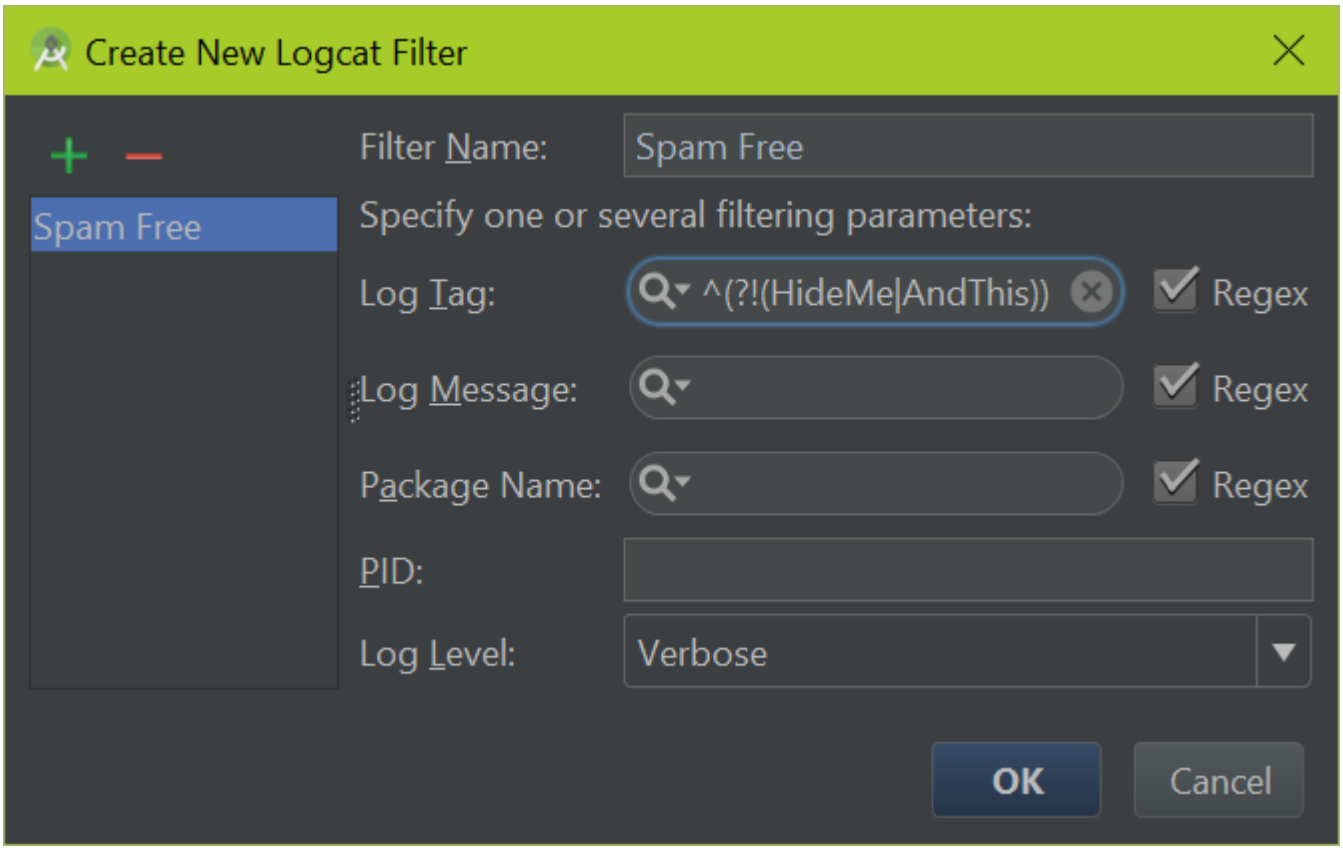
È utile filtrare l'output logcat perché ci sono molti messaggi che non interessano. Per filtrare l'output, apri il "Monitor Android" e fai clic sul menu a discesa in alto a destra e seleziona *Modifica configurazione filtro*



Ora puoi aggiungere filtri personalizzati per mostrare i messaggi che ti interessano, oltre a filtrare linee di log ben note che possono essere tranquillamente ignorate. Per ignorare una parte dell'output puoi definire un'espressione regolare . Ecco un esempio di esclusione dei tag corrispondenti:

```
^(?!(HideMe|AndThis))
```

Questo può essere inserito seguendo questo esempio:



Quanto sopra è un'espressione regolare che esclude gli input. Se si desidera aggiungere un altro tag alla *lista nera* , aggiungerlo dopo una pipe `|` personaggio. Ad esempio, se si desidera inserire nella lista nera "GC", si utilizzerà un filtro come questo:

```
^(?!(HideMe|AndThis|GC))
```

Per ulteriori documentazione ed esempi visita la registrazione e l'utilizzo di Logcat

Registrazione

Qualsiasi applicazione Android di qualità terrà traccia di ciò che sta facendo attraverso i registri delle applicazioni. Questi registri consentono una facile guida per il debug per lo sviluppatore per diagnosticare cosa sta succedendo con l'applicazione. La documentazione completa di Android può essere trovata qui , ma segue un riepilogo:

Registrazione di base

La classe `Log` è la fonte principale di scrittura dei log dello sviluppatore, specificando un `tag` e un `message` . Il tag è ciò che puoi usare per filtrare i messaggi di log per identificare quali linee provengono dalla tua particolare attività. Basta chiamare

```
Log.v(String tag, String msg);
```

E il sistema Android scriverà un messaggio 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
```

MANCIA:
Si noti l'id del processo e l'id del thread. Se sono uguali, il log proviene dal thread principale / dell'interfaccia utente!

È possibile utilizzare qualsiasi tag, ma è comune utilizzare il nome della classe come tag:

```
public static final String tag = MyAnimator.class.getSimpleName();
```

Registra i livelli

Il logger Android ha 6 diversi livelli, ognuno dei quali ha un determinato scopo:

- `ERROR` : `Log.e()`
  - Usato per indicare un errore critico, questo è il livello stampato quando si lancia `Exception` .
- `WARN` : `Log.w()`
  - Utilizzato per indicare un avvertimento, principalmente per guasti recuperabili
- `INFO` : `Log.i()`
  - Utilizzato per indicare informazioni di livello superiore sullo stato dell'applicazione
- `DEBUG` : `Log.d()`
  - Utilizzato per registrare le informazioni che sarebbero utili sapere quando si esegue il debug dell'applicazione, ma si intromettono durante l'esecuzione dell'applicazione
- `VERBOSE` : `Log.v()`
  - Utilizzato per registrare le informazioni che riflettono i piccoli dettagli sullo stato dell'applicazione
- `ASSERT` : `Log.wtf()`
  - Utilizzato per registrare informazioni su una condizione che non dovrebbe mai accadere.
  - *wtf* sta per "What a Terrible Failure".

Motivazione per la registrazione

La motivazione per la registrazione è quella di trovare facilmente errori, avvisi e altre informazioni dando un'occhiata alla catena di eventi dall'applicazione. Ad esempio, immagina un'applicazione che legge le righe da un file di testo, ma presuppone erroneamente che il file non sarà mai vuoto. La traccia di log (di un'app che non registra) sarebbe simile a questa:

```
E/MyApplication: Process: com.example.myapplication, PID: 25788
                         com.example.SomeRandomException: Expected string, got 'null' instead
```

Seguito da una serie di tracce di stack che alla fine porteranno alla linea illecita, in cui il passaggio

da un debugger alla fine porterebbe al problema

Tuttavia, la traccia di registro di un'applicazione con la registrazione abilitata potrebbe essere simile a questa:

```
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
```

Una rapida occhiata ai log ed è ovvio che il file era vuoto.

Cose da considerare durante la registrazione:

Sebbene la registrazione sia un potente strumento che consente agli sviluppatori di Android di ottenere una visione più approfondita del funzionamento interno della loro applicazione, la registrazione presenta alcuni inconvenienti.

Log Readability:

È normale che le applicazioni Android contengano più registri in modo sincrono. In quanto tale, è molto importante che ogni registro sia facilmente leggibile e contenga solo informazioni pertinenti e necessarie.

Prestazione:

La registrazione richiede una piccola quantità di risorse di sistema. In generale, ciò non giustifica la preoccupazione, tuttavia, in caso di uso eccessivo, la registrazione potrebbe avere un impatto negativo sulle prestazioni dell'applicazione.

Sicurezza:

Recentemente, diverse applicazioni Android sono state aggiunte al marketplace di Google Play che consentono all'utente di visualizzare i registri di tutte le applicazioni in esecuzione. Questa visualizzazione non voluta dei dati può consentire agli utenti di visualizzare informazioni riservate. Come regola generale, rimuovi sempre i registri che contengono dati non pubblici *prima di* pubblicare la tua applicazione sul mercato.

Conclusione:

La registrazione è una parte essenziale di un'applicazione Android, a causa del potere che offre

agli sviluppatori. La possibilità di creare un'utile traccia di log è uno degli aspetti più impegnativi dello sviluppo del software, ma la classe Log di Android aiuta a semplificare notevolmente.

Per ulteriori documentazione ed esempi visita la registrazione e l'utilizzo di Logcat

Accedi con il link al sorgente direttamente da Logcat

Questo è un bel trucco per aggiungere un collegamento al codice, quindi sarà facile passare al codice che ha emesso il log.

Con il seguente codice, questa chiamata:

```
MyLogger.logWithLink("MyTag","param="+param);
```

Risulterà in:

```
07-26...012/com.myapp D/MyTag: MyFrag:onStart(param=3)  (MyFrag.java:2366) // << logcat
converts this to a link to source!
```

Questo è il codice (all'interno di una classe chiamata 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());
}
```

Questo è un esempio di base, può essere facilmente esteso per emettere un collegamento al chiamante (suggerimento: lo stack sarà [4] invece di [3]), e puoi anche aggiungere altre informazioni rilevanti.

Utilizzando il logcat

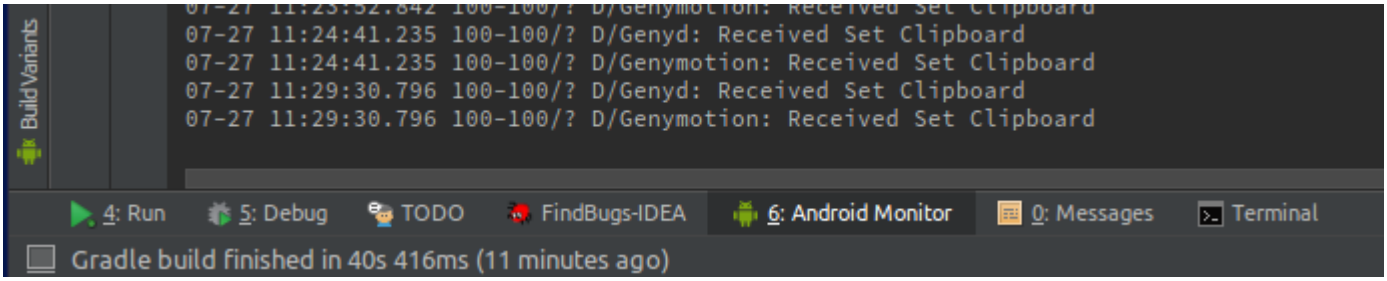
Logcat è uno strumento da riga di comando che esegue il dump di un registro dei messaggi di sistema, incluse le tracce dello stack quando il dispositivo genera un errore e i messaggi che hai scritto dalla tua app con la classe Log.

L'output di Logcat può essere visualizzato all'interno del monitor Android di Android Studio o con

la riga di comando adb.

In Android Studio

Mostra facendo clic sull'icona "Monitor Android":



Oppure premendo `Alt` + `6` su Windows / Linux o `CMD` + `6` su Mac.

tramite la riga di comando:

Uso semplice:

```
$ adb logcat
```

Con i timestamp:

```
$ adb logcat -v time
```

Filtra su un testo specifico:

```
$ adb logcat -v time | grep 'searchtext'
```

Ci sono molte opzioni e filtri disponibili per la *riga* di *comando logcat* , documentata qui .
Un esempio semplice ma utile è la seguente espressione di filtro che visualizza tutti i messaggi di registro con livello di priorità "errore", su tutti i tag:

```
$ adb logcat *:E
```

Generazione del codice di registrazione

`Live templates Android Studio` possono offrire alcune scorciatoie per la registrazione rapida.
Per utilizzare i modelli Live, è sufficiente iniziare a digitare il nome del modello e premere `TAB` o immettere per inserire l'istruzione.

Esempi:

- `logi` → diventa → `android.util.Log.i(TAG, "$METHOD_NAME$: $content$");`
  - `$METHOD_NAME$` verrà automaticamente sostituito dal nome del tuo metodo e il cursore attenderà che il contenuto venga riempito.
- `loge` → stesso, per errore
- ecc. per il resto dei livelli di registrazione.

Elenco completo dei modelli può essere trovato in `Android Studio` impostazioni `s'(Alt` + `s` e digitare "live"). Ed è possibile aggiungere anche i modelli personalizzati.

Se ritieni che i `Live templates Android Studio` non siano sufficienti per le tue esigenze, puoi prendere in considerazione il plug-in Postfix di Android

Questa è una libreria molto utile che ti aiuta a evitare di scrivere manualmente la linea di registrazione.

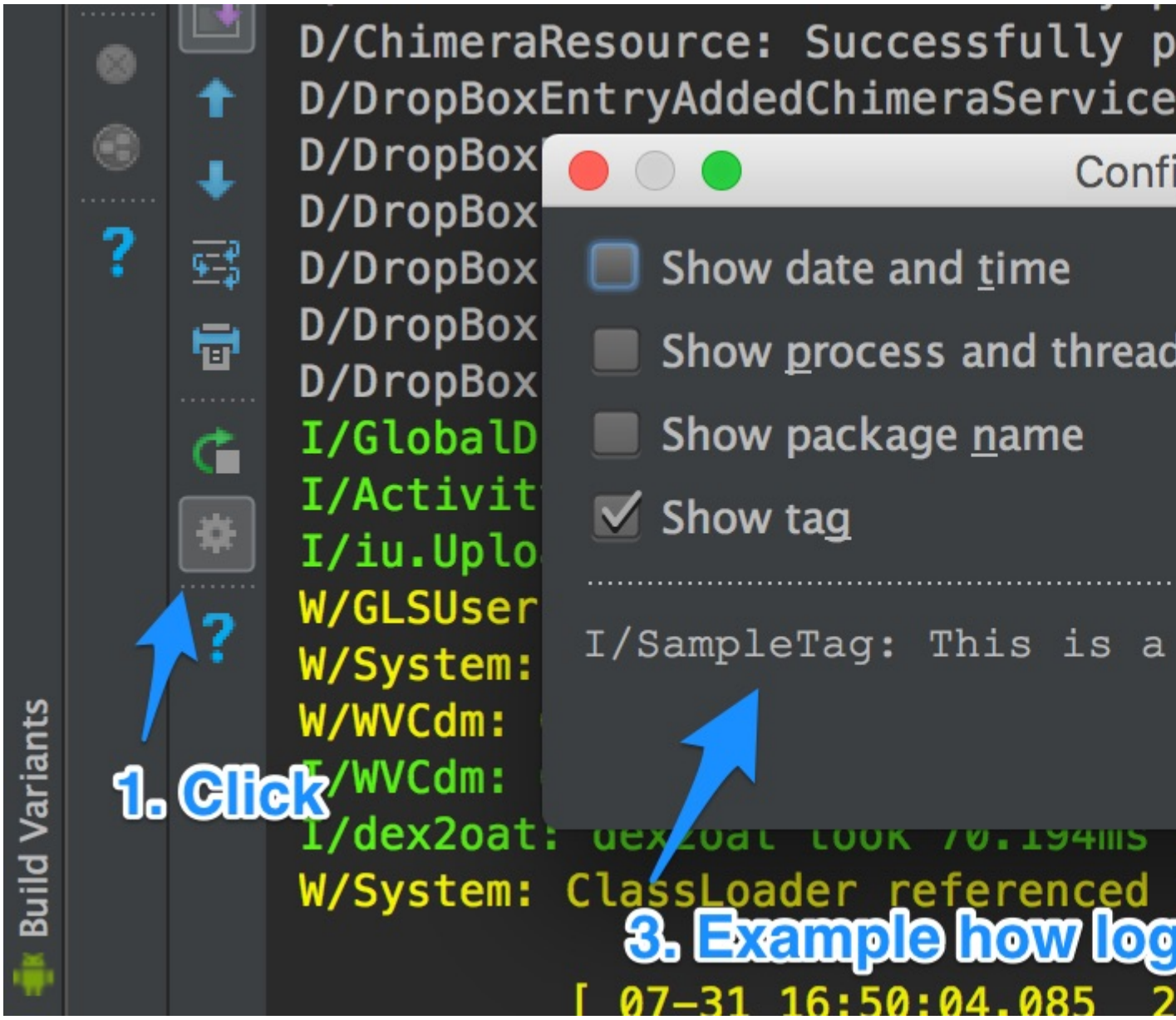
La sintassi è assolutamente semplice:

.log - Registrazione. Se c'è una variabile costante "TAG", usa "TAG". Altrimenti usa il nome della classe.

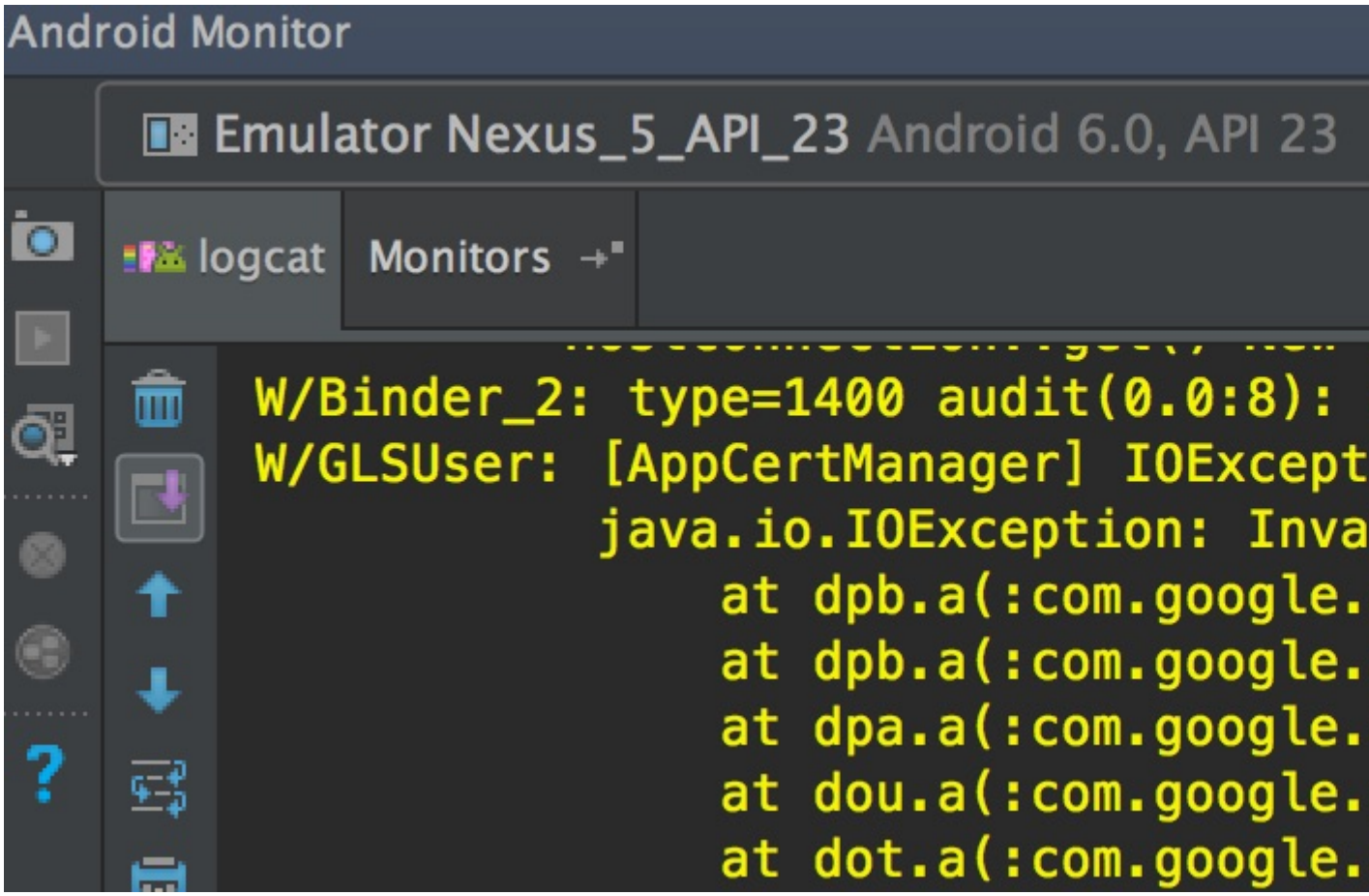
```
blic class MyActivity extends Activity {
  static final String T= "test";
  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      new View.OnClickListener() {
          @Override
          public void onClick(View view) {

          }
      };
  }
```

Utilizzo di Android Studio

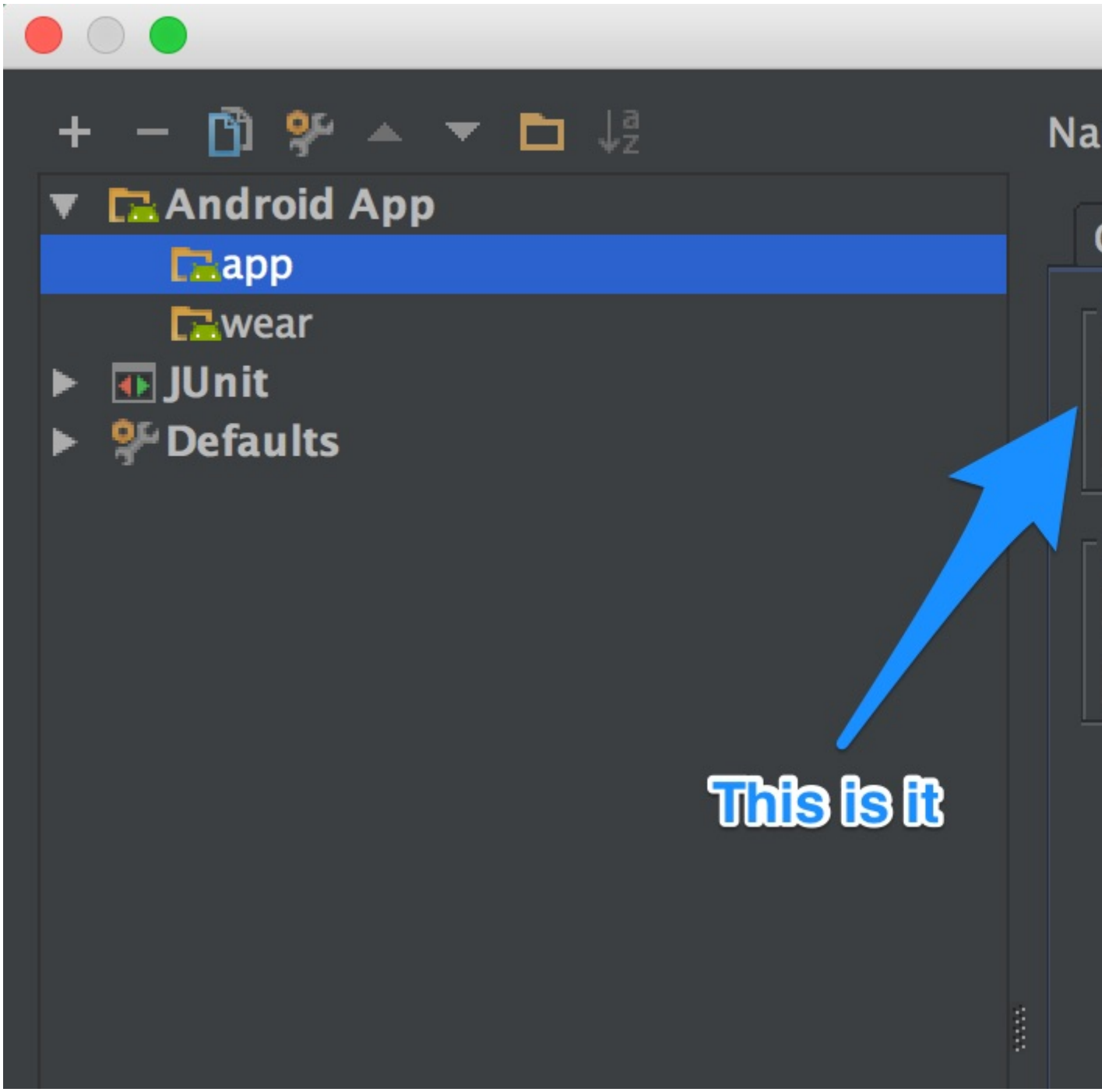
1. Nascondi / mostra le informazioni stampate:



2. Controlla la verbosità della registrazione:



3. Disabilita / abilita la finestra del registro di apertura all'avvio dell'applicazione run / debug



Cancella registri

Per cancellare (svuotare) l'intero registro:

```
adb logcat -c
```

Leggi Registrazione e utilizzo di Logcat online:
https://riptutorial.com/it/android/topic/1552/registrazione-e-utilizzo-di-logcat

Capitolo 201: Regno

introduzione

Realm Mobile Database è un'alternativa a SQLite. Il Realm Mobile Database è molto più veloce di un ORM e spesso più veloce di SQLite non elaborato.

Benefici

Funzionalità offline, query veloci, threading sicuro, app multipiattaforma, crittografia, architettura reattiva.

Osservazioni

Quando si utilizza Realm, è necessario ricordare che non è necessario passare istanze RealmObjects, RealmResults e Realm tra i thread. Se hai bisogno di una query su un determinato thread, apri un'istanza di Realm su quel thread. Alla fine del thread, dovresti chiudere il Realm.

> ***NOTA LEGALE*** : l'utente comprende che il Software potrebbe contenere funzioni crittografiche che potrebbero essere soggette a restrizioni sull'esportazione e dichiara e garantisce **di non trovarsi in un Paese** soggetto a restrizioni o embargo sulle esportazioni degli Stati Uniti, **tra cui Cuba, Iran, Nord Corea, Sudan, Siria o regione della Crimea** e che non si trovi nell'elenco del Dipartimento del commercio di persone negate, parti non verificate o affiliate a un'entità soggetta a restrizioni.

Examples

Aggiunta di regno al tuo progetto

Aggiungi la seguente dipendenza al file `build.gradle` livello di **progetto** .

```
dependencies {
    classpath "io.realm:realm-gradle-plugin:3.1.2"
}
```

Aggiungi il seguente diritto nella parte superiore del tuo file `build.gradle` livello di **app** .

```
apply plugin: 'realm-android'
```

Completa una sincronizzazione gradle e ora hai Realm aggiunto come dipendenza al tuo progetto!

Il reame richiede una chiamata iniziale dalla 2.0.0 prima di usarlo. Puoi farlo nella classe `Application` o nel metodo `onCreate` della tua prima attività.

```
Realm.init(this); // added in Realm 2.0.0
```

```
Realm.setDefaultConfiguration(new RealmConfiguration.Builder().build());
```

Modelli di regno

I modelli di realm devono estendere la classe base `RealmObject` , definiscono lo schema del database sottostante.

I tipi di campo supportati sono `boolean` , `byte` , `short` , `int` , `long` , `float` , `double` , `String` , `Date` , `byte[]` , collegamenti ad altri `RealmObject` `RealmList<T extends RealmModel>` e `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
}
```

Se aggiungi (o rimuovi) un nuovo campo a RealmObject (o aggiungi una nuova classe RealmObject o ne elimini uno esistente), sarà necessaria una **migrazione** . È possibile impostare `deleteIfMigrationNeeded()` in `RealmConfiguration.Builder` o definire la migrazione necessaria. La migrazione è necessaria anche quando si aggiunge (o si rimuove) `@Required` , o `@Index` o `@PrimaryKey` annotazione `@PrimaryKey` .

Le relazioni devono essere impostate manualmente, NON sono automatiche in base alle chiavi primarie.

Dalla versione 0.88.0, è anche possibile utilizzare campi pubblici anziché campi privati / getter / setter nelle classi RealmObject.

È anche possibile implementare `RealmModel` invece di estendere `RealmObject` , se la classe è anche annotata con `@RealmClass` .

```
@RealmClass
public class Person implements RealmModel {
    // ...
}
```

In tal caso, metodi come `person.deleteFromRealm()` o `person.addChangeListener()` vengono sostituiti con `RealmObject.deleteFromRealm(person)` e `RealmObject.addChangeListener(person)` .

Le limitazioni sono che da un `RealmObject` , solo `RealmObject` può essere esteso, e non c'è supporto per i campi `final` , `volatile` e `transient` .

È importante che una classe di RealmObject *gestita* possa essere modificata solo in una transazione. Un RealmObject *gestito* non può essere passato tra i thread.

Elenco di primitive (RealmList)

Attualmente Realm non supporta la memorizzazione di un elenco di primitive. È nella loro lista delle cose da fare (GitHub issue # 575), ma per il momento, ecco una soluzione alternativa.

Crea una nuova classe per il tuo tipo primitivo, questo usa Integer, ma cambialo per qualunque cosa tu voglia immagazzinare.

```
public class RealmInteger extends RealmObject {
    private int val;

    public RealmInteger() {
    }

    public RealmInteger(int val) {
        this.val = val;
    }

    // Getters and setters
}
```

Ora puoi usare questo nel tuo `RealmObject` .

```
public class MainObject extends RealmObject {

    private String name;
    private RealmList<RealmInteger> ints;

    // Getters and setters
}
```

Se stai utilizzando `GSON` per popolare `RealmObject` , dovrai aggiungere un adattatore di tipo personalizzato.

```
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();
```

risorse try-with-

```
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
}
```

Le risorse Prova con possono essere utilizzate solo da KITKAT (minSDK 19)

Query ordinate

Per ordinare una query, invece di usare `findAll()` , dovresti usare `findAllSorted()` .

```
RealmResults<SomeObject> results = realm.where(SomeObject.class)
                                        .findAllSorted("sortField", Sort.ASCENDING);
```

Nota:

`sort()` restituisce un nuovo RealmResults che è ordinato, ma un aggiornamento a questo RealmResults lo ripristinerà. Se usi `sort()` , dovresti sempre riordinarlo in `RealmChangeListener` , rimuovere `RealmChangeListener` dai precedenti `RealmResults` e aggiungerlo ai nuovi `RealmResults` restituiti. L'utilizzo di `sort()` su un `RealmResults` restituito da una query asincrona non ancora caricata avrà esito negativo.

`findAllSorted()` restituirà sempre i risultati ordinati per il campo, anche se viene aggiornato. Si consiglia di utilizzare `findAllSorted()` .

Query asincrone

Ogni metodo di ricerca sincrona (come `findAll()` o `findAllSorted()`) ha una controparte asincrona (`findAllAsync()` / `findAllSortedAsync()`).

Le query asincrone offload la valutazione di `RealmResults` su un altro thread. Per ricevere questi risultati sul thread corrente, il thread corrente deve essere un thread looper (leggi: le query asincrone di solito funzionano solo sul thread dell'interfaccia utente).

```
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);
```

Usare Realm con RxJava

Per le query, Realm fornisce il metodo `realmResults.asObservable()` . L'osservazione dei risultati è possibile solo sui thread looper (in genere il thread dell'interfaccia utente).

Perché ciò funzioni, la tua configurazione deve contenere quanto segue

```
realmConfiguration = new RealmConfiguration.Builder(context)       //
                         .rxFactory(new RealmObservableFactory()) //
                            //...
                         .build();
```

Successivamente, è possibile utilizzare i risultati come osservabili.

```
Observable<RealmResults<SomeObject>> observable = results.asObservable();
```

Per le query asincrone, è necessario filtrare i risultati per `isLoaded()` , in modo da ricevere un evento solo quando la query è stata eseguita. Questo `filter()` non è necessario per le query sincrone (`isLoaded()` restituisce sempre `true` nelle query di sincronizzazione).

```
    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));
```

Per le scritture, è necessario utilizzare il metodo `executeTransactionAsync()` o aprire un'istanza di Realm sul thread in background, eseguire la transazione in modo sincrono, quindi chiudere l'istanza di 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 di base

Impostazione di un'istanza

Per usare Realm devi prima ottenere un'istanza di esso. Ogni istanza di Realm viene mappata su un file su disco. Il modo più semplice per ottenere un'istanza è il seguente:

```
// 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();
```

Il metodo `Realm.getInstance()` crea il file di database se non è stato creato, altrimenti apre il file. L'oggetto `RealmConfiguration` controlla tutti gli aspetti di come viene creato un Reame, sia che si tratti di un database `inMemory()` , nome del file Realm, se il Reame deve essere cancellato se è necessaria una migrazione, i dati iniziali, ecc.

Si noti che le chiamate a `Realm.getInstance()` sono conteggiate con riferimento (ogni chiamata incrementa un contatore) e il contatore viene decrementato quando viene chiamato `realm.close()` .

Chiusura di un'istanza

Nei thread in background, è *molto importante* ***chiudere*** le istanze di Realm quando non vengono più utilizzate (ad esempio, la transazione è completa e l'esecuzione del thread termina). La mancata chiusura di tutte le istanze di Realm sul thread in background determina il blocco della versione e può causare una notevole crescita delle dimensioni del file.

```
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 notare che sopra il livello API 19, puoi sostituire questo codice con questo:

```
try(Realm realm = Realm.getDefaultInstance()) {
    // ...
}
```

Modelli

Il prossimo passo sarebbe creare i tuoi modelli. Qui potrebbe essere posta una domanda, "che cos'è un modello?". Un modello è una struttura che definisce le proprietà di un oggetto che viene memorizzato nel database. Ad esempio, nel seguito modelliamo 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;
    }
}
```

Nota che i tuoi modelli dovrebbero estendere la classe RealmObject. La chiave primaria viene inoltre specificata `@PrimaryKey` . Le chiavi primarie possono essere nulle, ma solo un elemento può avere `null` come chiave primaria. Inoltre puoi usare l'annotazione `@Ignore` per i campi che non dovrebbero essere persistenti sul disco:

```
@Ignore
private String isbn;
```

Inserimento o aggiornamento dei dati

Per poter archiviare un oggetto libro nell'istanza del database di Realm, è possibile prima creare un'istanza del modello e quindi memorizzarla nel database tramite il metodo `copyToRealm` . Per la creazione o l'aggiornamento è possibile utilizzare `copyToRealmOrUpdate` . (Un'alternativa più veloce è l' `insertOrUpdate()` appena aggiunto.

```
// 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);
    }
});
```

Si noti che tutte le modifiche ai dati devono avvenire in una transazione. Un altro modo per creare un oggetto è utilizzare il seguente modello:

```
Book book = realm.createObject(Book.class, primaryKey);
...
```

Interrogare il database

- Tutti i libri:

  ```
  RealmResults<Book> results = realm.where(Book.class).findAll();
  ```

- Tutti i libri con ID superiore a 10:

  ```
  RealmResults<Book> results = realm.where(Book.class)
                                    .greaterThan("id", 10)
                                    .findAll();
  ```

- Libri di `'Taylor Swift'` o `'%Peter%'` :

```
RealmResults<Book> results = realm.where(Book.class)
                                  .beginGroup()
                                      .equalTo("author", "Taylor Swift")
                                      .or()
                                      .contains("author", "Peter")
                                  .endGroup().findAll();
```

Cancellare un oggetto

Ad esempio, vogliamo eliminare tutti i libri di 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();
    }
});
```

Leggi Regno online: https://riptutorial.com/it/android/topic/3187/regno

Capitolo 202: RenderScript

introduzione

RenderScript è un linguaggio di scripting che consente di scrivere rendering grafico ad alte prestazioni e codice di calcolo raw. Fornisce un mezzo per scrivere codice critico delle prestazioni che il sistema compila successivamente in codice nativo per il processore su cui può essere eseguito. Questa potrebbe essere la CPU, una CPU multi-core o anche la GPU. In definitiva, ciò dipende da molti fattori che non sono prontamente disponibili per lo sviluppatore, ma dipende anche da quale architettura supporta il compilatore di piattaforma interno.

Examples

Iniziare

RenderScript è un framework per consentire il calcolo parallelo ad alte prestazioni su Android. Gli script scritti verranno eseguiti su tutti i processori disponibili (ad es. CPU, GPU, ecc.) In parallelo, consentendo all'utente di concentrarsi sull'attività che si desidera ottenere anziché su come è pianificata ed eseguita.

Gli script sono scritti in un linguaggio basato su C99 (C99 è una vecchia versione dello standard del linguaggio di programmazione C). Per ogni script viene creata una classe Java che consente di interagire facilmente con RenderScript nel codice Java.

Impostazione del tuo progetto

Esistono due modi diversi per accedere a RenderScript nella tua app, con le librerie di Android Framework o la libreria di supporto. Anche se non desideri utilizzare i dispositivi di destinazione prima dell'API 11, devi sempre utilizzare l'implementazione della Libreria di supporto perché garantisce la compatibilità dei dispositivi su molti dispositivi diversi. Per utilizzare l'implementazione della libreria di supporto è necessario utilizzare almeno gli strumenti di compilazione versione `18.1.0` !

Ora consente di configurare il file build.gradle dell'applicazione:

```
android {
    compileSdkVersion 24
    buildToolsVersion '24.0.1'

    defaultConfig {
        minSdkVersion 8
        targetSdkVersion 24

        renderscriptTargetApi 18
        renderscriptSupportModeEnabled true
    }
```

```
}
```

- `renderscriptTargetApi` : questo dovrebbe essere impostato sulla versione più recente del livello API che fornisce tutte le funzionalità RenderScript richieste.
- `renderscriptSupportModeEnabled` : abilita l'uso dell'implementazione RenderScript della libreria di supporto.

Come funziona RenderScript

Un tipico RenderScript consiste di due cose: i kernel e le funzioni. Una funzione è proprio quello che sembra: accetta un input, fa qualcosa con quell'input e restituisce un output. Un kernel è da dove proviene il vero potere di RenderScript.

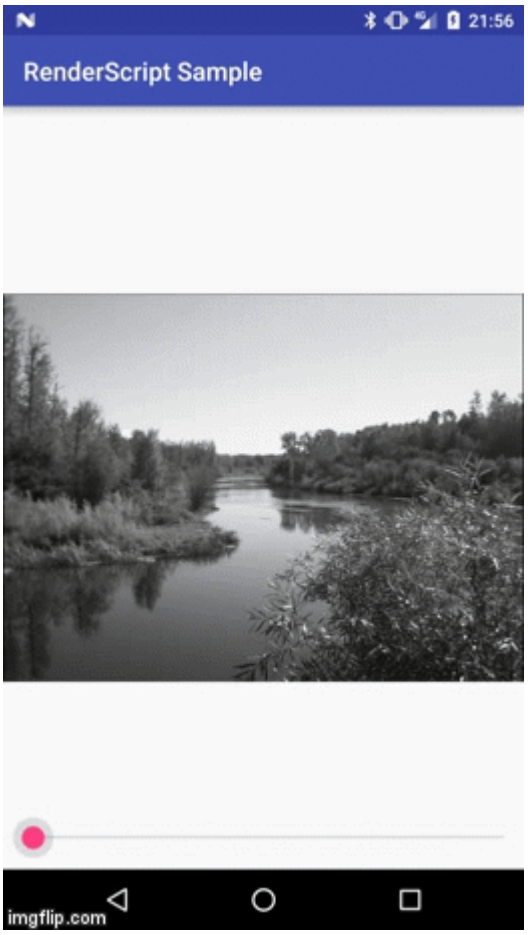
Un kernel è una funzione che viene eseguita su ogni elemento all'interno di `Allocation` . `Allocation` può essere utilizzata per passare dati come una `Bitmap` o una matrice di `byte` a un `RenderScript` e vengono anche utilizzati per ottenere un risultato da un kernel. I kernel possono prendere una `Allocation` come input e un'altra come output oppure possono modificare i dati all'interno di una sola `Allocation` .

Puoi scrivere il tuo kernel, ma ci sono anche molti kernel predefiniti che puoi usare per eseguire operazioni comuni come un Gaussian Image Blur.

Come già accennato per ogni file RenderScript, viene generata una classe per interagire con esso. Queste classi iniziano sempre con il prefisso `ScriptC_` seguito dal nome del file RenderScript. Ad esempio, se il tuo file RenderScript è chiamato `example` la classe Java generata verrà chiamata `ScriptC_example` . Tutti gli script predefiniti iniziano solo con lo `Script` prefisso, ad esempio lo script di sfocatura immagine gaussiana si chiama `ScriptIntrinsicBlur` .

Scrivi il tuo primo RenderScript

Il seguente esempio è basato su un esempio su GitHub. Esegue la manipolazione di base delle immagini modificando la saturazione di un'immagine. Puoi trovare il codice sorgente **qui** e verificarlo se vuoi giocarci da solo. Ecco una rapida gif di come dovrebbe apparire il risultato:



RenderScript Boilerplate

I file RenderScript risiedono nella cartella `src/main/rs` nel tuo progetto. Ogni file ha l'estensione del file `.rs` e deve contenere due istruzioni `#pragma` nella parte superiore:

```
#pragma version(1)
#pragma rs java_package_name(your.package.name)
```

- `#pragma version(1)` : può essere usato per impostare la versione di RenderScript che stai utilizzando. Attualmente c'è solo la versione 1.
- `#pragma rs java_package_name(your.package.name)` : può essere usato per impostare il nome del pacchetto della classe Java generata per interagire con questo particolare RenderScript.

C'è un altro `#pragma` che dovresti normalmente impostare in ciascuno dei tuoi file RenderScript e che è usato per impostare la precisione in virgola mobile. È possibile impostare la precisione in virgola mobile su tre diversi livelli:

- `#pragma rs_fp_full` : questa è l'impostazione più rigorosa con la massima precisione ed è anche il valore predefinito se non si specifica nulla. Dovresti usarlo se hai bisogno di una precisione in virgola mobile elevata.
- `#pragma rs_fp_relaxed` : questo garantisce un'elevata precisione in virgola mobile, ma su alcune architetture consente una serie di ottimizzazioni che possono far sì che i tuoi script possano girare più velocemente.

- `#pragma rs_fp_imprecise` : questo garantisce una precisione ancora minore e dovrebbe essere usato se la precisione in virgola mobile non è veramente importante per il tuo script.

La maggior parte degli script può usare `#pragma rs_fp_relaxed` meno che non sia veramente necessaria un'elevata precisione in virgola mobile.

Variabili globali

Ora, proprio come nel codice C, puoi definire variabili o costanti globali:

```
const static float3 gMonoMult = {0.299f, 0.587f, 0.114f};

float saturationLevel = 0.0f;
```

La variabile `gMonoMult` è di tipo `float3` . Ciò significa che è un vettore costituito da 3 numeri float. L'altra variabile `float` denominata `saturationValue` non è costante, pertanto è possibile impostarla in fase di esecuzione su un valore che ti piace. Puoi usare variabili come questa nei tuoi Kernel o funzioni e quindi sono un altro modo per dare input o ricevere output dai tuoi RenderScripts. Per ogni variabile non costante verrà generato un metodo getter e setter sulla classe Java associata.

Noccioli

Ma ora iniziamo ad implementare il Kernel. Per gli scopi di questo esempio non ho intenzione di spiegare la matematica usata nel kernel per modificare la saturazione dell'immagine, ma invece si concentrerà su come implementare un kernel e su come usarlo. Alla fine di questo capitolo spiegherò rapidamente cosa sta facendo il codice in questo kernel.

Kernels in generale

Diamo prima un'occhiata al codice sorgente:

```
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);
}
```

Come puoi vedere sembra una normale funzione C con un'eccezione: `__attribute__((kernel))` tra il tipo di ritorno e il nome del metodo. Questo è ciò che dice a RenderScript che questo metodo è un kernel. Un'altra cosa che potresti notare è che questo metodo accetta un parametro `uchar4` e restituisce un altro valore `uchar4` . `uchar4` è - come la variabile `float3` discussa nel capitolo precedente - un vettore. Contiene 4 valori `uchar` che sono solo valori byte nell'intervallo compreso tra 0 e 255.

È possibile accedere a questi singoli valori in molti modi diversi, ad esempio `in.r` restituirebbe il byte che corrisponde al canale rosso di un pixel. Usiamo un `uchar4` poiché ogni pixel è composto da 4 valori - `r` per rosso, `g` per verde, `b` per blu e `a` per alpha - e puoi accedervi con questa

stenografia. RenderScript consente anche di prendere qualsiasi numero di valori da un vettore e creare un altro vettore con essi. Ad esempio `in.rgb` restituirebbe un valore `uchar3` che contiene solo le parti rossa, verde e blu del pixel senza il valore alfa.

Durante il runtime RenderScript chiamerà questo metodo Kernel per ogni pixel di un'immagine, motivo per cui il valore restituito e il parametro sono solo un valore `uchar4` . RenderScript eseguirà molte di queste chiamate in parallelo su tutti i processori disponibili, motivo per cui RenderScript è così potente. Ciò significa anche che non devi preoccuparti del thread o della sicurezza dei thread, puoi semplicemente implementare ciò che vuoi fare su ciascun pixel e RenderScript si prenderà cura di tutto il resto.

Quando si chiama un Kernel in Java vengono fornite due variabili di `Allocation` , una che contiene i dati di input e un'altra che riceverà l'output. Vostro metodo Kernel verrà chiamato per ogni valore nell'input `Allocation` e scriverà il risultato all'uscita `Allocation` .

Metodi dell'API Runtime di RenderScript

Nel kernel sopra sono usati alcuni metodi forniti fuori dalla scatola. RenderScript fornisce molti di questi metodi e sono fondamentali per qualsiasi cosa tu debba fare con RenderScript. Tra questi ci sono i metodi per fare operazioni matematiche come `sin()` e metodi helper come `mix()` che mescola due valori in base ad altri valori. Ma ci sono anche metodi per operazioni più complesse quando si tratta di vettori, quaternioni e matrici.

Il **riferimento dell'API RenderScript Runtime** ufficiale è la migliore risorsa là fuori se vuoi saperne di più su un particolare metodo o stai cercando un metodo specifico che esegua un'operazione comune come il calcolo del prodotto punto di una matrice. Puoi trovare questa documentazione **qui** .

Implementazione del kernel

Ora diamo un'occhiata alle specifiche di ciò che questo kernel sta facendo. Ecco la prima riga del kernel:

```
float4 f4 = rsUnpackColor8888(in);
```

Chiama la prima linea di costruzione metodo **`rsUnpackColor8888()`** che trasforma `uchar4` valore di un `float4` valori. Ogni canale di colore viene anche trasformato nell'intervallo `0.0f - 1.0f` dove `0.0f` corrisponde a un valore di byte di `0` e `1.0f` a `255` . Lo scopo principale di questo è rendere molto più semplici tutti i calcoli in questo Kernel.

```
float3 dotVector = dot(f4.rgb, gMonoMult);
```

Questa riga successiva utilizza il metodo incorporato **`dot()`** per calcolare il prodotto punto di due vettori. `gMonoMult` è un valore costante abbiamo definito alcuni capitoli sopra. Dal momento che entrambi i vettori devono essere della stessa lunghezza per calcolare il prodotto punto e anche dal momento che vogliamo solo influenzare i canali di colore e non il canale alfa di un pixel usiamo la scorciatoia `.rgb` per ottenere un nuovo vettore `float3` che contiene solo il canali di colore rosso,

verde e blu. Quelli di noi che ancora ricordano a scuola come funziona il prodotto dot noteranno subito che il prodotto puntino dovrebbe restituire solo un valore e non un vettore. Tuttavia nel codice sopra stiamo assegnando il risultato a un vettore `float3` . Questa è ancora una caratteristica di RenderScript. Quando assegni un numero unidimensionale a un vettore, tutti gli elementi nel vettore saranno impostati su questo valore. Ad esempio il seguente snippet assegnerà `2.0f` a ciascuno dei tre valori nel vettore `float3` :

```
float3 example = 2.0f;
```

Quindi il risultato del prodotto punto sopra è assegnato a ciascun elemento nel vettore `float3` sopra.

Ora arriva la parte in cui effettivamente usiamo la variabile globale `saturationLevel` per modificare la saturazione dell'immagine:

```
float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
```

Questo usa il metodo integrato **`mix()`** per mescolare insieme il colore originale con il vettore di punti prodotto che abbiamo creato sopra. Il modo in cui vengono mescolati insieme è determinato dalla variabile global `saturationLevel` . Quindi un livello di `saturationLevel` pari a `0.0f` farà sì che il colore risultante non abbia parte dei valori di colore originali e consisterà solo di valori nel `dotVector` che si traduce in un'immagine in bianco e nero o in grigio. Un valore di `1.0f` farà sì che il colore risultante sia completamente composto da valori di colore originali e valori superiori a `1.0f` moltiplicheranno i colori originali per renderli più luminosi e intensi.

```
return rsPackColorTo8888(newColor);
```

Questa è l'ultima parte del kernel. **`rsPackColorTo8888()`** trasforma il vettore `float3` in un valore `uchar4` che viene quindi restituito. I valori di byte risultanti vengono bloccati su un intervallo compreso tra 0 e 255, pertanto valori float superiori a `1.0f` generano un valore di byte pari a 255 e valori inferiori a `0.0` generano un valore di byte pari a `0` .

E questa è l'intera implementazione del kernel. Ora c'è solo una parte rimanente: come chiamare un kernel in Java.

Richiamo di RenderScript in Java

Nozioni di base

Come già detto sopra per ogni file RenderScript, viene generata una classe Java che consente di interagire con gli script. Questi file hanno il prefisso `ScriptC_` seguito dal nome del file RenderScript. Per creare un'istanza di queste classi è necessario innanzitutto un'istanza della classe `RenderScript` :

```
final RenderScript renderScript = RenderScript.create(context);
```

Il metodo statico `create()` può essere utilizzato per creare un'istanza `RenderScript` da un `Context` . È quindi possibile creare un'istanza della classe Java che è stata generata per il tuo script. Se hai chiamato il file RenderScript `saturation.rs` la classe si chiamerà `ScriptC_saturation` :

```
final ScriptC_saturation script = new ScriptC_saturation(renderScript);
```

Su questa classe ora puoi impostare il livello di saturazione e chiamare il kernel. Il setter che è stato generato per la variabile `saturationLevel` avrà il prefisso `set_` seguito dal nome della variabile:

```
script.set_saturationLevel(1.0f);
```

C'è anche un getter con prefisso `get_` che ti permette di ottenere il livello di saturazione attualmente impostato:

```
float saturationLevel = script.get_saturationLevel();
```

I kernel definiti in RenderScript sono preceduti da `forEach_` seguito dal nome del metodo Kernel. Il kernel che abbiamo scritto si aspetta `Allocation` input e `Allocation` output come parametri:

```
script.forEach_saturation(inputAllocation, outputAllocation);
```

L' `Allocation` input deve contenere l'immagine di input e, al termine del metodo `forEach_saturation` , l'allocazione di output conterrà i dati di immagine modificati.

Una volta `Allocation` un'istanza di `Allocation` è possibile copiare i dati da e verso tali `Allocations` utilizzando i metodi `copyFrom()` e `copyTo()` . Ad esempio è possibile copiare una nuova immagine nel proprio input `Assegnazione chiamando:

```
inputAllocation.copyFrom(inputBitmap);
```

Allo stesso modo è possibile recuperare l'immagine risultato chiamando `copyTo()` sull'uscita `Allocation` :

```
outputAllocation.copyTo(outputBitmap);
```

Creare istanze di allocazione

Esistono molti modi per creare `Allocation` . Una volta che hai un'istanza di `Allocation` puoi copiare nuovi dati da e verso quelle `Allocations` con `copyTo()` e `copyFrom()` come spiegato sopra, ma per crearli inizialmente devi sapere con che tipo di dati stai lavorando esattamente. Iniziamo con l' `Allocation` degli input:

Possiamo usare il metodo statico `createFromBitmap()` per creare rapidamente l' `Allocation` input da una `Bitmap` :

```
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, image);
```

In questo esempio l'immagine di input non cambia mai, quindi non è più necessario modificare di nuovo l' `Allocation` input. Possiamo riutilizzarlo ogni volta che `saturationLevel` cambia per creare un nuovo `Bitmap` output.

Creare l'output L' `Allocation` è un po 'più complessa. Per prima cosa dobbiamo creare ciò che è chiamato un `Type` . Un `Type` è usato per dire `Allocation` con che tipo di dati sta trattando. Di solito si usa la classe `Type.Builder` per creare rapidamente un `Type` appropriato. Diamo prima un'occhiata al codice:

```
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
        .setX(inputBitmap.getWidth())
        .setY(inputBitmap.getHeight())
        .create();
```

Stiamo lavorando con un normale `Bitmap` a 32 bit (o in altre parole 4 byte) per pixel con 4 canali di colore. Ecco perché stiamo scegliendo `Element.RGBA_8888` per creare il `Type` . Quindi usiamo i metodi `setX()` e `setY()` per impostare la larghezza e l'altezza dell'immagine di output alla stessa dimensione dell'immagine di input. Il metodo `create()` crea quindi il `Type` con i parametri che abbiamo specificato.

Una volta che abbiamo il `Type` corretto possiamo creare l' `Allocation` dell'output con il metodo statico `createTyped()` :

```
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);
```

Ora abbiamo quasi finito. Abbiamo anche bisogno di un output `Bitmap` in cui possiamo copiare i dati dalla uscita `Allocation` . Per fare questo usiamo il metodo statico `createBitmap()` per creare una nuova `Bitmap` vuota con le stesse dimensioni e la stessa configurazione della `Bitmap` input.

```
final Bitmap outputBitmap = Bitmap.createBitmap(
        inputBitmap.getWidth(),
        inputBitmap.getHeight(),
        inputBitmap.getConfig()
);
```

E con questo abbiamo tutti i pezzi del puzzle per eseguire il nostro RenderScript.

Esempio completo

Ora mettiamo insieme tutto questo in un solo esempio:

```
// 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);
```

Conclusione

Con questa introduzione dovresti essere pronto a scrivere i tuoi kernel RenderScript per una semplice manipolazione delle immagini. Tuttavia ci sono alcune cose da tenere a mente:

- **RenderScript funziona solo nei progetti di applicazione** : attualmente i file RenderScript non possono far parte di un progetto di libreria.
- **Attenzione per la memoria** : RenderScript è molto veloce, ma può anche richiedere un uso intensivo della memoria. Non ci dovrebbe mai essere più di una istanza di `RenderScript` in qualsiasi momento. Dovresti anche riutilizzare il più possibile. Normalmente è sufficiente creare le istanze di `Allocation` una sola volta e riutilizzarle in futuro. Lo stesso vale per le `Bitmaps` output o le istanze di script. Riutilizzare il più possibile.
- **Fai il tuo lavoro in background** : Ancora RenderScript è molto veloce, ma non istantaneo in alcun modo. Qualsiasi kernel, specialmente quelli complessi, dovrebbe essere eseguito fuori dal thread dell'interfaccia utente in un `AsyncTask` o qualcosa di simile. Tuttavia per la maggior parte non devi preoccuparti di perdite di memoria. Tutte le classi correlate a RenderScript utilizzano solo l'applicazione `Context` e quindi non causano perdite di memoria. Ma devi ancora preoccuparti delle solite cose come perdite di `View` , `Activity` o qualsiasi istanza di `Context` che usi tu stesso!
- **Usa elementi incorporati** : esistono numerosi script predefiniti che eseguono attività come sfocatura dell'immagine, fusione, conversione, ridimensionamento. E ci sono molti altri metodi incorporati che ti aiutano a implementare i tuoi kernel. È probabile che se vuoi fare qualcosa, c'è uno script o un metodo che fa già ciò che stai cercando di fare. Non reinventare la ruota.

Se vuoi iniziare subito a giocare con il codice reale ti consiglio di dare un'occhiata al progetto GitHub di esempio che implementa l'esatto esempio di cui si parla in questo tutorial. Puoi trovare il progetto **qui** . Divertiti con RenderScript!

Sfocare un'immagine

Questo esempio dimostra come usare l'API Renderscript per sfocare un'immagine (usando Bitmap). Questo esempio utilizza ScriptInstrinsicBlur fornito da Android Renderscript API (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;
    }
}
```

Ogni script ha un kernel che elabora i dati ed è generalmente invocato tramite il metodo `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
    }
}
```

Questo ha concluso l'esempio qui. Si consiglia di eseguire l'elaborazione in un thread in background.

Sfocare 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);
```

Leggi RenderScript online: https://riptutorial.com/it/android/topic/5214/renderscript

Capitolo 203: Retrofit2

introduzione

La pagina ufficiale di Retrofit si descrive come

Un client REST sicuro per tipo per Android e Java.

Retrofit trasforma la tua API REST in un'interfaccia Java. Utilizza le annotazioni per descrivere le richieste HTTP, la sostituzione dei parametri URL e il supporto dei parametri di query è integrato per impostazione predefinita. Inoltre, fornisce funzionalità per corpo di richiesta multipart e upload di file.

Osservazioni

Dipendenze per la biblioteca di retrofit:

Dalla documentazione ufficiale :

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 semplice richiesta GET

Mostreremo come effettuare una richiesta `GET` a un'API che risponde con un oggetto `JSON` o un array `JSON` . La prima cosa che dobbiamo fare è aggiungere le dipendenze Retrofit e `GSON` Converter al file gradle del nostro modulo.

Aggiungi le dipendenze per la libreria di aggiornamento come descritto nella sezione Note.

Esempio di oggetto JSON previsto:

```
{
    "deviceId": "56V56C14SF5B4SF",
    "name": "Steven",
    "eventCount": 0
}
```

Esempio di array JSON:

```
[
    {
        "deviceId": "56V56C14SF5B4SF",
        "name": "Steven",
        "eventCount": 0
    },
    {
        "deviceId": "35A80SF3QDV7M9F",
        "name": "John",
        "eventCount": 2
    }
]
```

Esempio di classe modello corrispondente:

```
public class Device
{
    @SerializedName("deviceId")
    public String id;

    @SerializedName("name")
    public String name;

    @SerializedName("eventCount")
    public int eventCount;
}
```

Le annotazioni `@SerializedName` qui provengono dalla libreria `GSON` e ci consentono di `serialize` e `deserialize` questa classe in `JSON` utilizzando come chiavi il nome serializzato. Ora possiamo costruire l'interfaccia per l'API che preleverà effettivamente i dati dal server.

```
public interface DeviceAPI
{
    @GET("device/{deviceId}")
    Call<Device> getDevice (@Path("deviceId") String deviceID);

    @GET("devices")
    Call<List<Device>> getDevices();
}
```

C'è molto da fare qui in uno spazio piuttosto compatto quindi cerchiamo di scomporlo:

- L'annotazione `@GET` proviene da Retrofit e indica alla biblioteca che stiamo definendo una richiesta GET.
- Il percorso tra parentesi è l'endpoint che la nostra richiesta GET dovrebbe colpire (imposteremo l'URL di base un po 'più tardi).

- Le parentesi graffe ci consentono di sostituire parti del percorso in fase di esecuzione in modo da poter passare argomenti.
- La funzione che stiamo definendo si chiama `getDevice` e prende l'id del dispositivo che vogliamo come argomento.
- L'annotazione `@PATH` dice a Retrofit che questo argomento dovrebbe sostituire il segnaposto "deviceId" nel percorso.
- La funzione restituisce un oggetto `Call` di tipo `Device` .

Creazione di una classe wrapper:

Ora faremo una piccola classe wrapper per la nostra API per mantenere il codice di inizializzazione di Retrofit avvolto bene.

```
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);
    }
}
```

Questa classe crea un'istanza di GSON per essere in grado di analizzare la risposta JSON, crea un'istanza di Retrofit con il nostro URL di base e un GSONConverter e quindi crea un'istanza della nostra API.

Chiamando l'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());
    }
});
```

Questo utilizza la nostra interfaccia API per creare un oggetto `Call<Device>` e per creare una `Call<List<Device>>` rispettivamente. Chiamare l' `enqueue` dice a Retrofit di effettuare quella chiamata su un thread in background e restituire il risultato alla richiamata che stiamo creando qui.

Nota: l' analisi di un array JSON di oggetti primitivi (come *String, Integer, Boolean* e *Double*) è simile all'analisi di un array JSON. Tuttavia, non hai bisogno della tua classe di modello. Ad esempio, è possibile ottenere la matrice di stringhe con il tipo di restituzione della chiamata come `Call<List<String>>` .

Aggiungi la registrazione a Retrofit2

Le richieste di retrofit possono essere registrate utilizzando un'intercettatrice. Sono disponibili diversi livelli di dettaglio: NESSUNO, BASE, INTESTAZIONI, CORPO. Vedi il progetto Github qui .

1. Aggiungi dipendenza a build.gradle:

```
compile 'com.squareup.okhttp3:logging-interceptor:3.8.1'
```

2. Aggiungi l'intercettore di registrazione durante la creazione di 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();
```

L'esposizione dei registri nel terminale (monitor Android) è qualcosa che dovrebbe essere evitato nella versione di rilascio in quanto potrebbe portare all'esposizione indesiderata di informazioni

critiche come i token di autenticazione ecc.

Per evitare che i registri vengano esposti durante il periodo di esecuzione, verificare le seguenti condizioni

```
if(BuildConfig.DEBUG){
     //your interfector code here
    }
```

Per esempio:

```
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();
```

Caricamento di un file tramite Multipart

Dichiara la tua interfaccia con le annotazioni di Retrofit2:

```
public interface BackendApiClient {
    @Multipart
    @POST("/uploadFile")
    Call<RestApiDefaultResponse> uploadPhoto(@Part("file\"; filename=\"photo.jpg\" ")
RequestBody photo);
}
```

Dove `RestApiDefaultResponse` è una classe personalizzata contenente la risposta.

Costruire l'implementazione della tua API e accodare la chiamata:

```
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>);
```

Retrofit con intercettore OkHttp

Questo esempio mostra come usare un intercettore di richiesta con OkHttp. Questo ha numerosi casi d'uso come:

- Aggiunta di `header` universale alla richiesta. Ad esempio, l'autenticazione di una richiesta
- Debug delle applicazioni in rete
- Recupero della `response` elaborata
- Registrazione della transazione di rete ecc.
- Imposta agente utente personalizzato

```
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);
```

Vedi l'argomento OkHttp per maggiori dettagli.

Intestazione e corpo: un esempio di autenticazione

Le annotazioni `@Header` e `@Body` possono essere inserite nelle firme del metodo e Retrofit le creerà automaticamente in base ai modelli.

```
public interface MyService {
     @POST("authentication/user")
     Call<AuthenticationResponse> authenticateUser(@Body AuthenticationRequest request,
@Header("Authorization") String basicToken);
}
```

AuthenticaionRequest è il nostro modello, un POJO, contenente le informazioni richieste dal server. Per questo esempio, il nostro server vuole la chiave e il segreto del cliente.

```
public class AuthenticationRequest {
     String clientKey;
     String clientSecret;
}
```

Si noti che in `@Header("Authorization")` stiamo specificando che stiamo popolando l'intestazione Autorizzazione. Le altre intestazioni verranno popolate automaticamente poiché Retrofit può dedurre ciò che sono in base al tipo di oggetti che stiamo inviando e che si aspettano in cambio.

Creiamo il nostro servizio di retrofit da qualche parte. Ci assicuriamo di utilizzare HTTPS.

```
Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https:// some example site")
            .client(client)
            .build();
MyService myService = retrofit.create(MyService.class)
```

Quindi possiamo usare il nostro servizio.

```
AuthenticationRequest request = new AuthenticationRequest();
request.setClientKey(getClientKey());
request.setClientSecret(getClientSecret());
String basicToken = "Basic " + token;
myService.authenticateUser(request, basicToken);
```

Carica più file utilizzando Retrofit come multipart

Una volta impostato l'ambiente Retrofit nel tuo progetto, puoi utilizzare il seguente esempio che dimostra come caricare più file usando 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;
}
```

Di seguito è l'interfaccia

```
public interface WebAPIService {
    @Multipart
    @POST("main.php")
    Call<String> postFile(@PartMap Map<String,RequestBody> Files, @Part("json") String
description);
}
```

Scarica un file dal server usando Retrofit2

Dichiarazione dell'interfaccia per il download di un file

```
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);
}
```

L'opzione 1 viene utilizzata per scaricare un file dal server che sta avendo un URL fisso. e l'opzione 2 viene utilizzata per passare un valore dinamico come URL completo per richiedere una chiamata. Questo può essere utile quando si scaricano file, che dipendono da parametri come utente o tempo.

Imposta il retrofit per effettuare chiamate 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);
    }

}
```

Ora, realizza l'implementazione di api per scaricare il file dal server

```
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) {

            }
        });
    }
```

E dopo aver ricevuto la risposta nel callback, codifica alcuni IO standard per il salvataggio del file su disco. Ecco il codice:

```
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;
        }
    }
```

Nota abbiamo specificato **ResponseBody** come tipo restituito, altrimenti Retrofit tenterà di analizzarlo e convertirlo, il che non ha senso quando si scarica il file.

Se vuoi saperne di più sui prodotti Retrofit, accedi a questo link in quanto è molto utile. [1]: https://futurestud.io/blog/retrofit-getting-started-and-android-client

Debugging con Stetho

Aggiungi le seguenti dipendenze alla tua applicazione.

```
compile 'com.facebook.stetho:stetho:1.5.0'
compile 'com.facebook.stetho:stetho-okhttp3:1.5.0'
```

Nel metodo `onCreate` classe Application, chiama quanto segue.

```
Stetho.initializeWithDefaults(this);
```

Quando si crea l'istanza di `Retrofit` , creare un'istanza di OkHttp personalizzata.

```
OkHttpClient.Builder clientBuilder = new OkHttpClient.Builder();
clientBuilder.addNetworkInterceptor(new StethoInterceptor());
```

Quindi imposta questa istanza personalizzata di OkHttp nell'istanza di Retrofit.

```
Retrofit retrofit = new Retrofit.Builder()
    // ...
    .client(clientBuilder.build())
    .build();
```

Ora collega il telefono al computer, avvia l'app e digita `chrome://inspect` nel browser Chrome. Le chiamate di rete di retrofit dovrebbero ora essere visualizzate per essere ispezionate.

Retrofit 2 Custom Xml Converter

Aggiunta di dipendenze nel file build.gradle.

```
dependencies {
    ....
    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile ('com.thoughtworks.xstream:xstream:1.4.7') {
        exclude group: 'xmlpull', module: 'xmlpull'
    }
    ....
}
```

Quindi creare 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);
    }
}
```

creare una classe per gestire la richiesta del corpo.

```
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();
        }
    }
}
```

creare una classe per gestire la risposta del corpo.

```
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());
    }
}
```

Quindi, questo punto possiamo inviare e ricevere qualsiasi XML, Abbiamo solo bisogno di creare annotazioni XStream per le entità.

Quindi creare un'istanza di 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 semplice richiesta POST con GSON

Esempio JSON:

```
{
    "id": "12345",
    "type": "android"
}
```

Definisci la tua richiesta:

```
public class GetDeviceRequest {

    @SerializedName("deviceId")
    private String mDeviceId;

    public GetDeviceRequest(String deviceId) {
        this.mDeviceId = deviceId;
    }

    public String getDeviceId() {
        return mDeviceId;
```

```
    }

}
```

Definisci il tuo servizio (endpoint da colpire):

```
public interface Service {

    @POST("device")
    Call<Device> getDevice(@Body GetDeviceRequest getDeviceRequest);

}
```

Definire l'istanza singleton del client di rete:

```
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;
    }

}
```

Definire un oggetto modello semplice per il dispositivo:

```
public class Device {

    @SerializedName("id")
    private String mId;

    @SerializedName("type")
    private String mType;

    public String getId() {
        return mId;
    }
```

```
    public String getType() {
        return mType;
    }

}
```

Definire il controller per gestire le richieste per il 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
            }

        });
```

Lettura dell'URL del modulo XML con Retrofit 2

Useremo il retrofit 2 e SimpleXmlConverter per ottenere dati xml dall'URL e analizzare alla classe Java.

Aggiungi dipendenza allo script Gradle:

```
compile 'com.squareup.retrofit2:retrofit:2.1.0'
compile 'com.squareup.retrofit2:converter-simplexml:2.1.0'
```

Crea interfaccia

Crea anche wrapper classe xml nel nostro caso classe Rss

```
    public interface ApiDataInterface{

        // path to xml link on web site

        @GET (data/read.xml)
```

```
        Call<Rss> getData();

}
```

Funzione di lettura 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());
        }


    }
```

Questo è un esempio di classe Java con annotazioni SimpleXML

Ulteriori informazioni sulle annotazioni 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;
```

Leggi Retrofit2 online: https://riptutorial.com/it/android/topic/1132/retrofit2

Capitolo 204: Retrofit2 con RxJava

Examples

Retrofit2 con RxJava

Innanzitutto, aggiungi dipendenze rilevanti nel file 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'
    ....
}
```

Quindi crea il modello che desideri ricevere:

```
public class Server {
    public String name;
    public String url;
    public String apikey;
    public List<Site> siteList;
}
```

Creare un'interfaccia contenente i metodi utilizzati per lo scambio di dati con il server remoto:

```
public interface ApiServerRequests {

    @GET("api/get-servers")
    public Observable<List<Server>> getServers();
}
```

Quindi creare un'istanza di `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;
}
```

Quindi, ovunque dal codice, chiama il metodo:

```
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
       }
   });
```

Retrofit con RxJava per recuperare i dati in modo asincrono

Dal repository GitHub di RxJava, *RxJava è un'implementazione Java VM di Reactive Extensions: una libreria per la composizione di programmi asincroni e basati su eventi utilizzando sequenze osservabili. Estende il pattern dell'osservatore per supportare sequenze di dati / eventi e aggiunge operatori che consentono di comporre sequenze insieme in modo dichiarativo mentre estrapolano le preoccupazioni su argomenti come threading di basso livello, sincronizzazione, sicurezza dei thread e strutture di dati concorrenti.*

Retrofit è un client HTTP sicuro per tipo per Android e Java, utilizzando questo, gli sviluppatori possono rendere tutte le cose della rete molto più facili. Ad esempio, stiamo per scaricare alcuni JSON e mostrarli in RecyclerView come elenco.

Iniziare:

Aggiungi le dipendenze RxJava, RxAndroid e Retrofit nel tuo file build.gradle a livello di app:

```
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"
```

Definisci ApiClient e ApiInterface per scambiare dati dal server

```
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;
}}
```

interfaccia pubblica ApiInterface {

```
@GET("repos/{org}/{repo}/issues")
Observable<List<Issue>> getIssues(@Path("org") String organisation,
                                  @Path("repo") String repositoryName,
                                  @Query("page") int pageNumber);}
```

Nota che getRepos () restituisce un Osservabile e non solo un elenco di problemi.

Definire i modelli

Un esempio per questo è mostrato. Puoi utilizzare servizi gratuiti come JsonSchema2Pojo o questo.

```
public class Comment {

@SerializedName("url")
@Expose
private String url;
@SerializedName("html_url")
@Expose
private String htmlUrl;

//Getters and Setters
}
```

Crea istanza di Retrofit

```
ApiInterface apiService = ApiClient.getInstance().create(ApiInterface.class);
```

Quindi, utilizzare questa istanza per recuperare i dati dal server

```
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));
                }
            });
```

Ora, hai recuperato con successo i dati da un server usando Retrofit e RxJava.

Esempio di richieste nidificate: più richieste, combinazione di risultati

Supponiamo di avere un'API che ci consente di ottenere i metadati dell'oggetto in una singola richiesta (`getAllPets`) e altre richieste che hanno dati completi di una singola risorsa (`getSinglePet`). Come possiamo interrogarli tutti in una singola catena?

```
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;
    }
}
```

}

Leggi Retrofit2 con RxJava online: https://riptutorial.com/it/android/topic/7632/retrofit2-con-rxjava

Capitolo 205: Riconoscimento di attività

introduzione

Il riconoscimento dell'attività è il rilevamento dell'attività fisica dell'utente al fine di eseguire determinate azioni sul dispositivo, ad esempio il rilevamento di punti quando viene rilevata un'unità, disattivare la rete wifi quando il telefono è fermo o portare il volume della suoneria al massimo quando l'utente a piedi.

Examples

Google Play ActivityRecognitionAPI

Questo è solo un semplice esempio di come utilizzare ActivityRecognitionApi del servizio GooglePlay. Anche se questa è una grande libreria, non funziona su dispositivi su cui non è installato Google Play Services.

Docs for ActivityRecognition API

Manifesto

```
<!-- 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) {

    }
}
```

ActivityReceiver

```
public class ActivityReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

LocalBroadcastManager.getInstance(context).sendBroadcast(intent.setAction("activity"));
    }
}
```

Riconoscimento dell'attività PathSense

Il riconoscimento dell'attività PathSense è un'altra buona libreria per dispositivi che non dispongono di Google Play Services, in quanto hanno creato il proprio modello di riconoscimento delle attività, ma richiede agli sviluppatori di registrarsi su http://developer.pathsense.com per ottenere una chiave API e un ID client .

Manifesto

```
<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);
    }
}
```

Leggi Riconoscimento di attività online: https://riptutorial.com/it/android/topic/9831/riconoscimento-di-attivita

Capitolo 206: Rileva evento scossa in Android

Examples

Shake Detector in Android Esempio

```
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);
            }
        }
    }
}
```

Utilizzo del rilevamento scossa sismica

Sismico è una libreria di rilevamento del tremolio del dispositivo Android di Square. Per usarlo basta iniziare ad ascoltare gli eventi di shake emessi da esso.

```
@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();
}
```

Per definire una diversa soglia di accelerazione utilizzare `sd.setSensitivity(sensitivity)` con una `sensitivity` di `SENSITIVITY_LIGHT` , `SENSITIVITY_MEDIUM` , `SENSITIVITY_HARD` o qualsiasi altro valore intero ragionevole. I valori predefiniti forniti vanno da *11* a *15* .

Installazione

```
compile 'com.squareup:seismic:1.0.2'
```

Leggi Rileva evento scossa in Android online: https://riptutorial.com/it/android/topic/4501/rileva-evento-scossa-in-android

Capitolo 207: Rilevamento dei gesti

Osservazioni

Documentazione ufficiale: rilevamento di gesti comuni

Examples

Rileva swipe

```
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() {
    }

}
```

Applicato 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");
    }

});
```

Rilevamento del gesto di base

```
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;
    }

}
```

Leggi Rilevamento dei gesti online: https://riptutorial.com/it/android/topic/4711/rilevamento-dei-gesti

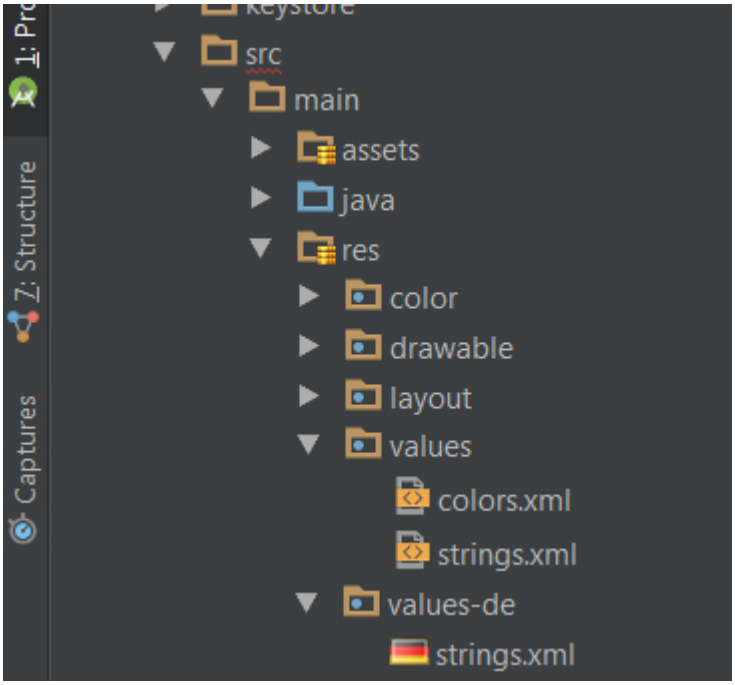
Capitolo 208: risorse

Examples

Traduci una stringa

Le stringhe possono essere internazionalizzate definendo un diverso string.xml per ogni lingua supportata.

Si aggiunge una nuova lingua creando una nuova directory dei valori con il codice della lingua ISO come suffisso. Ad esempio, quando si aggiunge un set tedesco, la struttura potrebbe essere simile a quanto segue:



Quando il sistema cerca la stringa richiesta, prima controlla l'xml specifico della lingua, se non viene trovato, viene restituito il valore dal file strings.xml predefinito. La chiave rimane la stessa per ogni lingua e solo il valore cambia.

Contenuti di esempio:

/res/values/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/values-fr/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resources>
    <string name="hello_world">Bonjour tout le monde !!!</string>
</resources>
```

Definisci stringhe

Le stringhe sono in genere memorizzate nel file di risorse `strings.xml` . Sono definiti utilizzando un elemento XML `<string>` .

Lo scopo di strings.xml è di consentire l'internazionalizzazione. È possibile definire un file strings.xml per ogni codice ISO. Pertanto, quando il sistema cerca la stringa "nome_app", verifica innanzitutto il file xml corrispondente alla lingua corrente e, se non viene trovato, cerca la voce nel file strings.xml predefinito. Ciò significa che puoi scegliere di localizzare solo alcune delle tue stringhe mentre non altre.

/res/values/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 volta definita una stringa in un file di risorse XML, può essere utilizzata da altre parti dell'app.

I file di progetto XML di un'app possono utilizzare un elemento `<string>` facendo riferimento a `@string/string_name` . Ad esempio, il file manifest di app (/manifests/AndroidManifest.xml) include la seguente riga per impostazione predefinita in Android Studio:

```
android:label="@string/app_name"
```

Questo dice ad Android di cercare una risorsa `<string>` chiamata "nome_app" da usare come nome per l'app quando viene installata o visualizzata in un launcher.

Un'altra volta che usereste una risorsa `<string>` da un file XML in Android si troverà in un file di layout. Ad esempio, il seguente rappresenta un TextView che mostra la stringa `hello_world` definita in precedenza:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world"/>
```

Puoi anche accedere alle `<string>` risorse dalla porzione java della tua app. Per richiamare la nostra stessa stringa `hello_world` dall'alto in una classe di attività, utilizzare:

```
String helloWorld = getString(R.string.hello_world);
```

Definisci array di stringhe

Per definire un array di stringhe scrivere in un file di risorse

res / valori / filename.xml

```
<string-array name="string_array_name">
    <item>text_string</item>
    <item>@string/string_id</item>
</string-array>
```

per esempio

res / valori / 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>
```

e usalo da java come

```
String[] strings = getResources().getStringArray(R.array.string_array_example;
Log.i("TAG",Arrays.toString(strings)));
```

Produzione

```
I/TAG: [HelloWorld, Hello World!]
```

Definire le dimensioni

Le dimensioni sono in genere memorizzate in nomi di file di risorse `dimens.xml` . Sono definiti usando un elemento `<dimen>` .

res / valori / 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>
```

Puoi usare diverse unità:

- **sp:** pixel indipendenti dalla scala. Per i caratteri.
- **dp:** pixel indipendenti dalla densità. Per tutto il resto.

- **pt:** punti
- **px:** Pixel
- **mm:** millimetri
- **im:** pollici

Le quote possono ora essere referenziate in XML con la sintassi `@dimen/name_of_the_dimension` .

Per esempio:

```
<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>
```

Definisci interi

Gli integer sono in genere memorizzati in un file di risorse denominato `integers.xml` , ma il nome del file può essere scelto arbitrariamente. Ogni intero è definito utilizzando un elemento `<integer>` , come mostrato nel seguente file:

res / valori / integers.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer name="max">100</integer>
</resources>
```

Gli integer possono ora essere referenziati in XML con la sintassi `@integer/name_of_the_integer` , come mostrato nell'esempio seguente:

```
<ProgressBar
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:max="@integer/max"/>
```

Definire l'array intero

Per definire una scrittura di array intero in un file di risorse

res / valori / filename.xml

```
<integer-array name="integer_array_name">
    <item>integer_value</item>
    <item>@integer/integer_id</item>
</integer-array>
```

per esempio

res / valori / 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>
```

e usalo da java come

```
int[] values = getResources().getIntArray(R.array.fibo);
Log.i("TAG",Arrays.toString(values)));
```

Produzione

```
I/TAG: [0, 1, 1, 2, 3, 5]
```

Definire i colori

I colori vengono solitamente memorizzati in un file di risorse denominato `colors.xml` nella `colors.xml /res/values/` .

Sono definiti da elementi `<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>
```

I colori sono rappresentati da valori di colore esadecimali per ciascun canale di colore `(0 - FF)` in uno dei seguenti formati:

- `#RGB`
- `#ARGB`
- `#RRGGBB`
- `#AARRGGBB`

Leggenda

- A - canale alfa - il valore 0 è completamente trasparente, il valore FF è opaco
- R - canale rosso
- G - canale verde

- B - canale blu

I colori definiti possono essere utilizzati in XML con la seguente sintassi `@color/name_of_the_color`

Per esempio:

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/blackOverlay">
```

Uso dei colori nel codice

Questi esempi presuppongono che `this` tratti di un riferimento di attività. Un riferimento al contesto può essere utilizzato anche al suo posto.

1.6

```
int color = ContextCompat.getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

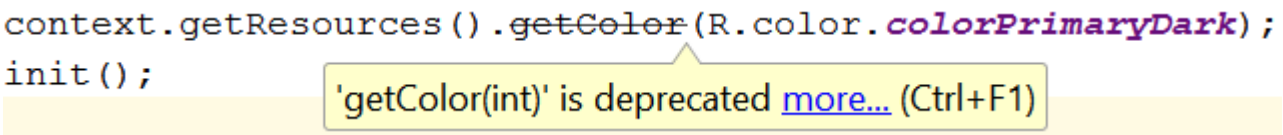
6.0

```
int color = this.getResources().getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

Nella dichiarazione sopra `colorPrimary` , `colorPrimaryDark` e `colorAccent` vengono utilizzati per definire i colori di progettazione del materiale che verranno utilizzati nella definizione del tema Android personalizzato in `styles.xml` . Vengono aggiunti automaticamente quando viene creato un nuovo progetto con Android Studio.

Ottenere risorse senza avvisi "deprecati"

Utilizzando l'API di Android 23 o superiore, molto spesso tale situazione può essere vista:



Questa situazione è causata dal cambiamento strutturale dell'API di Android per quanto riguarda il recupero delle risorse.
Ora la funzione:

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

dovrebbe essere usato. Ma la libreria `android.support.v4` ha un'altra soluzione.

Aggiungi la seguente dipendenza al file `build.gradle` :

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

Quindi sono disponibili tutti i metodi dalla libreria di supporto:

```
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));
```

Inoltre è possibile utilizzare più metodi dalla libreria di supporto:

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

Definire una risorsa di menu e utilizzarla all'interno di Attività / Frammento

Definire un menu in **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>
```

Per ulteriori opzioni di configurazione, fare riferimento a: Risorsa del menu

`Activity` interna:

```
@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);
}
```

Per invocare i metodi precedenti durante la visualizzazione della vista, chiamare `getActivity().invalidateOptionsMenu();`

All'interno di **`Fragment`** è necessaria una chiamata aggiuntiva:

```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    setHasOptionsMenu(true);
    super.onCreateView(inflater, container, savedInstanceState);
}
```

Formattazione delle stringhe in strings.xml

La definizione di stringhe nel file strings.xml consente anche la formattazione delle stringhe. L'unica avvertenza è che la stringa dovrà essere trattata nel codice come sotto, invece di collegarlo semplicemente a un layout.

```
<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);
```

Nell'esempio sopra,
% 1 $ s
'%' si separa dai caratteri normali,
'1' indica il primo parametro,
'$' è usato come separatore tra numero di parametro e tipo,
's' indica il tipo di stringa ('d' è usato per intero)

Nota che `getString()` è un metodo di `Context` o `Resources` , cioè puoi usarlo direttamente all'interno di un'istanza di `Activity` , altrimenti puoi usare `getActivity().getString()` o `getContext().getString()` rispettivamente.

Definire un elenco di stati colore

Gli elenchi degli stati del colore possono essere utilizzati come colori, ma cambiano a seconda dello stato della vista per cui sono utilizzati.

Per definirne uno, creare un file di risorse in `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>
```

Gli articoli vengono valutati nell'ordine in cui sono definiti e viene utilizzato il primo elemento i cui stati specificati corrispondono allo stato corrente della vista. Pertanto, è buona norma specificare un catch-all alla fine, senza specificare alcun selettore di stato.

Ogni elemento può utilizzare un letterale di colore o fare riferimento a un colore definito da qualche altra parte.

Definire i plurali della stringa

Per distinguere tra stringhe plurali e singolari, puoi definire un plurale nel tuo file *strings.xml* ed elencare le diverse quantità, come mostrato nell'esempio seguente:

```
<?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>
```

È possibile accedere a questa definizione dal codice Java utilizzando il metodo `getQuantityString()` della classe `Resources` , come illustrato nell'esempio seguente:

```
getResources().getQuantityString(R.plurals.hello_people, 3, 3);
```

Qui, il primo parametro `R.plurals.hello_people` è il nome della risorsa. Il secondo parametro (`3` in questo esempio) viene utilizzato per selezionare la stringa di `quantity` corretta. Il terzo parametro (anche `3` in questo esempio) è l'argomento di formato che verrà utilizzato per sostituire lo specificatore di formato `%d` .

I valori di quantità possibili (elencati in ordine alfabetico) sono:

```
few
many
one
other
two
zero
```

È importante notare che non tutti i locali supportano ogni denominazione di `quantity` . Ad esempio, la lingua cinese non ha il concetto di `one` oggetto. L'inglese non ha un oggetto `zero` , poiché è grammaticalmente uguale `other` . Le istanze di `quantity` non supportate verranno contrassegnate dall'IDE come avvertenze sui rifiuti, ma non causeranno errori di complicazione se vengono utilizzate.

Importa array di oggetti definiti in risorse

Ci sono casi in cui gli oggetti personalizzati devono essere creati e definiti nelle risorse dell'applicazione. Tali oggetti possono essere composti da tipi semplici di `Java` , ad esempio `Integer` , `Float` , `String` .

Ecco l'esempio di come importare un oggetto definito nelle risorse dell'applicazione. L'oggetto `Category` comprende 3 proprietà della categoria:

- ID
- Colore
- Nome

Questo `POJO` ha il suo equivalente nel file `categories.xml` , in cui ogni array ha le stesse proprietà definite per ogni categoria.

1. **Crea un modello per il tuo oggetto:**

```
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 il file nella cartella `res/values` :**

```
categories.xml
```

3. **Componi ogni modello composto da risorse:**

```
<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. **Definire un array nel file delle risorse:**

```
<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. **Creare una funzione per importarli:**

```
@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 patch

9 Le patch sono immagini **estensibili** in cui le aree che possono essere allungate sono definite da pennarelli neri su un bordo trasparente.

C'è un ottimo tutorial qui .
Nonostante sia così vecchio, è ancora così prezioso e ha aiutato molti di noi a comprendere a

fondo le 9 patch.

Sfortunatamente, recentemente quella pagina è stata messa giù per un po '(è attualmente di nuovo attiva).

Quindi, la necessità di avere una copia fisica di quella pagina per gli sviluppatori Android sul nostro server affidabile / s.

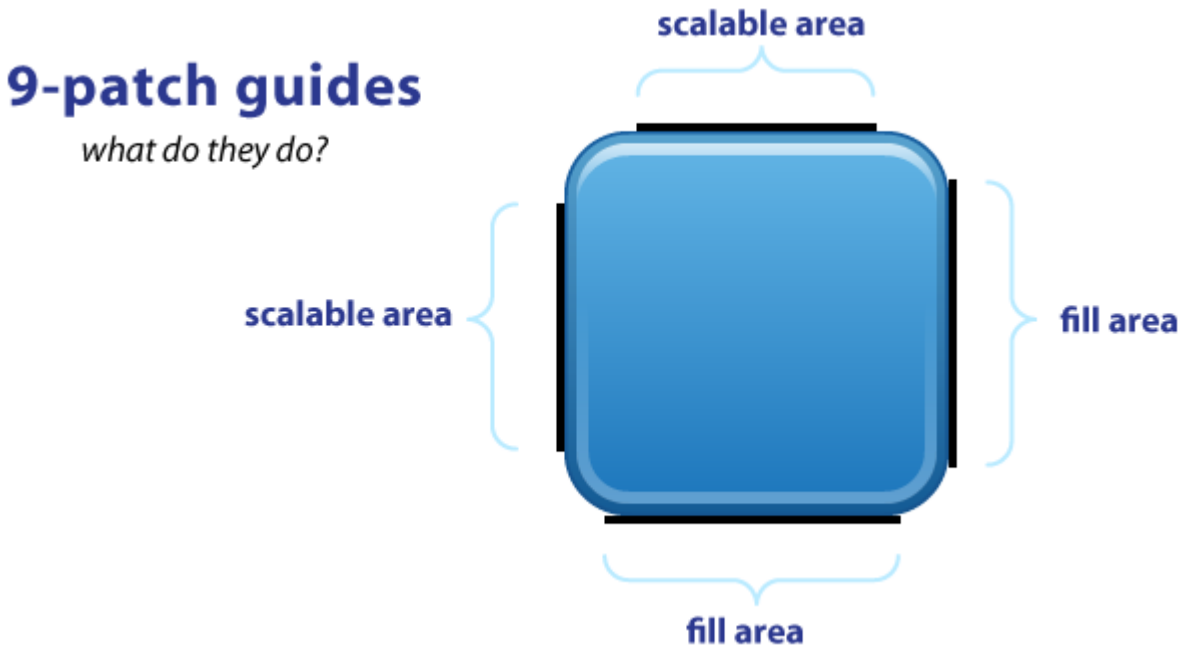
Ecco qui.

UNA GUIDA SEMPLICE A 9-PATCH PER ANDROID UI 18 maggio 2011

Mentre stavo lavorando alla mia prima app per Android, ho trovato 9 patch (aka 9.png) per essere confuso e scarsamente documentato. Dopo un po ', alla fine ho capito come funziona e ho deciso di mettere insieme qualcosa per aiutare gli altri a capirlo.

Fondamentalmente, 9-patch usa la trasparenza png per fare una forma avanzata di 9-slice o scale9. Le guide sono dritte, linee nere da 1 pixel disegnate sul bordo dell'immagine che definiscono il ridimensionamento e il riempimento dell'immagine. Denominando il file immagine nome.9.png, Android riconoscerà il formato 9.png e utilizzerà le guide nere per ridimensionare e riempire i bitmap.

Ecco una mappa guida di base:



Come puoi vedere, hai delle guide su ciascun lato dell'immagine. Le guide TOP e LEFT servono per ridimensionare l'immagine (ad esempio 9 sezioni), mentre le guide RIGHT e BOTTOM definiscono l'area di riempimento.

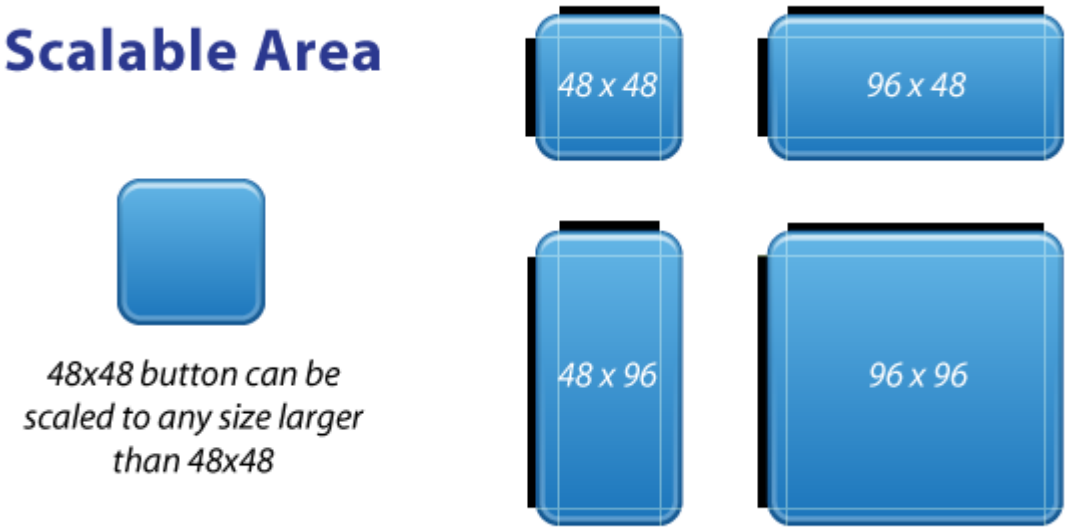
Le linee guida nere sono troncate / rimosse dalla tua immagine - non verranno mostrate nell'app.

Le guide devono avere un solo pixel, quindi se vuoi un pulsante 48 × 48, il tuo png sarà effettivamente 50 × 50. Qualsiasi cosa più spessa di un pixel rimarrà parte dell'immagine. (I miei esempi hanno guide larghe 4 pixel per una migliore visibilità e dovrebbero essere solo 1 pixel).

Le tue guide devono essere nere (# 000000). Anche una leggera differenza di colore (# 000001) o alpha causerà un errore e si allungherà normalmente. Anche questo fallimento non sarà ovvio *, fallisce silenziosamente! Sì. Veramente. Ora sai.

Inoltre, tieni presente che l'area rimanente del contorno di un pixel deve essere completamente trasparente. Questo include i quattro angoli dell'immagine - quelli dovrebbero essere sempre chiari. Questo può essere un problema più grande di quello che ti rendi conto. Ad esempio, se ridimensioni un'immagine in Photoshop, aggiungerai pixel anti-alias che potrebbero includere pixel quasi invisibili che causeranno anche il fallimento *. Se è necessario ridimensionare in Photoshop, utilizzare l'impostazione Vicini più vicini nel menu a discesa Ridimensionamento immagine (nella parte inferiore del menu a comparsa Dimensioni immagine) per mantenere i bordi nitidi sulle guide.

* (aggiornato 1/2012) Questa è in realtà una "correzione" nell'ultimo kit di sviluppo. In precedenza si manifestava quando tutte le altre immagini e risorse si rompevano all'improvviso, non l'immagine a 9 patch effettivamente rotta.

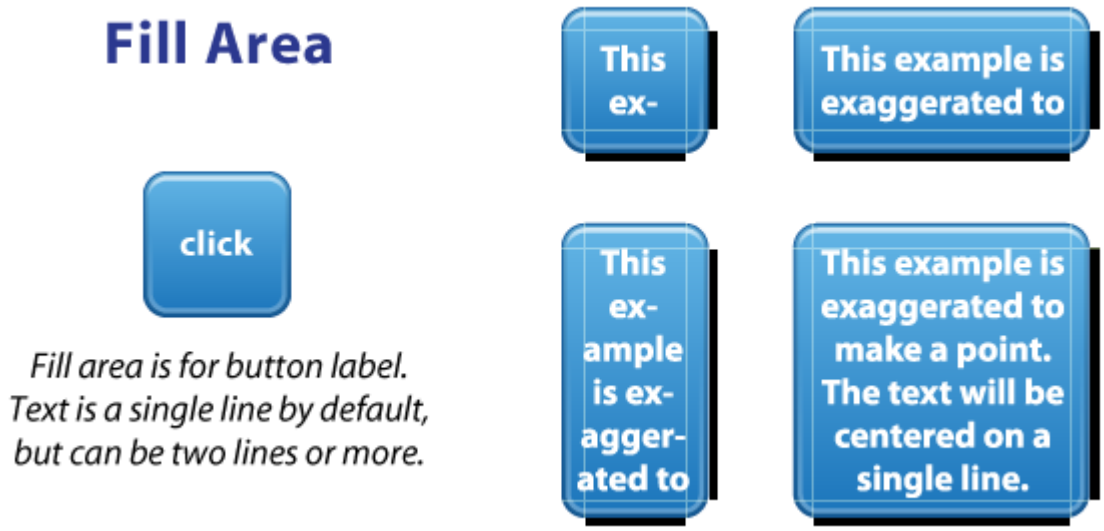


Le guide TOP e LEFT sono utilizzate per definire la porzione scalabile dell'immagine: SINISTRA per l'altezza di ridimensionamento, TOP per la larghezza di ridimensionamento. Usando l'immagine di un pulsante come esempio, questo significa che il pulsante può estendersi orizzontalmente e verticalmente all'interno della porzione nera e tutto il resto, come gli angoli, rimarrà della stessa dimensione. Ti consente di avere pulsanti che possono ridimensionare a qualsiasi dimensione e mantenere un aspetto uniforme.

È importante notare che le immagini a 9 patch non si ridimensionano, ma si scalano. Quindi è meglio iniziare il più piccolo possibile.

Inoltre, è possibile tralasciare porzioni nel mezzo della linea di scala. Ad esempio, se hai un pulsante con un bordo lucido e affilato nel mezzo, puoi lasciare alcuni pixel nel mezzo della guida

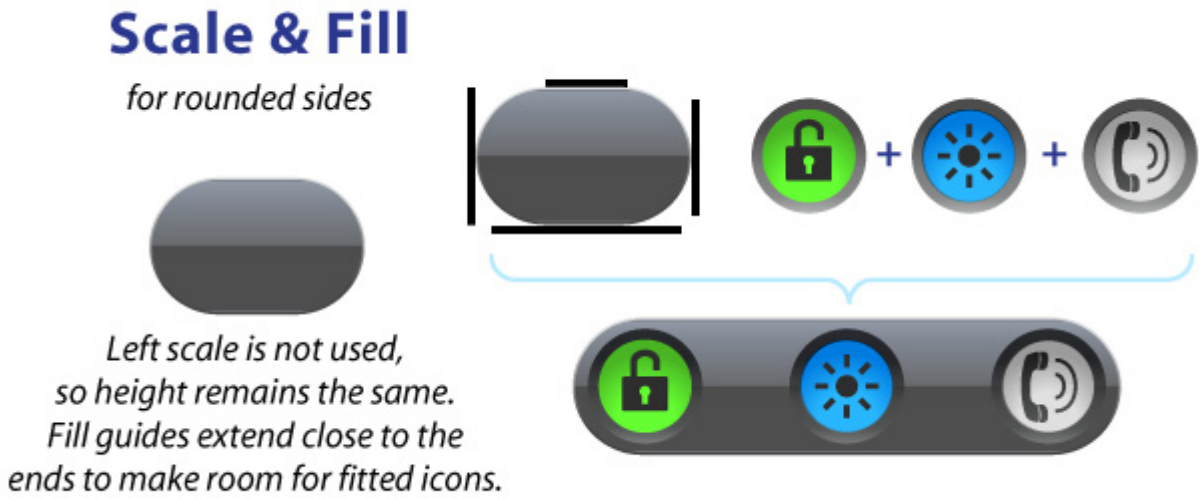
LEFT. L'asse orizzontale centrale della tua immagine non si ridimensiona, solo le parti sopra e sotto di essa, così il tuo gloss non risulterà anti-alias o sfocato.



Le guide dell'area di riempimento sono facoltative e forniscono un modo per definire l'area per elementi come l'etichetta di testo. Riempi determina la quantità di spazio all'interno dell'immagine per posizionare il testo, un'icona o altre cose. 9-patch non è solo per i pulsanti, funziona anche per le immagini di sfondo.

Il precedente esempio di pulsante ed etichetta è esagerato semplicemente per spiegare l'idea del riempimento: l'etichetta non è completamente accurata. Per essere onesti, non ho esperienza di come Android fa le etichette su più righe poiché un'etichetta di pulsante è in genere una singola riga di testo.

Infine, ecco una buona dimostrazione di come le guide di scala e riempimento possono variare, come ad esempio LinearLayout con un'immagine di sfondo e lati completamente arrotondati:



Con questo esempio, la guida LEFT non viene utilizzata, ma dobbiamo ancora avere una guida. L'immagine di sfondo non si scala verticalmente; scala solo orizzontalmente (in base alla guida

TOP). Osservando le guide di riempimento, le guide DESTRA e BASSA si estendono oltre il punto in cui incontrano i bordi curvi dell'immagine. Questo mi consente di posizionare i miei pulsanti rotondi vicino ai bordi dello sfondo per un look aderente e aderente.

Quindi è così. 9 patch è semplicissimo, una volta ottenuto. Non è un modo perfetto per eseguire il ridimensionamento, ma le guide della scala di riempimento e multilinea offrono una maggiore flessibilità rispetto al tradizionale 9-slice e scale9. Fare un tentativo e lo capirai rapidamente.

Livello di trasparenza del colore (alfa)

Hex Opacity Values

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

Se si desidera impostare il 45% sul rosso.

```
<color name="red_with_alpha_45">#73FF0000</color>
```

valore esadecimale per rosso - # FF0000

Puoi aggiungere 73 per il 45% di opacità nel prefisso - # 73FF0000

Lavorare con il file strings.xml

Una risorsa stringa fornisce stringhe di testo per l'applicazione con stile e formattazione del testo opzionali. Esistono tre tipi di risorse che possono fornire all'applicazione stringhe:

Stringa

```
XML resource that provides a single string.
```

Sintassi:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="string_name">text_string</string>
</resources>
```

E per usare questa stringa nel layout:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/string_name" />
```

Array di stringhe

```
XML resource that provides an array of strings.
```

Sintassi:

```
    <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);
```

Quantità di corde (plurali)

```
XML resource that carries different strings for pluralization.
```

Sintassi:

```
<?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);
```

Leggi risorse online: https://riptutorial.com/it/android/topic/108/risorse

Capitolo 209: RoboGuice

Examples

Semplice esempio

RoboGuice è un framework che porta la semplicità e la facilità di Dependency Injection su Android, utilizzando la libreria Guice di 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 );
    }
}
```

Installazione per progetti Gradle

Aggiungi il seguente pom alla sezione delle dipendenze del tuo file build gradle:

```
project.dependencies {
    compile 'org.roboguice:roboguice:3.+'
    provided 'org.roboguice:roboblender:3.+'
}
```

@ContentView annotation

L'annotazione @ContentView può essere utilizzata per alleviare ulteriormente lo sviluppo delle attività e sostituire l'istruzione 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!");
    }
}
```

Annotazione @InjectResource

È possibile iniettare qualsiasi tipo di risorsa, stringhe, animazioni, disegni, ecc.

Per iniettare la tua prima risorsa in un'attività, dovrai:

- Eredita da RoboActivity
- Annota le tue risorse con @InjectResource

Esempio

```
@InjectResource(R.string.app_name) String name;

@InjectResource(R.drawable.ic_launcher) Drawable icLauncher;

@InjectResource(R.anim.my_animation) Animation myAnimation;
```

@ Annotazione di InjectView

È possibile iniettare qualsiasi vista utilizzando l'annotazione @InjectView:

Avrai bisogno di:

- Eredita da RoboActivity
- Imposta la visualizzazione del contenuto
- Annota le tue viste con @InjectView

Esempio

```
@InjectView(R.id.textView1) TextView textView1;

@InjectView(R.id.textView2) TextView textView2;

@InjectView(R.id.imageView1) ImageView imageView1;
```

Introduzione a RoboGuice

`RoboGuice` è un framework che porta la semplicità e la facilità di Dependency Injection su Android, utilizzando la libreria Guice di Google.

RoboGuice 3 riduce il codice dell'applicazione. Meno codice significa minori opportunità di bug. Rende anche più semplice il tuo codice da seguire - non è più il tuo codice ingombrato dai meccanismi della piattaforma Android, ma ora può concentrarsi sulla logica effettiva esclusiva della tua applicazione.

Per darti un'idea, dai un'occhiata a questo semplice esempio di una tipica `Activity` Android:

```
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 );
        }
    }
```

Questo esempio è composto da 19 righe di codice. Se stai cercando di leggere su `onCreate()` , devi saltare più di 5 righe di inizializzazione di default per trovare l'unico che conta davvero: `name.setText()` . E le attività complesse possono finire con molto più di questo tipo di codice di inizializzazione.

Confronta questo con la stessa app, scritta usando `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 );
       }
   }
```

L'obiettivo di RoboGuice è quello di rendere il tuo codice sulla tua app, piuttosto che su tutto il codice di inizializzazione e ciclo di vita che devi mantenere in Android.

annotazioni:

Annotazione @ContentView:

L'annotazione @ContentView può essere utilizzata per alleviare ulteriormente lo sviluppo delle attività e sostituire l'istruzione 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!");
        }
    }
```

Annotazione @InjectResource:

Per prima cosa è necessaria un'attività che erediti da RoboActivity. Quindi, assumendo che tu

abbia un'animazione my_animation.xml nella tua cartella res / anim, ora puoi fare riferimento ad essa con un'annotazione:

```
public class MyActivity extends RoboActivity {
    @InjectResource(R.anim.my_animation) Animation myAnimation;
    // the rest of your code
}
```

Annotazione @Inject:

Assicurati che la tua attività si estenda da RoboActivity e annoti il tuo membro del servizio di sistema con @Inject. Roboguice farà il 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();
```

Oltre a Views, risorse, servizi e altre cose specifiche per Android, RoboGuice può iniettare Plain Old Java Objects. Di default Roboguice chiamerà un costruttore no argument sul tuo POJO

```
class MyActivity extends RoboActivity {
    @Inject Foo foo; // this will basically call new Foo();
}
```

Leggi RoboGuice online: https://riptutorial.com/it/android/topic/2563/roboguice

Capitolo 210: Robolectric

introduzione

Il test delle unità sta prendendo un pezzo di codice e testandolo in modo indipendente senza altre dipendenze o parti del sistema in esecuzione (ad esempio il database).

Robolectric è un framework di test unitario che dissolve il jar Android SDK in modo da poter testare lo sviluppo della tua app Android. I test vengono eseguiti all'interno della JVM sulla workstation in pochi secondi.

Combinandoli entrambi è possibile eseguire test rapidi su JVN utilizzando ancora le API di Android.

Examples

Test di Robolectric

```
@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!");
  }
}
```

Configurazione

Per configurare robolectric aggiungere `@Config` annotazione `@Config` per testare la classe o il metodo.

Esegui con una classe di applicazione personalizzata

```
@RunWith(RobolectricTestRunner.class)
@Config(application = MyApplication.class)
public final class MyTest {
}
```

Imposta l'SDK di destinazione

```
@RunWith(RobolectricTestRunner.class)
@Config(sdk = Build.VERSION_CODES.LOLLIPOP)
public final class MyTest {
}
```

Esegui con manifest personalizzato

Quando specificato, robolectric avrà un aspetto relativo alla directory corrente. Il valore predefinito è `AndroidManifest.xml`

Le risorse e le risorse saranno caricate rispetto al manifest.

```
@RunWith(RobolectricTestRunner.class)
@Config(manifest = "path/AndroidManifest.xml")
public final class MyTest {
}
```

Usa qualificazioni

Eventuali qualificazioni possono essere trovate nei documenti android .

```
@RunWith(RobolectricTestRunner.class)
public final class MyTest {

    @Config(qualifiers = "sw600dp")
    public void testForTablet() {
    }
}
```

Leggi Robolectric online: https://riptutorial.com/it/android/topic/8743/robolectric

Capitolo 211: Schermi di supporto con diverse risoluzioni, dimensioni

Osservazioni

Termini e concetti

Dimensione dello schermo

Dimensione fisica effettiva, misurata come diagonale dello schermo. Per semplicità, Android raggruppa tutte le dimensioni effettive dello schermo in quattro dimensioni generalizzate: piccola, normale, grande e molto grande.

Densità dello schermo

La quantità di pixel all'interno di un'area fisica dello schermo; di solito indicato come dpi (punti per pollice). Ad esempio, uno schermo con densità "bassa" ha meno pixel all'interno di una determinata area fisica, rispetto a uno schermo con densità "normale" o "alta". Per semplicità, Android raggruppa tutte le densità dello schermo effettive in sei densità generalizzate: bassa, media, alta, extra-alta, extra-extra-alta e extra-extra-extra-alta.

Orientamento

L'orientamento dello schermo dal punto di vista dell'utente. Questo è orizzontale o verticale, il che significa che le proporzioni dello schermo sono rispettivamente ampie o alte. Si noti che non solo i dispositivi diversi operano in diversi orientamenti per impostazione predefinita, ma l'orientamento può cambiare in fase di esecuzione quando l'utente ruota il dispositivo. Risoluzione Il numero totale di pixel fisici su uno schermo. Quando si aggiunge il supporto per più schermi, le applicazioni non funzionano direttamente con la risoluzione; le applicazioni dovrebbero riguardare solo le dimensioni e la densità dello schermo, come specificato dai gruppi di dimensioni e densità generalizzate. Pixel indipendente dalla densità (dp) Un'unità di pixel virtuale che dovresti usare quando definisci il layout dell'interfaccia utente, per esprimere le dimensioni del layout o la posizione in un modo indipendente dalla densità. Il pixel indipendente dalla densità è equivalente a un pixel fisico su uno schermo a 160 dpi, che è la densità di base assunta dal sistema per uno schermo a densità "media". In fase di esecuzione, il sistema gestisce in modo trasparente qualsiasi ridimensionamento delle unità dp, in base alle necessità, in base alla densità effettiva dello schermo in uso. La conversione delle unità dp in pixel dello schermo è semplice: px = dp * (dpi / 160). Ad esempio, su uno schermo a 240 dpi, 1 dp equivale a 1,5 pixel fisici. Dovresti sempre utilizzare le unità dp quando definisci l'interfaccia utente della

tua applicazione, per garantire la corretta visualizzazione dell'interfaccia utente su schermi con densità diverse.

---

unità

- ### px

  Pixel: corrisponde ai pixel effettivi sullo schermo.

- ### nel

  Pollici - in base alle dimensioni fisiche dello schermo. 1 pollice = 2,54 centimetri

- ### mm

  Millimetri - in base alle dimensioni fisiche dello schermo.

- ### pt

  Punti: 1/72 di pollice in base alle dimensioni fisiche dello schermo.

- ### dp o dip

  Pixel indipendenti dalla densità: un'unità astratta basata sulla densità fisica dello schermo. Queste unità sono relative a uno schermo da 160 dpi, quindi un dp è un pixel su uno schermo da 160 dpi. Il rapporto tra dp-to-pixel cambierà con la densità dello schermo, ma non necessariamente in proporzione diretta. Nota: il compilatore accetta sia "dip" che "dp", sebbene "dp" sia più coerente con "sp".

- ### sp

  Pixel indipendente dalla scala: è come l'unità dp, ma è anche ridimensionato dalla preferenza della dimensione del carattere dell'utente. Si consiglia di utilizzare questa unità quando si specificano le dimensioni dei caratteri, in modo che vengano regolate sia per la densità dello schermo che per le preferenze dell'utente. Dalla comprensione dell'indipendenza dalla densità in Android:

Unità	Descrizione	Unità per pollice fisico	Densità indipendente	Stessa dimensione fisica su ogni schermo
px	pixel	Varia	No	No
nel	Pollici	1	sì	sì

Unità	Descrizione	Unità per pollice fisico	Densità indipendente	Stessa dimensione fisica su ogni schermo
mm	millimetri	25,4	sì	sì
pt	Punti	72	sì	sì
dp	Density Independent Pixels	~ 160	sì	No
sp	Scala pixel indipendenti	~ 160	sì	No

Riferimenti:

- https://developer.android.com/guide/practices/screens_support.html
- http://developer.android.com/guide/topics/resources/more-resources.html

Examples

Utilizzo dei qualificatori di configurazione

Android supporta diversi qualificatori di configurazione che consentono di controllare il modo in cui il sistema seleziona le risorse alternative in base alle caratteristiche della schermata corrente del dispositivo. Un qualificatore di configurazione è una stringa che è possibile aggiungere a una directory di risorse nel progetto Android e specifica la configurazione per la quale sono progettate le risorse interne.

Per utilizzare un qualificatore di configurazione:

1. Crea una nuova directory nella directory res / del tuo progetto e nominala usando il formato: `<resources_name>-<qualifier>` . `<resources_name>` è il nome della risorsa standard (come drawable o layout).
2. `<qualifier>` è un qualificatore di configurazione, che specifica la configurazione dello schermo per cui queste risorse devono essere utilizzate (come hdpi o xlarge).

Ad esempio, le seguenti directory delle risorse dell'applicazione forniscono diversi progetti di layout per diverse dimensioni dello schermo e diversi drawable. Usa la `mipmap/` cartelle per le icone di avvio.

```
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
```

Conversione di dp e sp in pixel

Quando devi impostare un valore in pixel per qualcosa come `Paint.setTextSize` ma desideri comunque che venga ridimensionato in base al dispositivo, puoi convertire i valori dp e 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);
```

In alternativa, puoi convertire una risorsa di dimensione in pixel se hai un contesto da cui caricare la risorsa.

```
<?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);
```

Dimensioni del testo e diverse dimensioni dello schermo Android

A volte, è meglio avere solo tre opzioni

```
 style="@android:style/TextAppearance.Small"
 style="@android:style/TextAppearance.Medium"
 style="@android:style/TextAppearance.Large"
```

Usa piccoli e grandi per differenziarti dalle normali dimensioni dello schermo.

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="@android:style/TextAppearance.Small"/>
```

Per normale, non devi specificare nulla.

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

Usando questo, puoi evitare di testare e specificare le dimensioni per le diverse dimensioni dello schermo.

Leggi Schermi di supporto con diverse risoluzioni, dimensioni online:
https://riptutorial.com/it/android/topic/1086/schermi-di-supporto-con-diverse-risoluzioni--dimensioni

Capitolo 212: Scorri per aggiornare

Sintassi

1. **setColorSchemeResources** imposta i colori **dell'indicatore** SwipeToRefreshLayout
2. **setOnRefreshListener** imposta cosa fare quando si **scorre il** layout
3. **app: layout_behavior = "@ string / appbar_scrolling_view_behavior"** se disponi di una barra degli strumenti con il layout, aggiungi questo con gli scrollFlag nella barra degli strumenti e la barra degli strumenti scorrerà verso l'alto mentre scorri verso il basso e fai scorrere di nuovo mentre scorri verso l'alto.

Examples

Scorri per aggiornare con RecyclerView

Per aggiungere un layout **Swipe a Aggiorna** con **RecyclerView,** aggiungi quanto segue al tuo file di layout Attività / Frammento:

```
<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>
```

Nella tua attività / frammento aggiungi quanto segue per inizializzare **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
        }
    });
```

Come aggiungere Swipe-to-Refresh alla tua app

Assicurati che la seguente dipendenza venga aggiunta al file `build.gradle` dell'app in dipendenze:

```
compile 'com.android.support:support-core-ui:24.2.0'
```

Quindi aggiungi `SwipeRefreshLayout` nel tuo layout:

```
<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>
```

Infine implementa il listener `SwipeRefreshLayout.OnRefreshListener` .

```
mSwipeRefreshLayout = (SwipeRefreshLayout) findViewById(R.id.swipe_refresh_layout);
mSwipeRefreshLayout.setOnRefreshListener(new OnRefreshListener() {
    @Override
    public void onRefresh() {
         // your code
    }
});
```

Leggi Scorri per aggiornare online: https://riptutorial.com/it/android/topic/5241/scorri-per-aggiornare

Capitolo 213: Scrittura di test dell'interfaccia utente - Android

introduzione

Focus di questo documento è quello di rappresentare obiettivi e modi in cui scrivere UI di Android e test di integrazione. Espresso e UIAutomator sono forniti da Google, pertanto è necessario concentrarsi su questi strumenti e sui rispettivi wrapper, ad esempio Appium, Spoon ecc.

Sintassi

- **Risorsa al minimo**
- String getName () - Restituisce il nome della risorsa inattiva (utilizzata per la registrazione e l'idempotenza della registrazione).
- booleano isIdleNow () - Restituisce vero se la risorsa è al momento inattiva.
- void registerIdleTransitionCallback (IdlingResource.ResourceCallback callback) - Registra il dato IdlingResource.ResourceCallback con la risorsa

Osservazioni

Regole di JUnit:

Come puoi vedere nell'esempio MockWebServer e ActivityTestRule rientrano tutti nella categoria di regole JUnit che puoi creare tu stesso che poi dovrebbero essere eseguite per ogni test definendo il suo comportamento @see: https://github.com/junit-team/junit4/ wiki / regole

Appium

parametri

Poiché i parametri presentano alcuni problemi, posizionali qui finché il bug della documentazione non viene risolto:

Parametro	Dettagli
Class activityClass	quale attività iniziare
initialTouchMode	se l'attività viene posizionata in modalità touch all'avvio: https://android-developers.blogspot.de/2008/12/touch-mode.html
launchActivity	vero se l'attività deve essere avviata una volta per metodo di prova. Verrà

Parametro	Dettagli
	avviato prima del primo metodo Before e terminato dopo l'ultimo metodo After.

Examples

Esempio di MockWebServer

Nel caso in cui le tue attività, i frammenti e l'interfaccia utente richiedano qualche elaborazione in background, una cosa buona da usare è un MockWebServer che gira localmente su un dispositivo Android che porta un ambiente chiuso e testabile per l'interfaccia utente.

https://github.com/square/okhttp/tree/master/mockwebserver

Il primo passo è includere la dipendenza gradle:

```
testCompile 'com.squareup.okhttp3:mockwebserver:(insert latest version)'
```

Ora i passaggi per l'esecuzione e l'utilizzo del server di simulazione sono:

- creare un oggetto server fittizio
- avviarlo all'indirizzo e alla porta specifici (in genere localhost: portnumber)
- accodare le risposte per richieste specifiche
- inizia il test

Questo è ben spiegato nella pagina github del mockwebserver, ma nel nostro caso vogliamo qualcosa di più bello e riusabile per tutti i test, e le regole di JUnit saranno ben giocate qui:

```
/**
 *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”);
        }
    }
}
```

Ora supponiamo di avere la stessa identica attività come nell'esempio precedente, solo in questo caso quando spingiamo l'app pulsante recupereremo qualcosa dalla rete, ad esempio: https://someapi.com/name

Ciò restituirebbe una stringa di testo che verrebbe concatenata nel testo dello snackbar, ad esempio NOME + testo digitato.

```
/**
* 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;
    }
```

Vorrei suggerire di avvolgere il dispatcher in una sorta di Builder in modo da poter facilmente concatenare e aggiungere nuove risposte per i tuoi schermi. per esempio

```
 return newDispatcherBuilder()
            .withSerializedJSONBody("/authenticate", Mocks.getAuthenticationResponse())
            .withSerializedJSONBody("/getUserInfo", Mocks.getUserInfo())
            .withSerializedJSONBody("/checkNotBot", Mocks.checkNotBot());
```

IdlingResource

Il potere delle risorse inattive consiste nel non dover attendere l'elaborazione di alcune app (networking, calcoli, animazioni, ecc.) Per finire con `sleep()` , che porta alla falla e / o prolunga i test eseguiti. La documentazione ufficiale può essere trovata qui .

Implementazione

Ci sono tre cose che devi fare quando si implementa `IdlingResource` interfaccia `IdlingResource` :

- **`getName()`** - Restituisce il nome della risorsa inattiva.
- **`isIdleNow()`** - Controlla se il tuo oggetto xyz, operazione, ecc. è inattivo al momento.
- **`registerIdleTransitionCallback`** (`IdlingResource.ResourceCallback` callback) - Fornisce un callback da chiamare quando il tuo oggetto passa in standby.

Ora dovresti creare la tua logica e determinare quando la tua app è inattiva e quando no, dato che dipende dall'app. Di seguito troverai un semplice esempio, giusto per mostrare come funziona. Esistono altri esempi online, ma l'implementazione specifica di app porta a specifiche implementazioni di risorse inattive.

GLI APPUNTI

- Ci sono stati alcuni esempi di Google in cui hanno inserito `IdlingResources` nel codice dell'app. **Non farlo.** Presumibilmente l'hanno messo lì solo per mostrare come funzionano.
- Mantenere il proprio codice pulito e mantenere un unico principio di responsabilità dipende da voi!

Esempio

Diciamo che hai un'attività che fa cose strane e richiede molto tempo per il caricamento del frammento e quindi i test Espresso falliscono non potendo trovare risorse dal tuo frammento (dovresti cambiare il modo in cui la tua attività viene creata e quando per accelerarlo). Ma in ogni caso per mantenerlo semplice, l'esempio seguente mostra come dovrebbe essere.

Il nostro esempio di risorsa inattiva otterrebbe due oggetti:

- Il **tag** del frammento che devi trovare e in attesa di essere collegato all'attività.
- Un oggetto **FragmentManager** che viene utilizzato per trovare il frammento.

```
/**
 * 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;
    }
}
```

Ora che hai scritto `IdlingResource` , devi usarlo da qualche parte, giusto?

uso

Saltiamo l'intera configurazione della classe di test e guardiamo come apparirebbe un caso di test:

```
@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);
}
```

Combinazione con la regola JUnit

Questo non è difficile; puoi anche applicare la risorsa inattiva sotto forma di una regola di test JUnit. Ad esempio, diciamo che hai qualche SDK che contiene Volley e vuoi che Espresso lo aspetti. Invece di esaminare ogni caso di test o applicarlo in fase di configurazione, è possibile creare una regola JUnit e scrivere semplicemente:

```
@Rule
public final SDKIdlingRule mSdkIdlingRule = new
```

```
SDKIdlingRule(SDKInstanceHolder.getInstance());
```

Ora, poiché questo è un esempio, non darlo per scontato; tutto il codice qui è immaginario e usato solo a scopo dimostrativo:

```
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);
                    }
                }
            }
        };
    }
}
```

Leggi Scrittura di test dell'interfaccia utente - Android online:
https://riptutorial.com/it/android/topic/3530/scrittura-di-test-dell-interfaccia-utente---android

Capitolo 214: SearchView

Examples

AppCompat SearchView con RxBindings watcher

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 in barra degli strumenti con frammento

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);
    }
}
```

Schermata di riferimento:

Hello Android

`menu.xml` , abbiamo bisogno di capire che dipende completamente dallo stile applicato alla barra degli strumenti sottostante. Per ottenere il tema, la barra degli strumenti applica i seguenti passaggi.

Crea uno stile in `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>
```

Applica lo stile alla barra degli strumenti.

```
<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" />
```

Questo dà il colore desiderato a tutte le viste corrispondenti alla barra degli strumenti (pulsante Indietro, icone Menu e SearchView).

Leggi SearchView online: https://riptutorial.com/it/android/topic/4786/searchview

Capitolo 215: Secure SharedPreferences

introduzione

Le preferenze condivise sono **file XML basati su valori-chiave** . Si trova in `/data/data/package_name/shared_prefs/<filename.xml>` .

Quindi un utente con privilegi di root può navigare in questa posizione e può cambiarne i valori. Se si desidera proteggere i valori nelle preferenze condivise, è possibile scrivere un semplice meccanismo di crittografia e decrittografia.

Dovresti sapere che le preferenze condivise non sono mai state create per essere sicure, è solo un modo semplice per conservare i dati.

Sintassi

1. public static String encrypt (String input);
2. public static String decrypt (String input);

Parametri

Parametro	Definizione
ingresso	Valore stringa da crittografare o decrittografare.

Osservazioni

Le preferenze condivise non sono mai state create per essere sicure, è solo un modo semplice per mantenere i dati.

Non è una buona idea utilizzare le preferenze condivise per archiviare informazioni importanti come le credenziali dell'utente. Per salvare le credenziali dell'utente (come le password) è necessario utilizzare altri metodi come `AccountManager` di Android.

Examples

Protezione di una preferenza condivisa

Semplice codec

Qui per illustrare il principio di funzionamento possiamo usare la crittografia e la decifrazione semplici come segue.

```
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));
}
```

Tecnica di implementazione

```
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);
```

Leggi Secure SharedPreferences online: https://riptutorial.com/it/android/topic/9887/secure-sharedpreferences

Capitolo 216: Secure SharedPreferences

introduzione

Le preferenze condivise sono **file XML basati su valori-chiave** . Si trova in / data / data / nome_pacchetto / shared_prefs / <nomefile.xml>.

Quindi un utente con privilegi di root può navigare in questa posizione e può cambiarne i valori. Se si desidera proteggere i valori nelle preferenze condivise, è possibile scrivere un semplice meccanismo di crittografia e decrittografia.

Dovresti sapere che le preferenze condivise non sono mai state create per essere sicure, è solo un modo semplice per conservare i dati.

Sintassi

1. public static String encrypt (String input);
2. public static String decrypt (String input);

Parametri

Parametro	Definizione
ingresso	Valore stringa da crittografare o decrittografare.

Osservazioni

Le preferenze condivise non sono mai state create per essere sicure, è solo un modo semplice per mantenere i dati.

Non è una buona idea utilizzare le preferenze condivise per archiviare informazioni importanti come le credenziali dell'utente. Per salvare le credenziali dell'utente (come le password) è necessario utilizzare altri metodi come `AccountManager` di Android.

Examples

Protezione di una preferenza condivisa

Semplice codec

Qui per illustrare il principio di funzionamento possiamo usare la crittografia e la decifrazione semplici come segue.

```
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));
}
```

Tecnica di implementazione

```
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);
```

Leggi Secure SharedPreferences online: https://riptutorial.com/it/android/topic/9890/secure-sharedpreferences

Capitolo 217: SensorManager

Examples

Recupero degli eventi del sensore

Recupero delle informazioni del sensore dai sensori di 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
    }
}
```

Trasformazione del sensore al sistema di coordinate del mondo

I valori dei sensori restituiti da Android sono relativi al sistema di coordinate del telefono (ad es. + Punti Y verso la parte superiore del telefono). Possiamo trasformare questi valori di sensore in un sistema di coordinate del mondo (ad es. + Punti Y verso nord magnetico, tangenti al suolo) usando la matrice di rotazione dei gestori dei sensori

Innanzitutto, è necessario dichiarare e inizializzare le matrici / array in cui verranno archiviati i dati (ad esempio, è possibile farlo nel metodo `onCreate`):

```
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];
```

Successivamente, abbiamo bisogno di rilevare i cambiamenti nei valori dei sensori, memorizzarli negli array corrispondenti (se vogliamo usarli successivamente / altrove), quindi calcolare la matrice di rotazione e la trasformazione risultante in coordinate globali:

```
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];

}
```

Decidi se il tuo dispositivo è statico o meno, usando l'accelerometro

Aggiungi il seguente codice 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);
```

Potrebbe essere necessario regolare la sensibilità adattando il `movementThreshold` Threshold per tentativi ed errori. Quindi, sovrascrivere il metodo `onSensorChanged()` come segue:

```
@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);
    }
}
```

Se vuoi impedire che la tua app venga installata su dispositivi che non dispongono di un accelerometro, devi aggiungere la seguente riga al manifest:

```
<uses-feature android:name="android.hardware.sensor.accelerometer" />
```

Leggi SensorManager online: https://riptutorial.com/it/android/topic/3344/sensormanager

Capitolo 218: Servizio

introduzione

Un servizio viene eseguito in **background** per eseguire operazioni di lunga durata o per eseguire lavori per processi remoti. Un servizio non fornisce alcuna interfaccia utente che viene eseguita solo in background con l'input dell'utente. Ad esempio, un servizio può riprodurre musica in sottofondo mentre l'utente si trova in un'app diversa, oppure potrebbe scaricare dati da Internet senza bloccare l'interazione dell'utente con il dispositivo Android.

Osservazioni

Se non hai definito il tuo servizio nel tuo AndroidManifest.xml, riceverai una `ServiceNotFoundException` quando tenti di avviarlo.

Nota:

Per informazioni su `IntentService` , vedere qui: Esempio di IntentService

Examples

Avvio di un servizio

Avviare un servizio è molto semplice, basta chiamare `startService` con un intento, all'interno di un'attività:

```
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 di vita di un servizio

Il ciclo di vita dei servizi presenta le seguenti richiamate

- `onCreate()` :

Eseguito quando il servizio viene creato per configurare le configurazioni iniziali che potrebbero essere necessarie. Questo metodo viene eseguito solo se il servizio non è già in esecuzione.

- `onStartCommand()` :

Eseguito ogni volta che `startService()` viene invocato da un altro componente, come un'attività o un BroadcastReceiver. Quando si utilizza questo metodo, il servizio verrà eseguito finché non si chiama `stopSelf()` o `stopService()` . Si noti che indipendentemente da quante volte si chiama

`onStartCommand()` , i metodi `stopSelf()` e `stopService()` devono essere richiamati una sola volta per arrestare il servizio.

- `onBind()` :

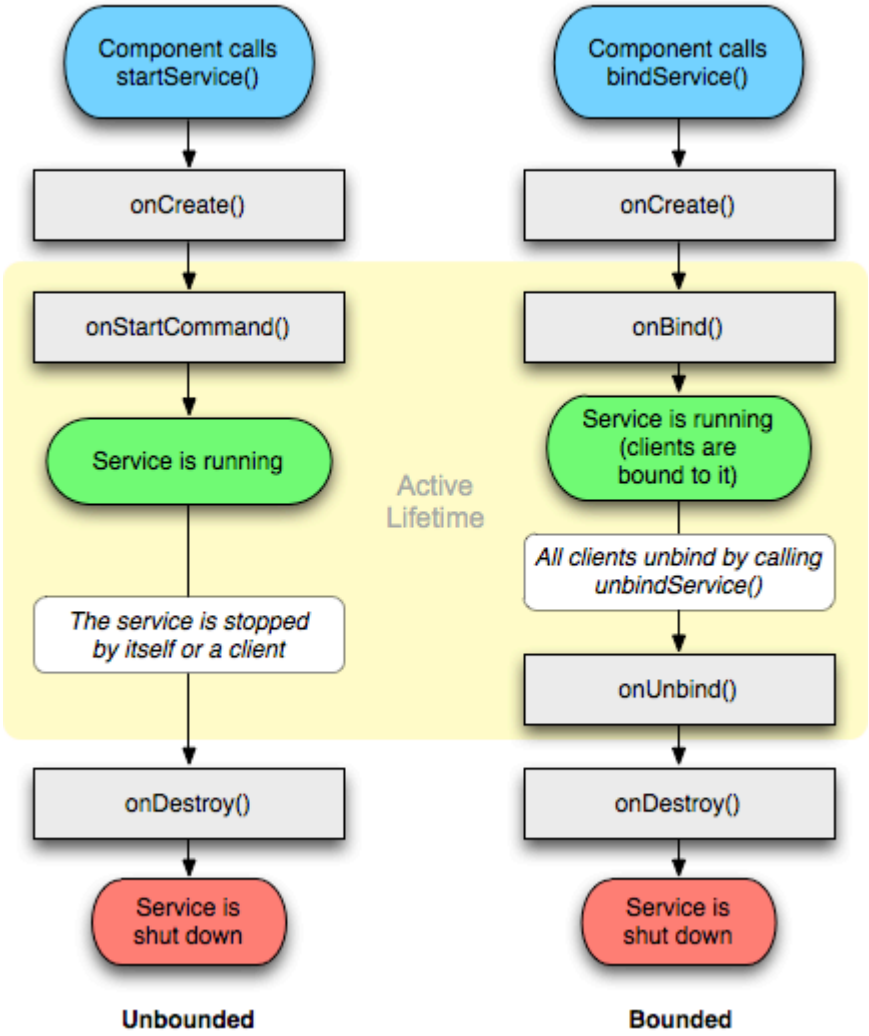
Eseguito quando un componente chiama `bindService()` e restituisce un'istanza di IBInder, fornendo un canale di comunicazione al Servizio. Una chiamata a `bindService()` manterrà il servizio in esecuzione finché ci sono client ad esso associati.

- `onDestroy()` :

Eseguito quando il servizio non è più in uso e consente lo smaltimento delle risorse assegnate.

È importante notare che durante il ciclo di vita di un servizio potrebbero essere richiamati altri callback come `onConfigurationChanged()` e `onLowMemory()`

https://developer.android.com/guide/components/services.html



Definire il processo di un servizio

Il campo `android:process` definisce il nome del processo in cui deve essere eseguito il servizio. Normalmente, tutti i componenti di un'applicazione vengono eseguiti nel processo predefinito creato per l'applicazione. Tuttavia, un componente può sovrascrivere il valore predefinito con il

proprio attributo di processo, consentendo di distribuire l'applicazione su più processi.

Se il nome assegnato a questo attributo inizia con due punti (':'), il servizio verrà eseguito nel suo processo separato.

```
<service
  android:name="com.example.appName"
  android:process=":externalProcess" />
```

Se il nome del processo inizia con un carattere minuscolo, il servizio verrà eseguito in un processo globale con quel nome, a condizione che abbia il permesso di farlo. Ciò consente ai componenti di diverse applicazioni di condividere un processo, riducendo l'utilizzo delle risorse.

Creazione del servizio associato con l'aiuto di Binder

Crea una classe che estende la classe di `Service` e in metodo sovrascritto `onBind` restituisce l'istanza di binder locale:

```
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;
    }
}
```

Quindi, durante l'attività, `onStart` binding al servizio in callback di `onStart` , utilizzando l'istanza `ServiceConnection` e separandoti da esso in `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;
        }
    };
}
```

Creazione di un servizio remoto (tramite AIDL)

Descrivi la tua interfaccia di accesso al servizio tramite il file `.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();
}
```

Ora dopo l'applicazione di build, gli strumenti sdk `.java` file `.java` appropriato. Questo file conterrà la classe `Stub` che implementa la nostra interfaccia aidl e che dobbiamo estendere:

```
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;
    }
}
```

Quindi in attività:

```
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;
    }
}
```

Creazione di un servizio non associato

La prima cosa da fare è aggiungere il servizio a `AndroidManifest.xml` , all'interno del tag `<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>
```

Se si intende gestire la classe di servizio in un pacchetto separato (ad esempio:. AllServices.RecordingService), sarà necessario specificare dove si trova il servizio. Quindi, nel caso di cui sopra modificheremo:

```
android:name=".RecordingService"
```

a

```
android:name=".AllServices.RecordingService"
```

o il modo più semplice per farlo è specificare il nome completo del pacchetto.

Quindi creiamo la classe di servizio effettiva:

```
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();
    }

}
```

Tutto questo servizio viene visualizzato quando è in esecuzione e può visualizzare toasts quando viene chiamato il metodo `doSomething()` .

Come si noterà, è implementato come un singleton , tenendo traccia della propria istanza, ma senza il solito metodo statico di fabbrica singleton perché i servizi sono naturalmente singleton e sono creati da intenti. L'istanza è utile all'esterno per ottenere un "handle" per il servizio quando è in esecuzione.

Infine, dobbiamo avviare e interrompere il servizio da un'attività:

```
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);
    }
}
```

In questo esempio, il servizio viene avviato e interrotto con lo stesso metodo, a seconda del suo stato corrente.

Possiamo anche invocare il metodo `doSomething()` dalla nostra attività:

```
public void makeServiceDoSomething(){
    if( RecordingService.isRunning )
        RecordingService.instance.doSomething();
}
```

Leggi Servizio online: https://riptutorial.com/it/android/topic/137/servizio

Capitolo 219: SharedPreferences

introduzione

SharedPreferences fornisce un modo per salvare i dati su disco sotto forma di coppie **chiave-valore** .

Sintassi

- **Metodo di contesto**
  - public SharedPreferences getSharedPreferences (Nome stringa, modalità int)
- **Metodo di attività**
  - public SharedPreferences getPreferences ()
- **Metodi SharedPreferences**
  - public SharedPreferences.Editor edit ()
  - contiene booleano pubblico ()
  - Mappa pubblica <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)
  - public Set getStringSet (String key, Set defValues)
  - public void registerOnSharedPreferenceChangeListener (SharedPreferences.OnSharedPreferenceChangeListener listener)
  - public void unregisterOnSharedPreferenceChangeListener (SharedPreferences.OnSharedPreferenceChangeListener listener)
- **SharedPreferences.Editor Methods**
  - public public apply ()
  - commit booleano pubblico ()
  - public SharedPreferences.Editor clear ()
  - public SharedPreferences.Editor putBoolean (Chiave stringa, valore booleano)
  - public SharedPreferences.Editor putFloat (Chiave stringa, valore float)
  - public SharedPreferences.Editor putInt (Chiave stringa, valore int)
  - public SharedPreferences.Editor putLong (Chiave stringa, valore lungo)
  - public SharedPreferences.Editor putString (Chiave stringa, Valore stringa)
  - public SharedPreferences.Editor putStringSet (Chiave stringa, Imposta valori)
  - public SharedPreferences.Editor remove (Chiave stringa)

Parametri

Parametro	Dettagli
chiave	`String` non nulla che identifica il parametro. Può contenere spazi bianchi o non stampabili. Questo è usato solo all'interno della tua app (e nel file XML), quindi non deve essere namespace, ma è una buona idea averlo come costante nel tuo codice sorgente. Non localizzarlo.
defValue	Tutte le funzioni get assumono un valore predefinito, che viene restituito se la chiave specificata non è presente in `SharedPreferences` . Non viene restituito se la chiave è presente ma il valore ha il tipo sbagliato: in tal caso si ottiene un `ClassCastException` .

Osservazioni

- `SharedPreferences` non deve essere utilizzato per l'archiviazione di grandi quantità di dati. Per tali scopi, è molto meglio usare `SQLiteDatabase` .
- `SharedPreferences` sono solo processo singolo, a meno che non si usi la modalità deprecata `MODE_MULTI_PROCESS` . Pertanto, se la tua app ha più processi, non sarai in grado di leggere le `SharedPreferences` del processo principale in un altro processo. In questi casi, è necessario utilizzare un altro meccanismo per condividere i dati tra i processi, ma non utilizzare `MODE_MULTI_PROCESS` poiché non è affidabile e deprecato.
- È preferibile utilizzare `SharedPreferences` istanza `SharedPreferences` nella classe `Singleton` per accedere a tutto il `context` dell'applicazione. Se vuoi usarlo solo per Attività particolari vai su `getPreferences()` .
- Evita di memorizzare informazioni sensibili in testo in chiaro mentre usi `SharedPreferences` poiché può essere letto facilmente.

Documentazione ufficiale

https://developer.android.com/reference/android/content/SharedPreferences.html

Examples

Leggere e scrivere valori su 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()` è un metodo della classe `Context` , che `Activity` estende. Se avete bisogno di accedere alle `getSharedPreferences()` metodo dal altre classi, è possibile utilizzare `context.getSharedPreferences()` con un `Context` di riferimento oggetto da un `Activity` , `View` , o `Application` .

Rimozione delle chiavi

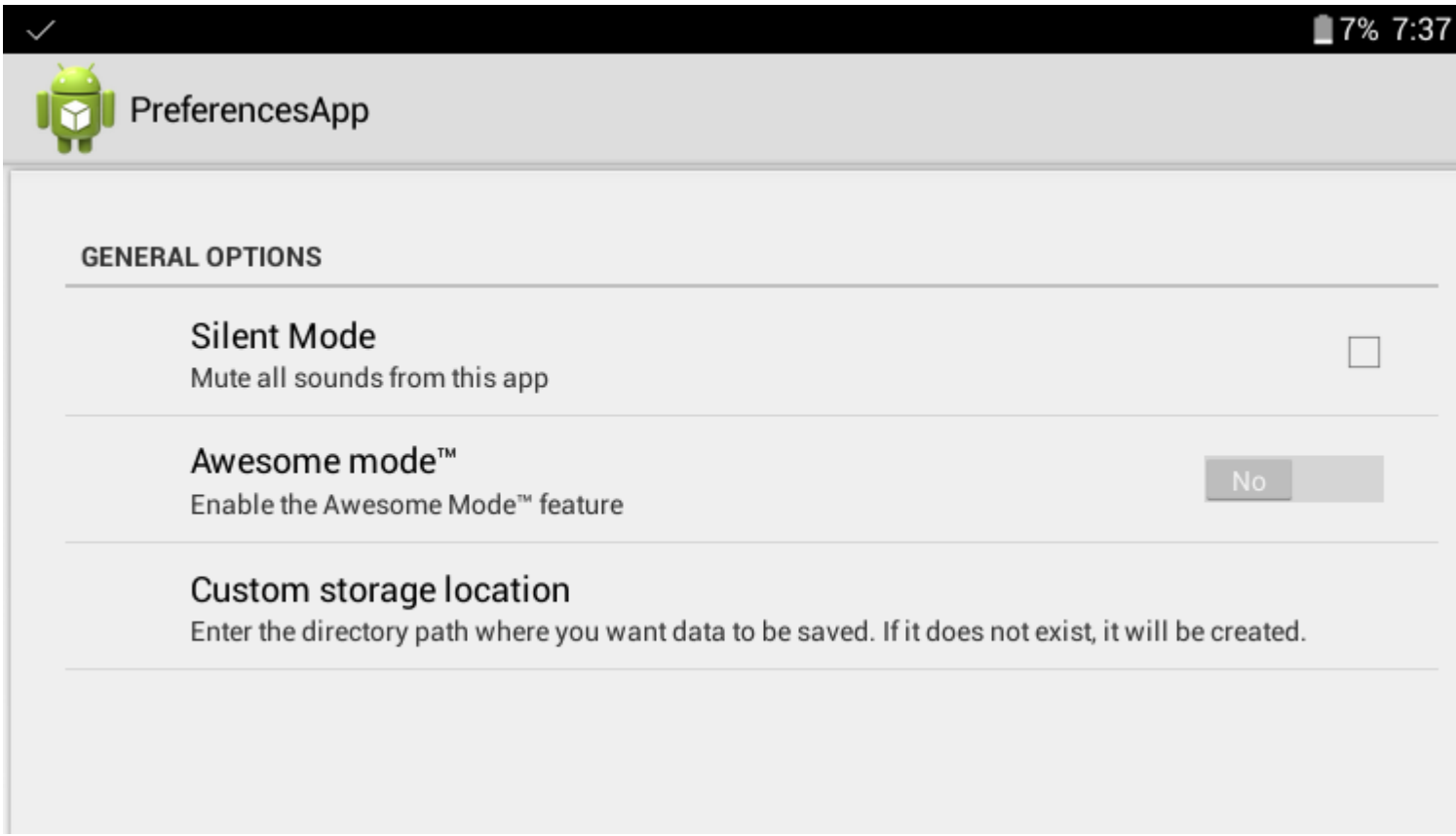
```
private static final String MY_PREF = "MyPref";

// ...

SharedPreferences prefs = ...;
```

```
// ...

SharedPreferences.Editor editor = prefs.edit();
editor.putString(MY_PREF, "value");
editor.remove(MY_PREF);
editor.apply();
```

Dopo `apply()` , `prefs` contiene "chiave" -> "valore", oltre a qualsiasi cosa contenesse già. Anche se sembra che ho aggiunto "chiave" e poi rimosso, la rimozione avviene effettivamente prima. Le modifiche `Editor` vengono applicate tutte in una volta, non nell'ordine in cui sono state aggiunte. Tutte le rimosse accadono prima di tutte le put.

Implementazione di una schermata delle impostazioni usando SharedPreferences

Un uso di `SharedPreferences` consiste nell'implementare una schermata "Impostazioni" nella tua app, in cui l'utente può impostare le proprie preferenze / opzioni. Come questo:



A PreferenceScreen salva le preferenze dell'utente in `SharedPreferences` . Per creare un PreferenceScreen, hai bisogno di alcune cose:

Un file XML per definire le opzioni disponibili:

Questo va in `/res/xml/preferences.xml` , e per la schermata delle impostazioni sopra, appare come questo:

```
<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>
```

Definisce le opzioni disponibili nella schermata delle impostazioni. Esistono molti altri tipi di `Preference` elencati nella documentazione degli sviluppatori Android sulla classe di preferenza .

Successivamente, abbiamo bisogno di **un'attività per ospitare la nostra** interfaccia utente delle **preferenze** . In questo caso, è piuttosto breve e assomiglia a questo:

```
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);
    }
}
```

Estende `PreferenceActivity` e fornisce l'interfaccia utente per la schermata delle preferenze. Può essere avviato come un'attività normale, in questo caso, con qualcosa di simile:

```
Intent i = new Intent(this, PreferencesActivity.class);
startActivity(i);
```

Non dimenticare di aggiungere `PreferencesActivity` al tuo `AndroidManifest.xml` .

Ottenere i valori delle preferenze all'interno della tua app è abbastanza semplice, basta chiamare `setDefaultValues()` per impostare i valori predefiniti definiti nel tuo XML e quindi ottenere le `SharedPreferences` predefinite. Un esempio:

```
//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", "");
```

Recupera tutte le voci memorizzate da un particolare file SharedPreferences

Il metodo `getAll()` recupera tutti i valori dalle preferenze. Possiamo usarlo, ad esempio, per registrare il contenuto corrente di `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 documentazione ti avverte della modifica della `Collection` restituita da `getAll` :

> Si noti che non è necessario modificare la raccolta restituita da questo metodo o alterarne il contenuto. La coerenza dei dati memorizzati non è garantita se lo fai.

Ascolto delle modifiche di SharedPreferences

```
SharedPreferences sharedPreferences = ...;
sharedPreferences.registerOnSharedPreferenceChangeListener(mOnSharedPreferenceChangeListener);


private final SharedPreferences.OnSharedPreferenceChangeListener
mOnSharedPreferenceChangeListener = new SharedPreferences.OnSharedPreferenceChangeListener() {
    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
        //TODO
    }
}
```

Notare che:

- Il listener scatterà solo se il valore è stato aggiunto o modificato, l'impostazione dello stesso valore non la chiamerà;
- Il listener deve essere salvato in una variabile membro e **NON** con una classe anonima, perché `registerOnSharedPreferenceChangeListener` memorizza con un riferimento debole, quindi sarebbe garbage collection;

- Invece di utilizzare una variabile membro, può anche essere implementata direttamente dalla classe e quindi chiamare `registerOnSharedPreferenceChangeListener(this);`
- Ricordarsi di annullare la registrazione del listener quando non è più necessario utilizzare `unregisterOnSharedPreferenceChangeListener` .

Lettura e scrittura di dati su SharedPreferences con Singleton

Classe SharedPreferences Manager (Singleton) per leggere e scrivere tutti i tipi di dati.

```
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` che viene implementata dalle classi che vanno a `Gson` per evitare l'offuscamento proguard.

```
public interface Model {

}
```

Regole Proguard per l'interfaccia `Model` :

```
-keep interface com.example.app.Model
-keep class * implements com.example.app.Model { *;}
```

Diversi modi di creare un'istanza di un oggetto di SharedPreferences

È possibile accedere a SharedPreferences in diversi modi:

Ottieni il file SharedPreferences predefinito:

```
import android.preference.PreferenceManager;
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
```

Ottieni un file SharedPreferences specifico:

```
public static final String PREF_FILE_NAME = "PrefFile";
SharedPreferences prefs = getSharedPreferences(PREF_FILE_NAME, MODE_PRIVATE);
```

Ottieni SharedPreferences da un'altra app:

```
// 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)`

restituisce le preferenze salvate dal `Activity's class name` come descritto nei documenti :

> Recupera un oggetto SharedPreferences per accedere alle preferenze private di questa attività. Questo semplicemente chiama il metodo getSharedPreferences (String, int) sottostante passando il nome della classe di questa attività come nome delle preferenze.

Durante l'utilizzo del metodo getSharedPreferences (nome String, modalità int) vengono restituiti i prefs salvati con il `name` . Come nei documenti:

> Recupera e mantieni il contenuto del file delle preferenze 'nome', restituendo un oggetto SharedPreferences attraverso il quale è possibile recuperare e modificare i suoi valori.

Pertanto, se il valore da salvare in `SharedPreferences` deve essere utilizzato nell'app, è necessario utilizzare `getSharedPreferences (String name, int mode)` con un nome fisso. Come, usando `getPreferences(int)` ritorna / salva le preferenze che appartengono `Activity` chiamandola.

Commit vs. Applica

Il metodo `editor.apply()` è **asincrono** , mentre `editor.commit()` è **sincrono** .

Ovviamente, dovresti chiamare `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()` stato aggiunto in 2.3 (API 9), esegue il commit senza restituire un valore booleano che indica il successo o il fallimento.

`commit()` restituisce true se il salvataggio funziona, false altrimenti.

`apply()` stato aggiunto mentre il team di sviluppo Android ha notato che quasi nessuno ha preso nota del valore restituito, quindi applicare è più veloce in quanto è asincrono.

A differenza di `commit()` , che scrive le sue preferenze in memoria permanente in modo sincrono, `apply()` `SharedPreferences` immediatamente le sue modifiche alle `SharedPreferences` memoria ma

avvia un commit asincrono su disco e non viene notificato alcun errore. Se un altro editor su questo `SharedPreferences` esegue un `commit()` regolare mentre un `apply()` è ancora in sospeso, il `commit()` si bloccherà fino a quando non saranno completati tutti i commit asincroni (apply) e tutti gli altri commit di sincronizzazione che potrebbero essere in sospeso.

Tipi di dati supportati in SharedPreferences

`SharedPreferences` consente di memorizzare solo tipi di dati primitivi (`boolean` , `float` , `long` , `int` , `String` e `string set`). Non è possibile archiviare oggetti più complessi in `SharedPreferences` e, in quanto tale, è pensato per essere un luogo in cui archiviare le impostazioni dell'utente o simili, non è pensato per essere un database per conservare i dati dell'utente (ad esempio il salvataggio di un elenco di attività effettuato dall'utente, ad esempio).

Per memorizzare qualcosa in `SharedPreferences` usi una chiave e un valore. La chiave è come puoi fare riferimento a ciò che hai archiviato in seguito e ai dati del valore che desideri memorizzare.

```
    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();
```

Archivia, recupera, rimuove e cancella i dati da SharedPreferences

Crea SharedPreferences BuyyaPref

```
SharedPreferences pref = getApplicationContext().getSharedPreferences("BuyyaPref",
MODE_PRIVATE);
Editor editor = pref.edit();
```

Memorizzazione dei dati come coppia 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
```

Ottieni dati SharedPreferences

Se il valore per la chiave non esiste, restituisce il valore del secondo parametro (in questo caso null, questo è come il valore predefinito)

```
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
```

Eliminazione del valore chiave da 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
```

Cancella tutti i dati da SharedPreferences

```
 editor.clear();
 editor.commit(); // commit changes
```

Supporta pre-Honeycomb con StringSet

Ecco la classe di utilità:

```
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 esempio per salvare le preferenze come tipo di dati StringSet è:

```
Set<String> sets = new HashSet<>();
sets.add("John");
sets.add("Nicko");
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
SharedPreferencesCompat.putStringSet(preferences.edit(), "pref_people", sets);
```

Per recuperarli di nuovo:

```
Set<String> people = SharedPreferencesCompat.getStringSet(preferences, "pref_people", new
HashSet<String>());
```

Riferimento: preferenza di supporto Android

Aggiungi filtro per EditTextPreference

Crea questa classe:

```
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;
    }
}
```

Uso :

```
EditText compressPic = ((EditTextPreference)
findPreference(getString("pref_key_compress_pic"))).getEditText();
compressPic.setFilters(new InputFilter[]{ new InputFilterMinMax(1, 100) });
```

Leggi SharedPreferences online: https://riptutorial.com/it/android/topic/119/sharedpreferences

Capitolo 220: ShortcutManager

Examples

Scorciatoie di avvio dinamico

```
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));
```

Possiamo rimuovere facilmente tutte le scorciatoie dinamiche chiamando: -

```
 shortcutManager.removeAllDynamicShortcuts();
```

Possiamo aggiornare gli Shorcuts dinamici esistenti usando

```
shortcutManager.updateShortcuts(Arrays.asList(shortcut);
```

Tieni presente che `setDynamicShortcuts(List)` viene utilizzato per ridefinire l'intero elenco di scorciatoie dinamiche, `addDynamicShortcuts(List)` viene utilizzato per aggiungere scorciatoie dinamiche all'elenco esistente di scorciatoie dinamiche

Leggi ShortcutManager online: https://riptutorial.com/it/android/topic/7661/shortcutmanager

Capitolo 221: Sicurezza

Examples

Verifica firma app - Rilevazione manomissione

Questa tecnica spiega come assicurarsi che il tuo .apk sia stato firmato con il certificato del tuo sviluppatore e sfrutta il fatto che il certificato rimane coerente e che solo tu hai accesso ad esso. Possiamo rompere questa tecnica in 3 semplici passaggi:

- Trova la firma del certificato dello sviluppatore.
- Incorpora la tua firma in una costante String nella tua app.
- Verifica che la firma in fase di runtime corrisponda alla firma dello sviluppatore incorporato.

Ecco lo snippet di codice:

```
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;

}
```

Leggi Sicurezza online: https://riptutorial.com/it/android/topic/4664/sicurezza

Capitolo 222: Sincronizzazione dei dati con l'adattatore di sincronizzazione

Examples

Dummy Sync Adapter con Stub Provider

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");
    }

}
```

Servizio di sincronizzazione

```
/**
 * 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();
    }
}
```

Autenticatore

```
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();
    }
}
```

Servizio Authenticator

```
/**
 * 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();
    }
}
```

Aggiunte AndroidManifest.xml

```
<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;
    }
}
```

Chiamare questa funzione su login riuscito per creare un account con l'ID utente registrato

```
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;
    }
```

Forzare una sincronizzazione

```
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);
```

Leggi Sincronizzazione dei dati con l'adattatore di sincronizzazione online: https://riptutorial.com/it/android/topic/1944/sincronizzazione-dei-dati-con-l-adattatore-di-sincronizzazione

Capitolo 223: Smart card

Examples

Smart card invia e ricevi

Per la connessione, ecco uno snippet per aiutarti a capire:

```
//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);
}
```

Ora devi capire che in Java la comunicazione avviene usando il pacchetto javax.smarcard che non è disponibile per Android, quindi dai un'occhiata qui per avere un'idea di come puoi comunicare o inviare / ricevere APDU (comando smartcard).

Ora come detto nella risposta di cui sopra

Non è possibile semplicemente inviare un APDU (comando smartcard) sull'endpoint bulk-out e prevedere di ricevere un APDU di risposta sull'endpoint bulk-in. Per ottenere gli endpoint, consulta lo snippet di codice riportato di seguito:

```
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;
                }
            }
```

```
        }
    }
```

Ora disponi degli endpoint bulk-in e bulk-out per inviare e ricevere comandi APDU e blocchi di risposta APDU:

Per l'invio di comandi, guarda il frammento di codice qui sotto:

```
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));
    }
}
```

E per ricevere / leggere una risposta, vedi il frammento di codice qui sotto:

```
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;
}
```

Ora se vedi questa risposta qui il primo comando da inviare è:

PC_to_RDR_IccPowerOn comando per attivare la scheda.

che è possibile creare leggendo la sezione 6.1.1 del documento Specifiche della classe dispositivo USB qui.

Ora facciamo un esempio di questo comando come quello qui: `62000000000000000000` Come puoi inviare questo è:

```
write(connection, epOut, "62000000000000000000");
```

Ora dopo aver inviato correttamente il comando APDU, puoi leggere la risposta usando:

```
read(connection, epIn);
```

E ricevere qualcosa come

```
80 18000000 00 00 00 00 00 3BBF11008131FE45455041000000000000000000000000F1
```

Ora la risposta ricevuta nel codice qui sarà nella variabile `result` del metodo `read()` dal codice

Leggi Smart card online: https://riptutorial.com/it/android/topic/10945/smart-card

Capitolo 224: Snack bar

Sintassi

- Snackbar make (Visualizza vista, testo CharSequence, durata int)
- Snackbar make (Visualizza vista, int resId, int duration)

Parametri

Parametro	Descrizione
vista	Visualizza: la vista per trovare un genitore da.
testo	CharSequence: il testo da mostrare. Può essere il testo formattato
resid	int: l'ID risorsa della risorsa stringa da utilizzare. Può essere il testo formattato
durata	int: Quanto tempo è necessario per visualizzare il messaggio. Questo può essere LENGTH_SHORT, LENGTH_LONG o LENGTH_INDEFINITE

Osservazioni

Snackbar fornisce un feedback leggero su un'operazione. Visualizza un breve messaggio nella parte inferiore dello schermo su dispositivo mobile e in basso a sinistra su dispositivi più grandi. Gli snack spuntano sopra tutti gli altri elementi sullo schermo e solo uno può essere visualizzato alla volta.

Scompaiono automaticamente dopo un timeout o dopo l'interazione dell'utente altrove sullo schermo, in particolare dopo le interazioni che evocano una nuova superficie o attività. Lo snack bar può essere cancellato dallo schermo.

Prima di utilizzare `SnackBar` è necessario aggiungere la dipendenza della libreria del supporto di progettazione nel file `build.gradle` :

```
dependencies {
    compile 'com.android.support:design:25.3.1'
}
```

Documentazione ufficiale

https://developer.android.com/reference/android/support/design/widget/Snackbar.html

Examples

Creare un semplice snack bar

La creazione di uno `Snackbar` può essere eseguita come segue:

```
Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG).show();
```

La `view` viene utilizzata per trovare un genitore adatto da utilizzare per visualizzare lo `Snackbar` . Normalmente questo sarebbe un `CoordinatorLayout` che hai definito nel tuo XML, che abilita l'aggiunta di funzionalità come swipe per chiudere e spostare automaticamente altri widget (ad es. `FloatingActionButton`). Se non c'è alcun `CoordinatorLayout` viene utilizzata la vista del contenuto della decorazione della finestra.

Molto spesso aggiungiamo anche un'azione allo `Snackbar` . Un caso di uso comune sarebbe un'azione "Annulla".

```
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();
```

Puoi creare uno `Snackbar` e mostrarlo in un secondo momento:

```
Snackbar snackbar = Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG);
snackbar.show();
```

Se vuoi cambiare il colore del testo di `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();
```

Per impostazione predefinita, `Snackbar` si chiude con il tasto destro del mouse. Questo esempio mostra come eliminare lo snackBar sul suo swipe sinistro .

Snack Bar personalizzato

Funzione per personalizzare lo snack bar

```
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;

    }
```

Chiama la funzione dal frammento o dall'attività

```
 SnackBar.makeText(MyActivity.this, "Please Locate your address at Map",
Snackbar.LENGTH_SHORT).show();
```

Snackbar con callback

Puoi utilizzare Snackbar.Callback per ascoltare se la barra degli snack è stata chiusa dall'utente o dal timeout.

```
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 personalizzato

Questo esempio mostra uno Snackbar bianco con l'icona di annullamento personalizzata.

```
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 toast: quale dovrei usare?

I toast sono generalmente utilizzati quando vogliamo mostrare all'utente informazioni relative ad alcune azioni che hanno avuto successo (o meno) e questa azione non richiede all'utente di intraprendere altre azioni. Come quando un messaggio è stato inviato, ad esempio:

```
Toast.makeText(this, "Message Sent!", Toast.LENGTH_SHORT).show();
```

Gli snack vengono anche utilizzati per visualizzare un'informazione. Ma questa volta, possiamo dare all'utente la possibilità di agire. Ad esempio, diciamo che l'utente ha cancellato un'immagine per errore e vuole recuperarlo. Possiamo fornire uno Snackbar con l'azione "Annulla". Come questo:

```
Snackbar.make(getCurrentFocus(), "Picture Deleted", Snackbar.LENGTH_SHORT)
        .setAction("Undo", new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Return his picture
            }
        })
        .show();
```

Conclusione: i toast vengono utilizzati quando non è necessaria l'interazione dell'utente. Gli snack vengono utilizzati per consentire agli utenti di eseguire un'altra azione o annullare una precedente.

Snackbar personalizzato (non serve visualizzare)

Creazione di uno snack senza la necessità di passare la vista a Snackbar, tutti i layout creano in Android in 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();
    }
}
```

per call class

nuovo CustomSnackBar (findViewById (CustomSnackBar.VIEW_PARENT), "message", CustomSnackBar.STATE_ERROR);

Leggi Snack bar online: https://riptutorial.com/it/android/topic/1500/snack-bar

Capitolo 225: SpannableString

Sintassi

- `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 (Object what)
- `void setSpan (Object what, int start, int end, int flags)`
- `CharSequence subSequence (int start, int end)`
- `String toString ()`
- `SpannableString valueOf (CharSequence source)`

Examples

Aggiungi stili a una TextView

Nell'esempio seguente, creiamo un'attività per visualizzare un singolo TextView.

TextView userà una `SpannableString` come suo contenuto, che illustrerà alcuni degli stili disponibili.

Qui 'cosa faremo con il testo:

- Ingrandilo
- Grassetto
- Sottolineare
- stampare in corsivo
- Strike-through
- Colorato
- evidenziato
- Mostra come apice
- Mostra come pedice
- Mostra come collegamento
- Rendilo cliccabile.

```
@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);

}
```

E il risultato sarà simile a questo:

Multi stringa, con multi colore

Metodo: **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`
```

Leggi SpannableString online: https://riptutorial.com/it/android/topic/10553/spannablestring

Capitolo 226: Spinner

Examples

Aggiungere uno spinner alla tua attività

In /res/values/strings.xml:

```
<string-array name="spinner_options">
    <item>Option 1</item>
    <item>Option 2</item>
    <item>Option 3</item>
</string-array>
```

Nel layout XML:

```
<Spinner
    android:id="@+id/spinnerName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_options" />
```

In attività:

```
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) {}
});
```

Esempio di spinner di base

Spinner È un tipo di input a discesa. Innanzitutto nel layout

```
  <Spinner
    android:id="@+id/spinner"     <!-- id to refer this spinner from JAVA-->
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

</Spinner>
```

Ora in secondo luogo popola i valori nella casella di selezione Esistono principalmente due modi per popolare i valori nella `spinner` .

1. Da XML stesso creare un **array.xml** nella directory dei **valori** sotto **res** . Crea questo `array`

```
  <string-array name="defaultValue">
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
</string-array>
```

Ora aggiungi questa riga in sppiner XML

```
            android:entries="@array/defaultValue"
```

2. Puoi anche aggiungere valori tramite JAVA

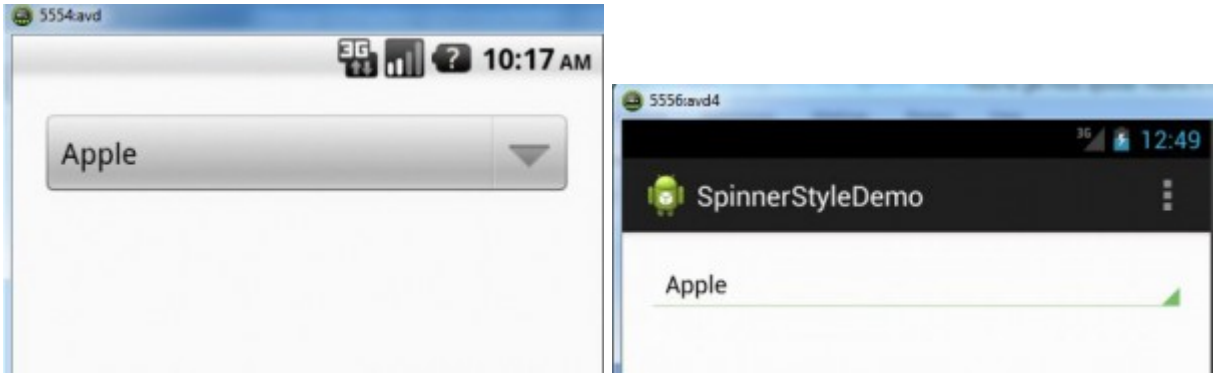
se stai usando in `activity` cityArea = (Spinner) findViewById (R.id.cityArea); altrimenti se stai usando in `fragment`

```
cityArea = (Spinner) findViewById(R.id.cityArea);
```

Ora crea un `arrayList` di `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));
```

Questo sarà simile



Secondo la versione Android del dispositivo renderà lo stile

Di seguito sono riportati alcuni dei temi predefiniti

Se un'app non richiede esplicitamente un tema nel file manifest, Android System determinerà il tema predefinito basato su targetSdkVersion dell'app per mantenere le aspettative originali dell'app:

Versione di Android SDK	Tema predefinito
Versione <11	@android: Stile / a tema
Versione tra 11 e 13	@android: Stile / Theme.Holo

Versione di Android SDK	Tema predefinito
14 e oltre	@android: Stile / Theme.DeviceDefault

`Spinner` può essere facilmente personalizzato con l'aiuto di xml es

```
 android:background="@drawable/spinner_background"

 android:layout_margin="16dp"

 android:padding="16dp"
```

Crea uno sfondo personalizzato in XML e usalo.

ottenere facilmente la posizione e altri dettagli dell'elemento selezionato nella casella di selezione

```
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 il colore del testo dell'elemento selezionato nella casella di selezione

Questo può essere fatto in due modi in XML

```
<item android:state_activated="true" android:color="@color/red"/>
```

Questo cambierà il colore dell'elemento selezionato nel popup.

e da JAVA fai questo (nel setOnItemSelectedListener (...))

```
 @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long
id) {
                 ((TextView) parent.getChildAt(0)).setTextColor(0x00000000);
// similarly change `background color` etc.
            }
```

Leggi Spinner online: https://riptutorial.com/it/android/topic/3459/spinner

Capitolo 227: SQLite

introduzione

SQLite è un sistema di gestione di database relazionale scritto in C. Per iniziare a lavorare con i database SQLite all'interno del framework Android, definire una classe che estenda SQLiteOpenHelper e personalizzare secondo necessità.

Osservazioni

La classe `SQLiteOpenHelper` definisce i `onCreate()` statici `onCreate()` e `onUpgrade()` . Questi metodi sono chiamati nei metodi corrispondenti di una sottoclasse `SQLiteOpenHelper` che personalizzi con le tue tabelle.

Examples

Utilizzando la classe 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;
            }
        }
    }
}
```

Inserisci i dati nel database

```
// 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
);
```

metodo onUpgrade ()

`SQLiteOpenHelper` è una classe helper per gestire la creazione del database e la gestione delle versioni.

In questa classe, il metodo `onUpgrade()` è responsabile dell'aggiornamento del database quando si apportano modifiche allo schema. Viene chiamato quando il file di database esiste già, ma la sua versione è inferiore a quella specificata nella versione corrente dell'app. Per ogni versione del database, è necessario applicare le modifiche specifiche apportate.

```
@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;
        }
    }
}
```

Lettura dei dati da un cursore

Ecco un esempio di un metodo che vivrebbe all'interno di una sottoclasse `SQLiteOpenHelper` . Utilizza la stringa `searchTerm` per filtrare i risultati, scorrere i contenuti del cursore e restituirli in un `List` di oggetti del `Product` .

Innanzitutto, definisci la classe POJO `Product` che sarà il contenitore per ogni riga recuperata dal database:

```
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;
  }
}
```

Quindi, definire il metodo che interrogherà il database e restituire un `List` di oggetti 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();
    }
}
```

Crea un contratto, un aiutante e un fornitore per SQLite in 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;
    }

}
```

Come usare:

```
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;
}
```

Aggiornamento di una riga in una tabella

```
// 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())
        }
);
```

Esecuzione di una transazione

Le transazioni possono essere utilizzate per apportare modifiche multiple al database atomicamente. Qualsiasi transazione normale segue questo schema:

```
// 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();
}
```

Chiamare `beginTransaction()` all'interno di una transazione attiva non ha alcun effetto.

Elimina riga (e) dalla tabella

Per eliminare tutte le righe dalla tabella

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();
```

```
db.delete(TABLE_NAME, null, null);
db.close();
```

Per eliminare tutte le righe dalla tabella e ottenere il conteggio della riga eliminata nel valore restituito

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

int numRowsDeleted = db.delete(TABLE_NAME, String.valueOf(1), null);
db.close();
```

Per cancellare le righe con la condizione `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();
```

Salva l'immagine in SQLite

Impostazione del database

```
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);
    }
}
```

Inserisci nel database:

```
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 );
}
```

Recupero dei dati :

```
 byte[] image = cursor.getBlob(1);
```

Nota:

1. Prima di inserirlo nel database, è necessario convertire prima l'immagine Bitmap in array di byte, quindi applicarla utilizzando la query del database.
2. Quando si recupera dal database, si dispone certamente di una matrice di byte di immagini, ciò che è necessario convertire l'array di byte nell'immagine originale. Quindi, devi usare BitmapFactory per decodificarlo.

Di seguito è riportata una classe di utilità che spero possa aiutarti:

```
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);
    }
}
```

Crea database dalla cartella delle risorse

Inserisci il tuo nome dbname.sqlite o dbname.db nella cartella delle risorse del tuo progetto.

```
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) {

        }
    }
```

Ecco come puoi accedere all'oggetto del database alla tua attività.

```
// Create Databasehelper class object in your activity.
   private Databasehelper db;
```

Quindi, nel metodo onCreate, inizializzalo e chiama il metodo createDatabase () come mostrato di seguito.

```
db = new Databasehelper(MainActivity.this);
        try {
            db.createDataBase();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

Esegui tutto il tuo inserimento, aggiorna, cancella e seleziona l'operazione come mostrato di seguito.

```
String query = "select Max(Id) as Id from " + TABLE_NAME;
        db.openDataBase();
        int count = db.getId(query);
        db.close();
```

Esportare e importare un database

Ad esempio, potresti voler importare ed esportare il tuo database per bacukups. Non dimenticare le autorizzazioni.

```
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 di massa

Ecco un esempio di inserimento di grandi blocchi di dati contemporaneamente. Tutti i dati che vuoi inserire sono raccolti all'interno di un array 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;
}
```

Ed ecco un esempio di come 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);
```

Leggi SQLite online: https://riptutorial.com/it/android/topic/871/sqlite

Capitolo 228: Strumenti Attributi

Osservazioni

Android ha uno spazio dei nomi XML dedicato destinato agli strumenti per poter registrare le informazioni nel file XML.

L'URI dello spazio dei nomi è:

`http://schemas.android.com/tools` e di solito è associato agli `tools:` prefisso.

Examples

Attributi di layout Designtime

Questi attributi vengono utilizzati quando il layout viene renderizzato in Android Studio, ma non hanno alcun impatto sul runtime.

In generale è possibile utilizzare qualsiasi attributo di framework Android, utilizzando solo gli `tools:` namespace piuttosto che `android:` spazio dei nomi per l'anteprima del layout. È possibile aggiungere sia l'attributo `android:` namespace (che viene utilizzato in fase di esecuzione) che gli `tools:` corrispondenza `tools:` attributo (che sovrascrive l'attributo di runtime solo nell'anteprima del layout).

Basta definire lo spazio dei nomi degli strumenti come descritto nella sezione commenti.

Ad esempio l'attributo di `text` :

```
<EditText
      tools:text="My Text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
```

Oppure l'attributo `visibility` per annullare l'impostazione di una vista per l'anteprima:

```
<LinearLayout
        android:id="@+id/ll1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:visibility="gone" />
```

Oppure l'attributo `context` per associare il layout con l'attività o il frammento

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity" >
```

Oppure l'attributo `showIn` per vedere e includere l'anteprima del layout in un altro layout

```
<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" />
```

Leggi Strumenti Attributi online: https://riptutorial.com/it/android/topic/1676/strumenti-attributi

Capitolo 229: Studio Android

Examples

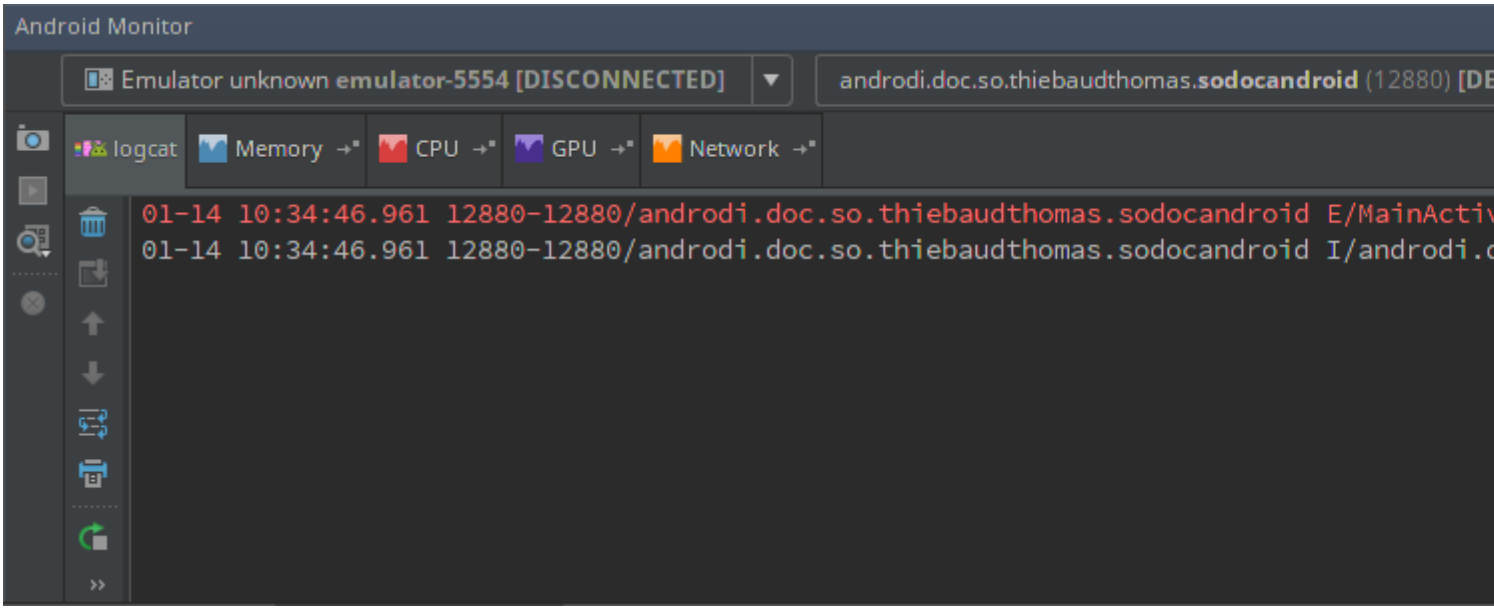
Filtra i registri dall'interfaccia utente

I log di Android possono essere filtrati direttamente dall'interfaccia utente. Usando questo codice

```
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");
    }
}
```

Se uso la regex `TAG1|TAG2` e il livello `verbose` che ottengo

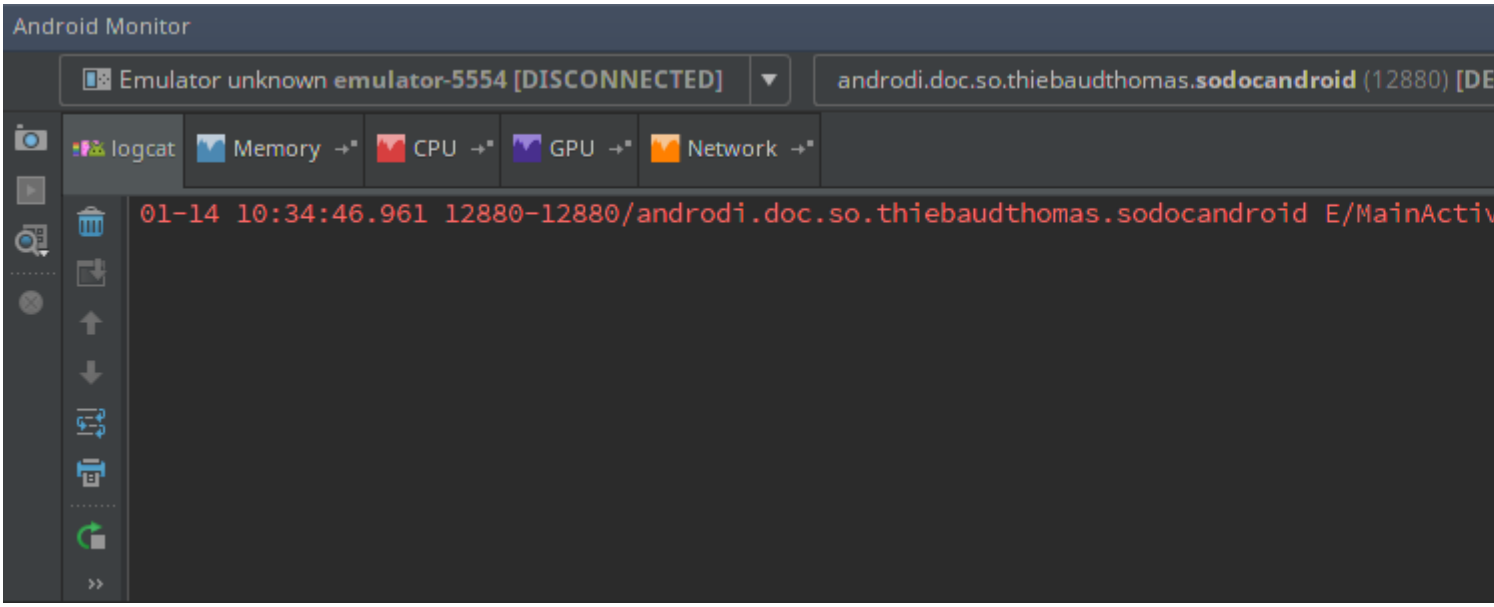
```
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log
from onCreate method with TAG1
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid
I/androdi.doc.so.thiebaudthomas.sodocandroid.MainActivity: Log from onCreate method with TAG2
```



Il livello può essere impostato per ottenere registri con un determinato livello e sopra. Ad esempio il livello `verbose` cattura i log `verbose, debug, info, warn, error and assert` .

Utilizzando lo stesso esempio, se si imposta il livello in `error` , ottengo solo

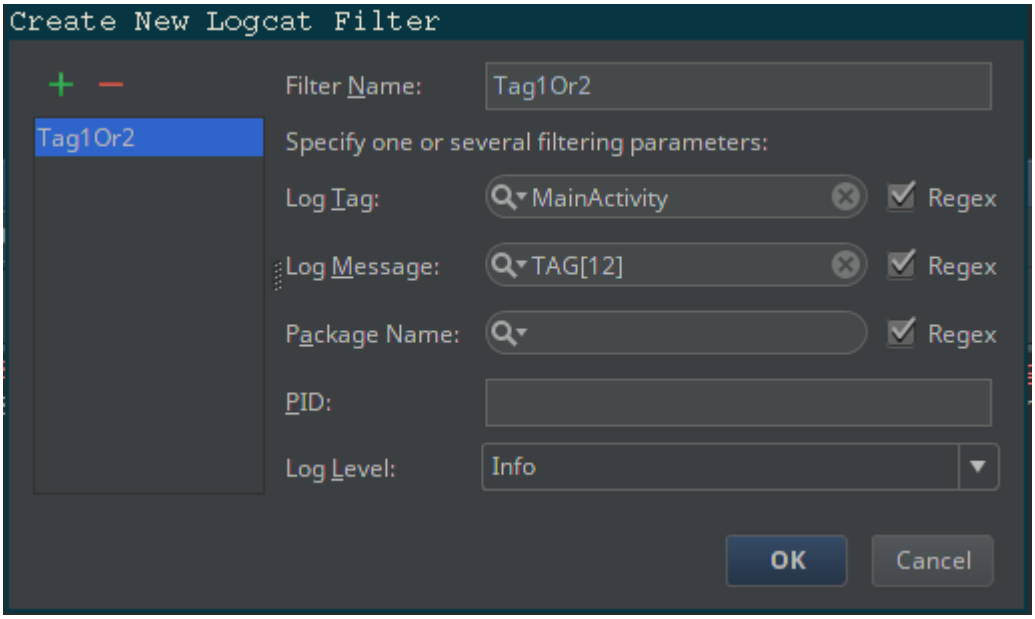
```
01-14 10:34:46.961 12880-12880/androdi.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log from onCreate method with TAG1
```



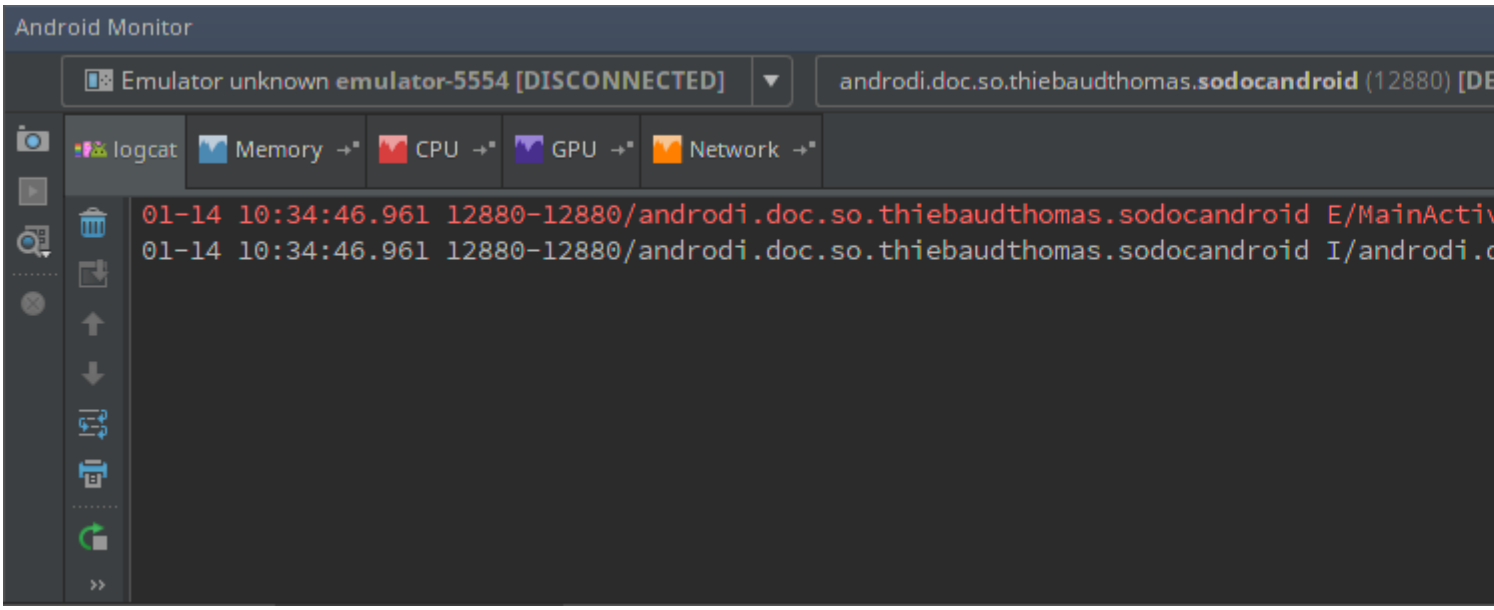
Crea la configurazione dei filtri

I filtri personalizzati possono essere impostati e salvati dall'interfaccia utente. Nella scheda `AndroidMonitor` , fai clic sul menu a discesa a destra (deve contenere `Show only selected application` o `No filters`) e seleziona `Edit filter configuration` .

Inserisci il filtro che desideri

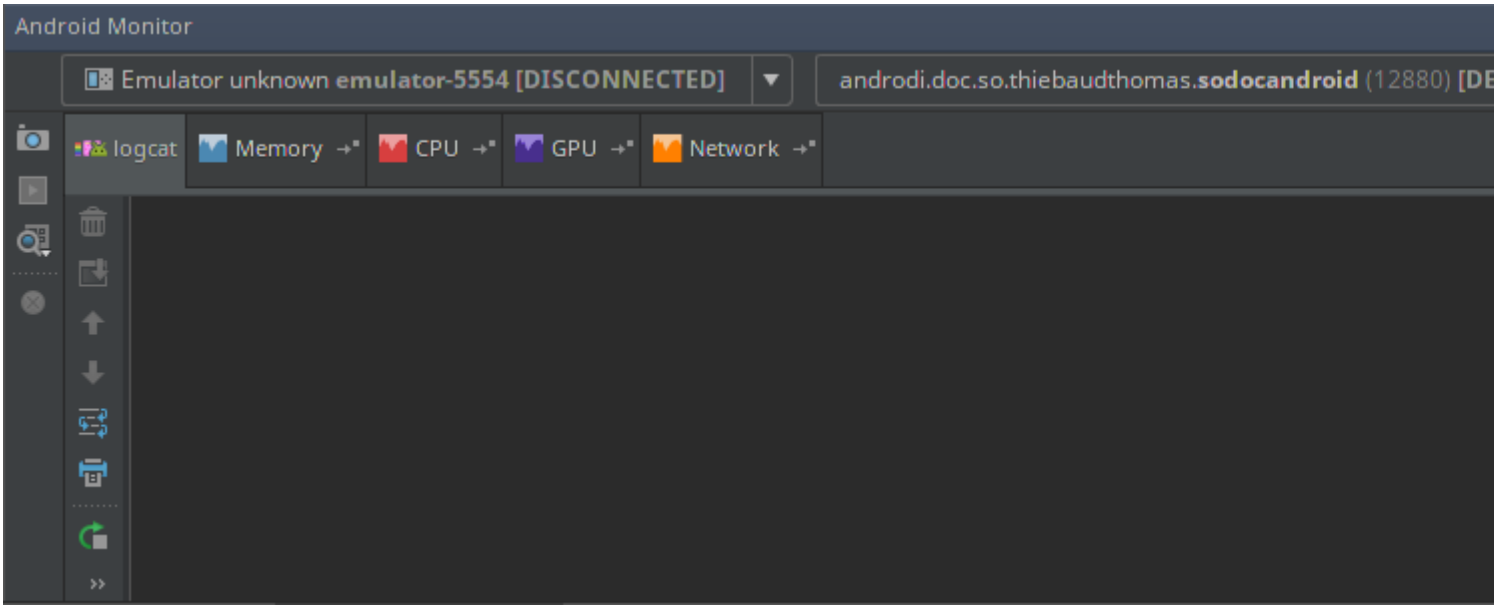


E usalo (puoi selezionarlo dallo stesso menu a discesa)

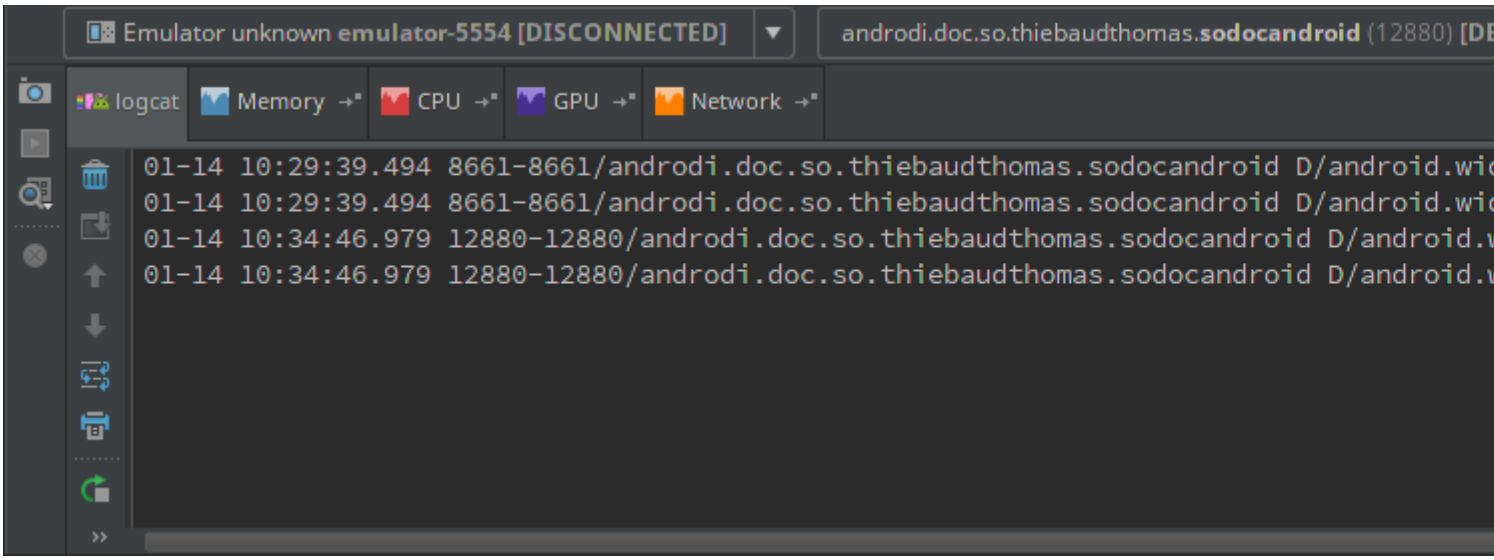


Importante Se aggiungi un input nella barra dei filtri, Android Studio considererà sia il tuo filtro che i tuoi input.

Con sia l'input che il filtro non c'è uscita



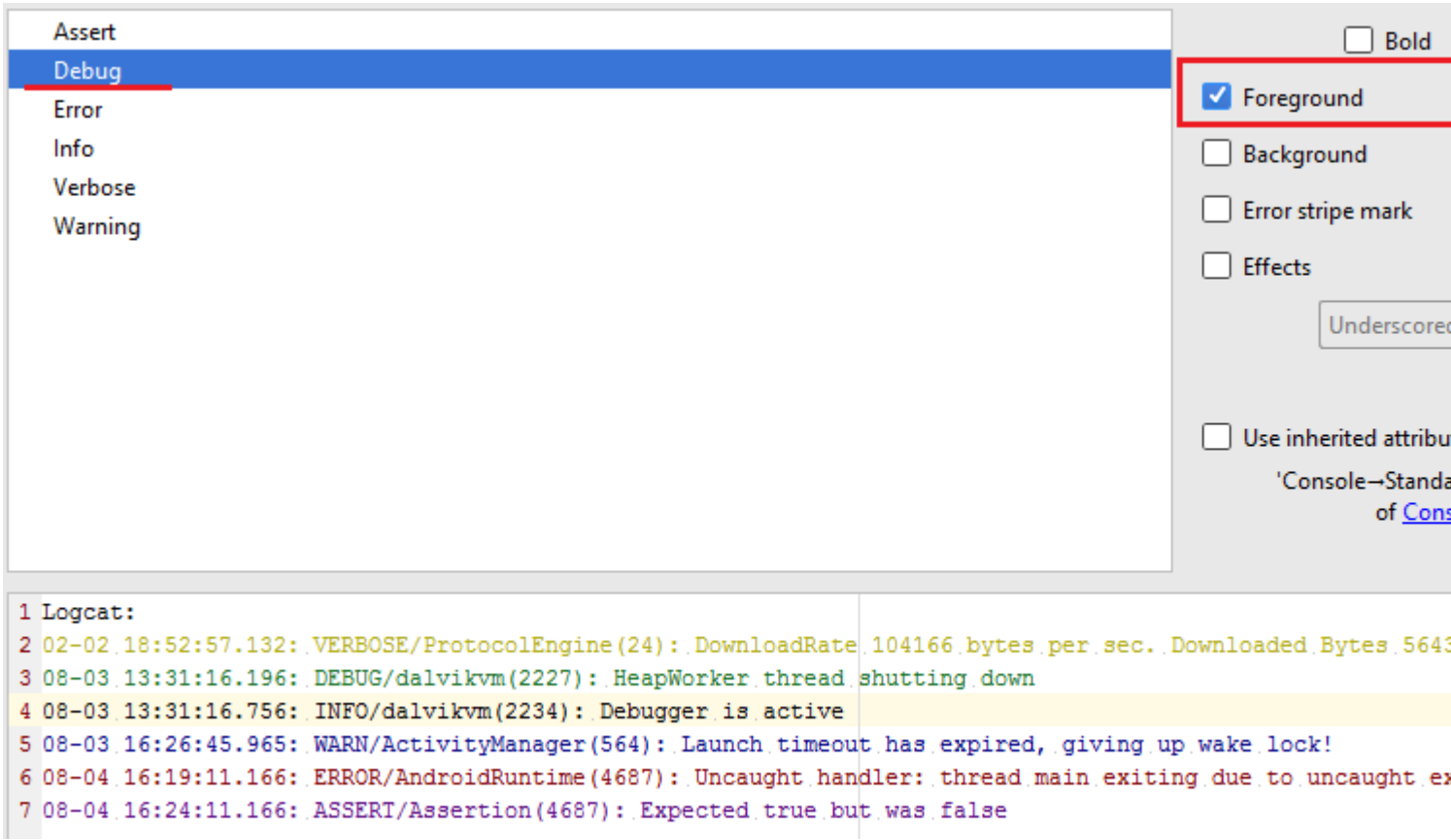
Senza filtro, ci sono alcune uscite



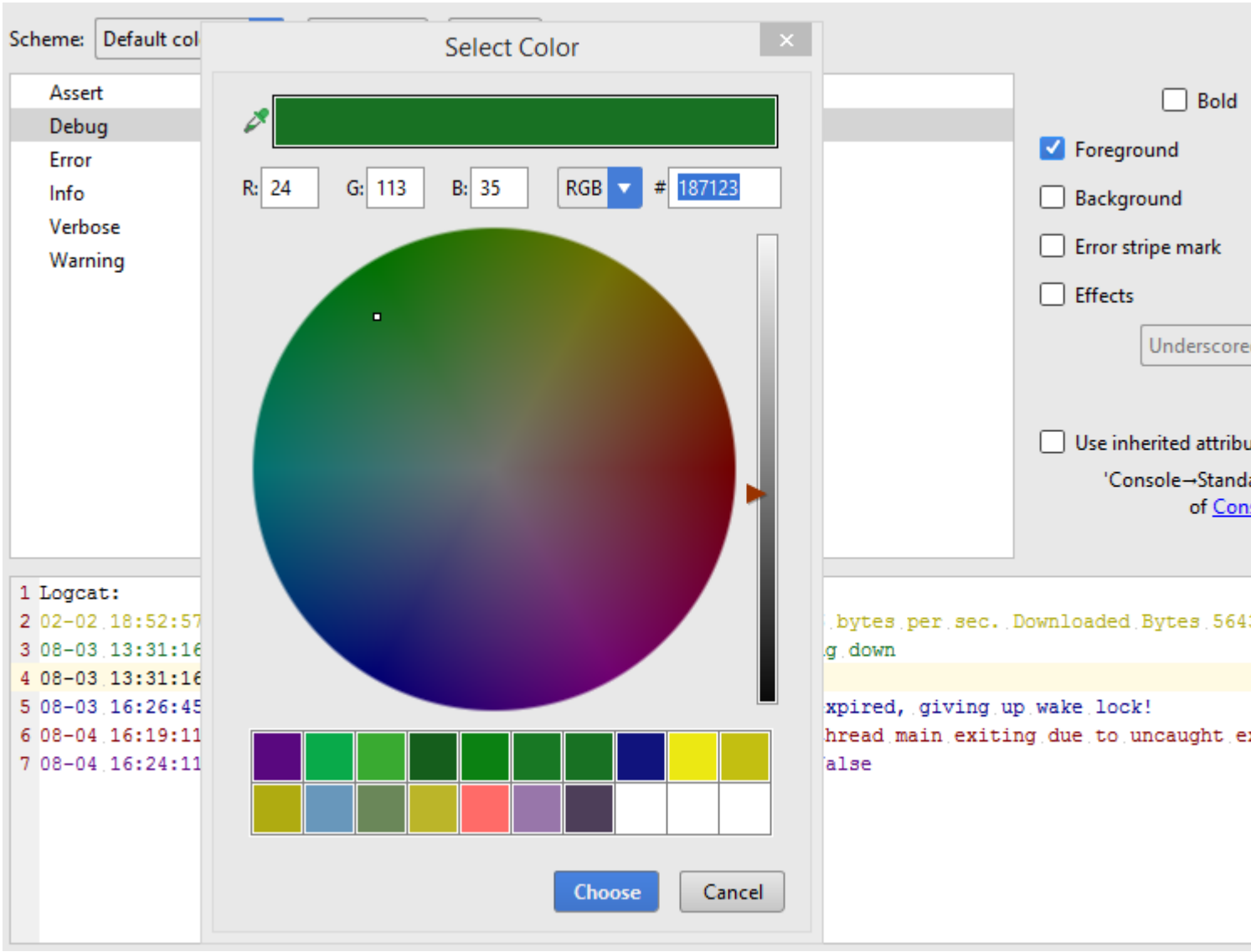
Colori personalizzati del messaggio logcat in base all'importanza del messaggio

Vai a File -> Impostazioni -> Editor -> Colori e tipi di carattere -> Logcat Android

Cambia i colori di cui hai bisogno:

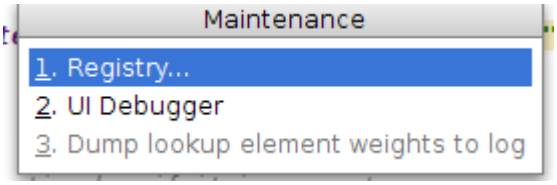


Scegli il colore appropriato:

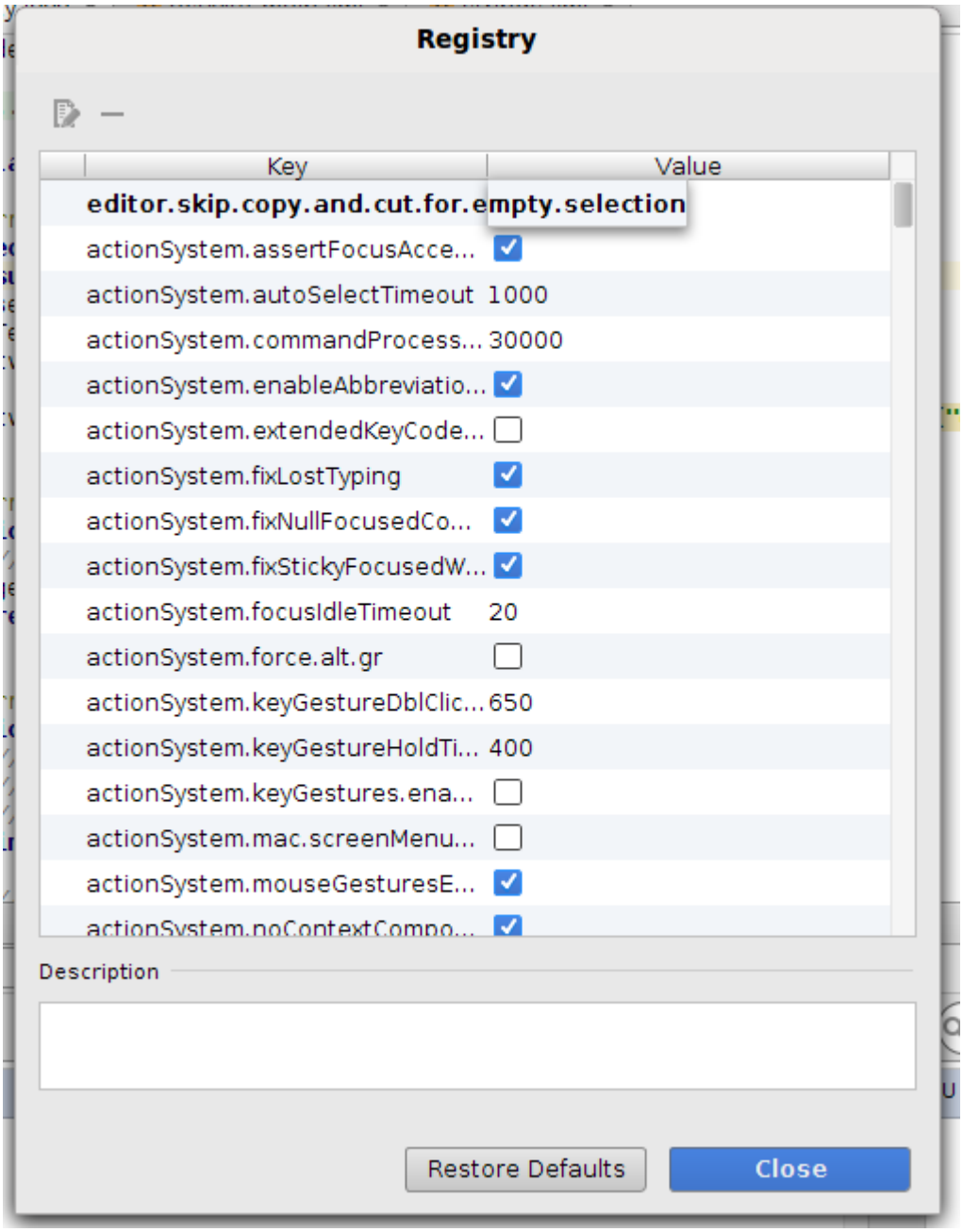


Abilita / disabilita la copia della linea vuota

`ctrl + alt + shift + /` (`cmd + alt + shift + /` su `MacOS`) dovrebbe mostrare la seguente finestra di dialogo:



Cliccando su `Registry` otterrete



Il tasto che si desidera abilitare / disabilitare è

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

Testato su `Linux Ubuntu` e `MacOS` .

Scorciatoie utili per Android Studio

Le seguenti sono alcune delle scorciatoie più comuni / utili.

Questi sono basati sulla mappa di collegamento IntelliJ predefinita. Puoi passare ad altre comuni mappe dei collegamenti IDE tramite `File -> Settings -> Keymap -> <Choose Eclipse/Visual Studio/etc from Keymaps dropdown>`

Azione	scorciatoia
Codice formato	CTRL + ALT + L
Aggiungi metodi non implementati	CTRL + I
Mostra logcat	ALT + 6
Costruire	CTRL + F9
Costruisci ed esegui	CTRL + F10
Trova	CTRL + F
Trova nel progetto	CTRL + MAIUSC + F
Trova e sostituisci	CTRL + R
Trova e sostituisci nel progetto	CTRL + MAIUSC + R
Sovrascrivi i metodi	CTRL + O
Mostra il progetto	ALT + 1
Nascondi progetto - logcat	MAIUSC + ESC
Comprimi tutto	CTRL + MAIUSC + NumPad +
Visualizza i punti di debug	CTRL + MAIUSC + F8
Espandi tutto	CTRL + MAIUSC + NumPad -
Apri Impostazioni	ALT + s
Seleziona destinazione (apre il file corrente nella vista Progetto)	ALT + F1 → INVIO
Cerca ovunque	SHIFT → SHIFT (Double shift)
Codice | Con Surround	CTRL → ALT + T
Crea il codice del modulo selezionato per il metodo	ALT + CTRL

Refactor:

Azione	scorciatoia
Refactor Questo (menu / selettore per tutte le azioni refactor applicabili dell'elemento corrente)	Mac CTRL + T - Win / Linux CTRL + ALT + T

Azione	scorciatoia
Rinominare	`MAIUSC` + `F6`
Metodo di estrazione	Mac `CMD` + `ALT` + `M` - Win / Linux `CTRL` + `ALT` + `M`
Estrai parametro	Mac `CMD` + `ALT` + `P` - Win / Linux `CTRL` + `ALT` + `P`
Estrai variabile	Mac `CMD` + `ALT` + `V` - Win / Linux `CTRL` + `ALT` + `V`

Studio Android Migliora il suggerimento sulle prestazioni

Abilita il lavoro offline:

1. Fai clic su File -> Impostazioni. Cerca "gradle" e fai clic nella casella di `Offline work` .
2. Vai al compilatore (nella stessa finestra di dialogo delle impostazioni appena sotto `Gradle`) e aggiungi `--offline` alla casella di testo `Command-line Options` di `Command-line Options` .

Migliora le prestazioni di Gradle

Aggiungi le seguenti due righe di codice nel tuo file gradle.properties.

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

Aumentando il valore di `-Xmx` e `-Xms` nel file `studio.vmoptions`

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

Finestra

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

Mac

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

Linux

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

Imposta Android Studio

Requisiti di sistema

- Microsoft® Windows® 8/7 / Vista / 2003 (32 o 64 bit).
- Mac® OS X® 10.8.5 o versioni successive, fino a 10.9 (Mavericks)
- Desktop GNOME o KDE

Installazione

Finestra

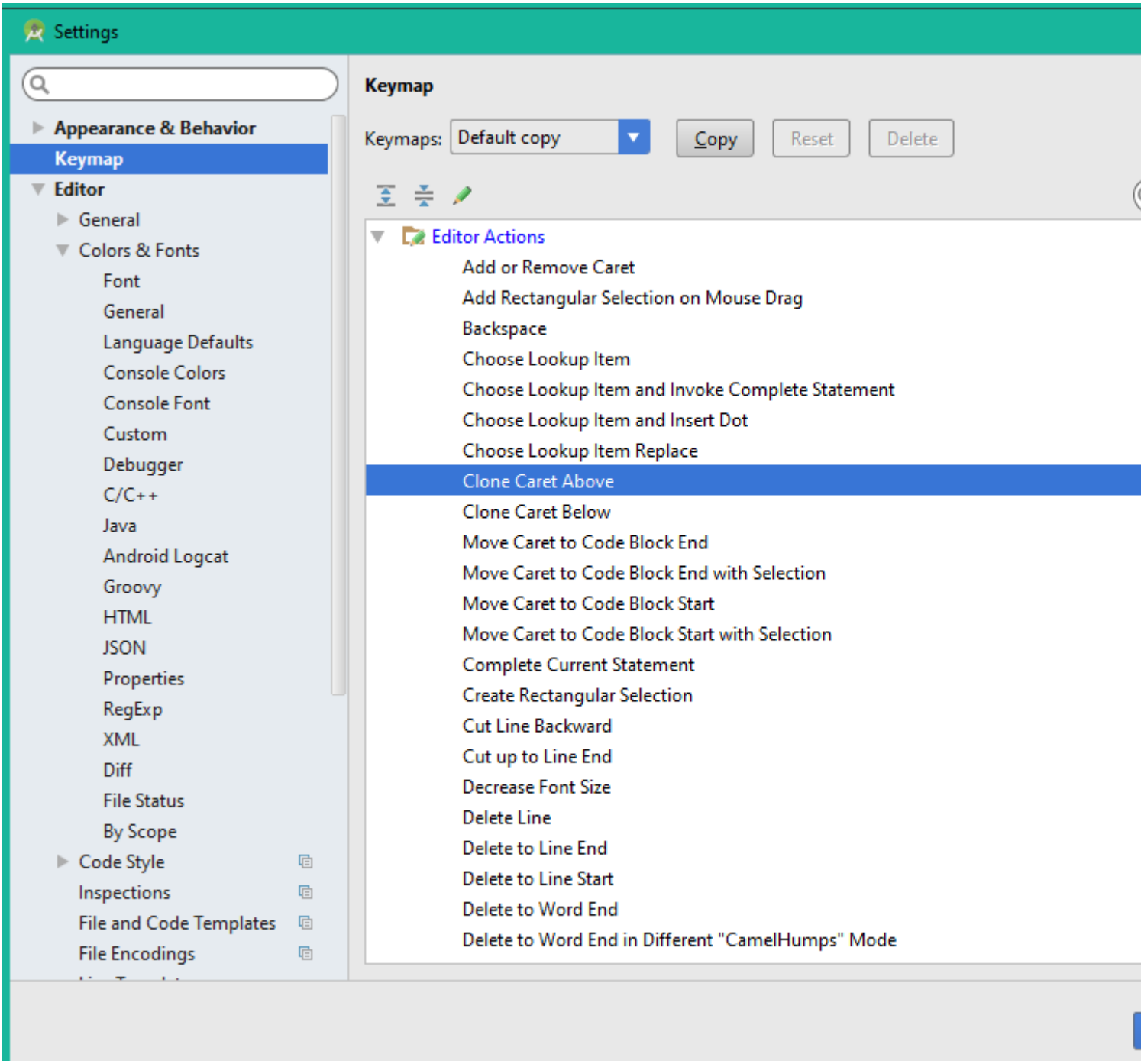
1. Scarica e installa JDK (Java Development Kit) versione 8
2. Scarica Android Studio
3. Avvia `Android Studio.exe` quindi menziona il percorso JDK e scarica l'ultimo SDK

Linux

1. Scarica e installa JDK (Java Development Kit) versione 8
2. Scarica Android Studio
3. Estrai il file zip
4. Aprire il terminale, cd nella cartella estratta, cd in bin (esempio `cd android-studio/bin`)
5. Esegui ./studio.sh

Visualizza e aggiungi scorciatoie in Android Studio

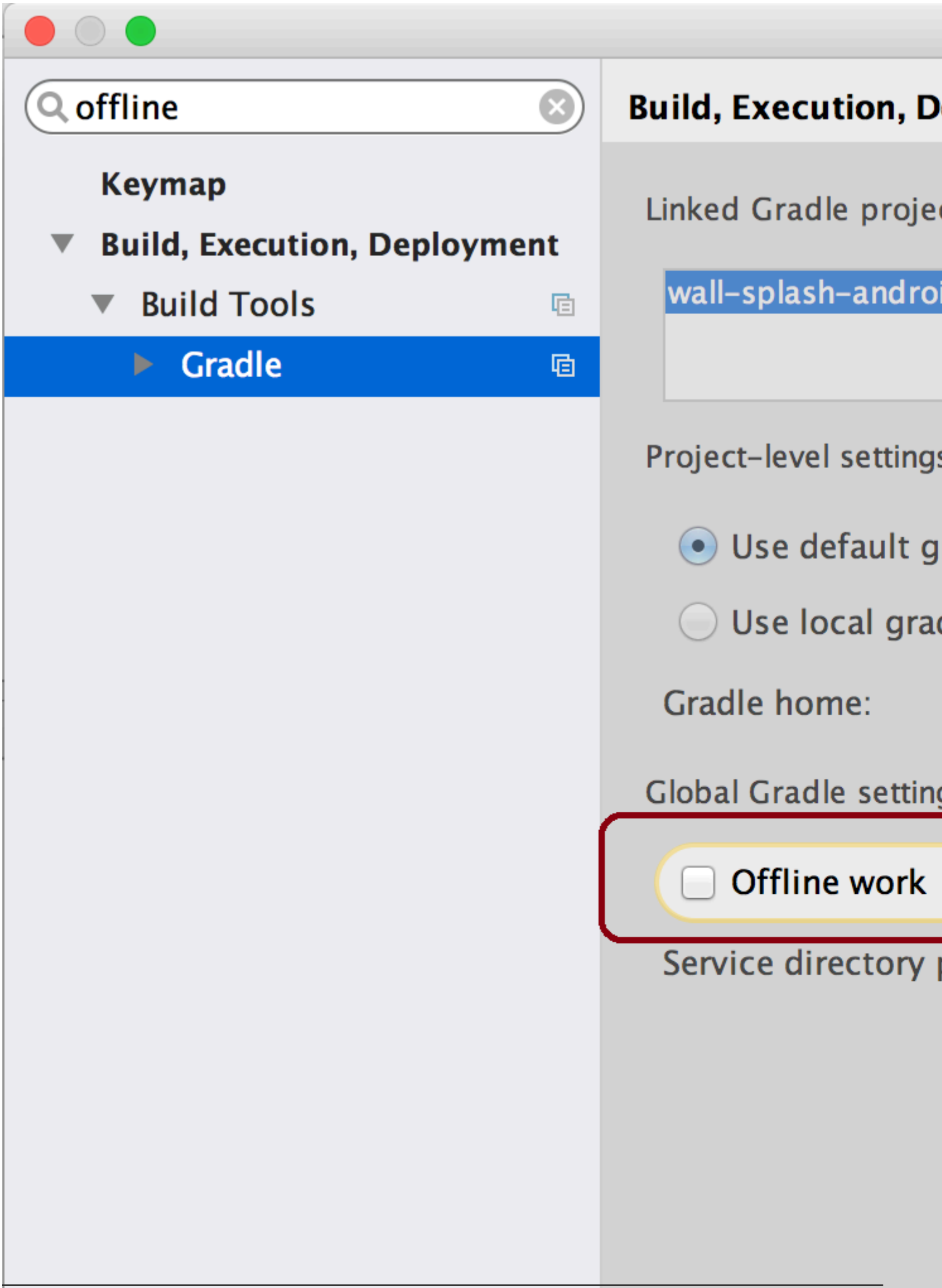
Andando su Impostazioni >> Keymap Viene visualizzata una finestra che mostra tutte le `Editor Actions` con il loro nome e scorciatoie. Alcune delle `Editor Actions` non hanno scorciatoie. Quindi, fai clic con il pulsante destro del mouse e aggiungi una nuova scorciatoia.
Controlla l'immagine qui sotto



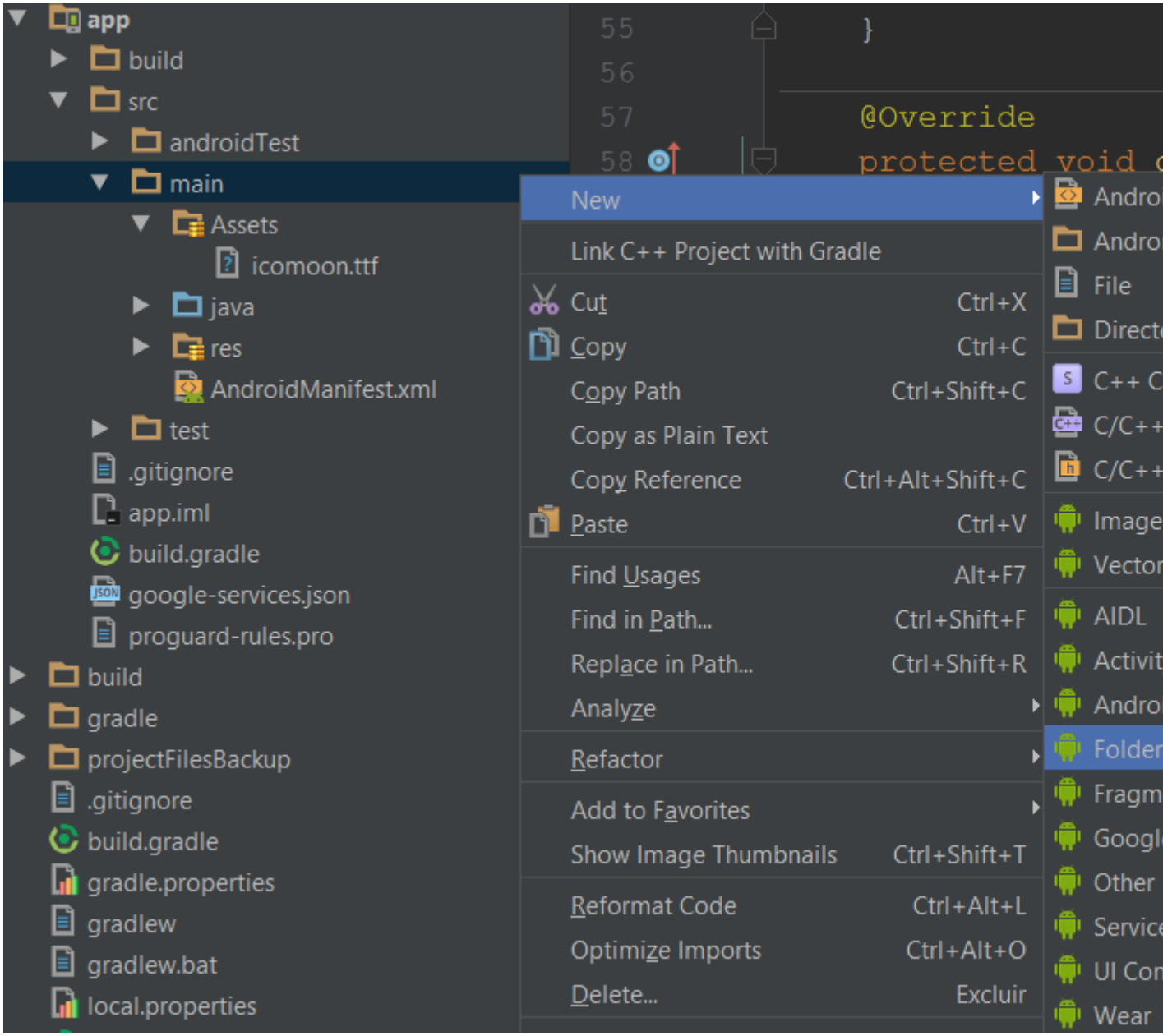
Il progetto di costruzione di Gradle dura per sempre

Android Studio -> **Preferenze** -> **Gradle** -> Spunta **Offline,** quindi riavvia il tuo studio Android.

Schermata di riferimento:


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 cartella delle risorse si trova nella cartella MAIN con lo stesso simbolo della cartella RES.
- In questo esempio ho inserito un file di font.



Leggi Studio Android online: https://riptutorial.com/it/android/topic/107/studio-android

Capitolo 230: Suono e supporti Android

Examples

Come scegliere immagini e video per api> 19

Ecco un codice testato per immagini e video. Funzionerà per tutte le API meno di 19 e anche oltre 19.

Immagine:

```
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);
                    }
```

Video:

```
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();
        }
```

Riproduci suoni tramite 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;
    }
}
```

Leggi Suono e supporti Android online: https://riptutorial.com/it/android/topic/4730/suono-e-supporti-android

Capitolo 231: Sviluppo di giochi Android

introduzione

Una breve introduzione alla creazione di un gioco sulla piattaforma Android utilizzando Java

Osservazioni

- Il primo esempio illustra le nozioni di base: non ci sono obiettivi, ma mostra come si crea una parte di base di un gioco 2D usando SurfaceView.
- Assicurati di salvare tutti i dati importanti quando crei un gioco; tutto il resto andrà perso

Examples

Gioco con Canvas e SurfaceView

Questo spiega come è possibile creare un gioco 2D di base usando SurfaceView.

Innanzitutto, abbiamo bisogno di un'attività:

```
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.
    }

}
```

L'attività deve essere dichiarata anche nel Manifest Android.

Ora per il gioco stesso. Innanzitutto, iniziamo implementando un thread di gioco:

```
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");
    }
}
```

Questa è la parte fondamentale. Ora hai la possibilità di disegnare sullo schermo.

Ora, iniziamo aggiungendo ai numeri interi:

```
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;
```

Per questa parte successiva, avrai bisogno di un'immagine. Dovrebbe essere di circa 100x100 ma può essere più grande o più piccolo. Per l'apprendimento, può anche essere usato un Rect (ma ciò richiede un cambiamento nel codice un po 'più in basso)

Ora dichiariamo una bitmap:

```
private Bitmap PLAYER_BMP = BitmapFactory.decodeResource(getResources(),
R.drawable.my_player_drawable);
```

Nel rendering, dobbiamo disegnare questa bitmap.

```
...
c.drawBitmap(PLAYER_BMP, x, y, null);
...
```

PRIMA DI LANCIO ci sono ancora alcune cose da fare

Abbiamo bisogno di un primo booleano:

```
boolean up = false;
```

in onTouchEvent, aggiungiamo:

```
if(ev.getAction() == MotionEvent.ACTION_DOWN){
    up = true;
}else if(ev.getAction() == MotionEvent.ACTION_UP){
    up = false;
}
```

E in tick abbiamo bisogno di questo per spostare il giocatore:

```
if(up){
    velY -=1;
}
```

```
else{
    velY +=1;
}
if(velY >14)velY = 14;
if(velY <-14)velY = -14;
y += velY *2;
```

e ora abbiamo bisogno di questo in 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();
```

E abbiamo bisogno di questi per variabili:

```
public static int WIDTH, HEIGHT;
```

A questo punto, il gioco è eseguibile. Significa che puoi lanciarlo e testarlo.

Ora dovresti avere un'immagine del giocatore o un rect che va su e giù per lo schermo. Il giocatore può essere creato come una classe personalizzata, se necessario. Quindi tutte le cose relative ai giocatori possono essere spostate in quella classe e usare un'istanza di quella classe per spostare, eseguire il rendering e fare altra logica.

Ora, come probabilmente hai visto sotto test, vola fuori dallo schermo. Quindi dobbiamo limitarlo.

Innanzitutto, dobbiamo dichiarare il Rect:

```
private Rect screen;
```

In init, dopo aver inizializzato la larghezza e l'altezza, creiamo un nuovo rect che è lo schermo.

```
screen = new Rect(0,0,WIDTH,HEIGHT);
```

Ora abbiamo bisogno di un altro rect sotto forma di un metodo:

```
private Rect getPlayerBound(){
    return new Rect(x, y, x + PLAYER_BMP.getWidth(), y + PLAYER_BMP.getHeight();
}
```

e in tick:

```
if(!getPlayerBound().intersects(screen){
    gameOver = true;
}
```

L'implementazione di gameover può anche essere utilizzata per mostrare l'inizio di un gioco.

Altri aspetti di un gioco degno di nota:

Salvataggio (attualmente mancante nella documentazione)

Leggi Sviluppo di giochi Android online: https://riptutorial.com/it/android/topic/10011/sviluppo-di-giochi-android

Capitolo 232: SyncAdapter esegue periodicamente la sincronizzazione dei dati

introduzione

Il componente dell'adattatore di sincronizzazione nella tua app incapsula il codice per le attività che trasferiscono i dati tra il dispositivo e un server. In base alla pianificazione e ai trigger forniti nella tua app, il framework dell'adattatore di sincronizzazione esegue il codice nel componente dell'adattatore di sincronizzazione.

Recentemente ho lavorato su SyncAdapter, voglio condividere le mie conoscenze con gli altri, può aiutare gli altri.

Examples

Adattatore di sincronizzazione con ogni valore minimo richiesto dal server.

```
<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>
```

Questo codice deve essere aggiunto nel file manifest

Nel codice precedente abbiamo il servizio syncservice e conteprovider e authenticatorservice.

In app dobbiamo creare il pacchetto xml per aggiungere file xc syncadpter e 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"/>
```

Autenticatore

```
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,COL

        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;
    }
}
```

Attività principale

```
     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();
}
```

}

Leggi SyncAdapter esegue periodicamente la sincronizzazione dei dati online: https://riptutorial.com/it/android/topic/10774/syncadapter-esegue-periodicamente-la-sincronizzazione-dei-dati

Capitolo 233: TabLayout

Examples

Utilizzando un TabLayout senza ViewPager

Il più delle volte un `TabLayout` viene utilizzato insieme a un ViewPager , al fine di ottenere la funzionalità di scorrimento che viene fornito con esso.

È possibile utilizzare `TabLayout` senza `ViewPager` utilizzando `TabLayout.OnTabSelectedListener` .

Innanzitutto, aggiungi un `TabLayout` al file XML della tua attività:

```
<android.support.design.widget.TabLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:id="@+id/tabLayout" />
```

Per la navigazione all'interno di `Activity` , popolare manualmente l'interfaccia utente in base alla scheda selezionata.

```
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) {

    }
});
```

Leggi TabLayout online: https://riptutorial.com/it/android/topic/7601/tablayout

Capitolo 234: Tastiera

Examples

Nascondi tastiera quando l'utente tocca da qualsiasi altra parte sullo schermo

Aggiungi codice nella tua **attività** .

Funzionerebbe anche per **Fragment** , **non c'è bisogno** di aggiungere questo codice in **Fragment** .

```
@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);
}
```

Registrare una richiamata per la tastiera aperta e chiusa

L'idea è di misurare un layout prima e dopo ogni modifica e se c'è un cambiamento significativo puoi essere piuttosto sicuro che sia il softkey.

```
// 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;
        }
    }
};
```

quindi nel nostro `onCreate` imposta il valore iniziale per `mLastContentHeight`

```
mLastContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();
```

e aggiungi l'ascoltatore

```
rootView.getViewTreeObserver().addOnGlobalLayoutListener(keyboardLayoutListener);
```

non dimenticare di rimuovere l'ascoltatore su `destroy`

```
rootView.getViewTreeObserver().removeOnGlobalLayoutListener(keyboardLayoutListener);
```

Leggi Tastiera online: https://riptutorial.com/it/android/topic/5606/tastiera

Capitolo 235: Tema DayNight (AppCompat v23.2 / API 14+)

Examples

Aggiunta del tema DayNight a un'app

Il tema DayNight offre a un'app la capacità di cambiare schemi di colori in base all'ora del giorno e all'ultima posizione nota del dispositivo.

Aggiungi quanto segue al tuo `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>
```

I temi che puoi estendere da aggiungere alla funzionalità di commutazione del tema giorno notte sono i seguenti:

- `"Theme.AppCompat.DayNight"`
- `"Theme.AppCompat.DayNight.NoActionBar"`
- `"Theme.AppCompat.DayNight.DarkActionBar"`

Oltre a `colorPrimary` , `colorPrimaryDark` e `colorAccent` , puoi anche aggiungere qualsiasi altro colore che vorresti cambiare, ad esempio `textColorPrimary` o `textColorSecondary` . Puoi anche aggiungere i colori personalizzati della tua app a questo `style` .

Per il passaggio da un tema all'altro, è necessario definire un `colors.xml` predefinito nella directory `res/values` e un altro `colors.xml` nella `colors.xml res/values-night` e definire i colori giorno / notte in modo appropriato.

Per cambiare tema, chiama il `AppCompatDelegate.setDefaultNightMode(int)` dal tuo codice Java. (Questo cambierà lo schema dei colori per l'intera app, non solo una qualsiasi attività o frammento.) Ad esempio:

```
AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
```

Puoi passare uno dei seguenti tre in base alla tua scelta:

- `AppCompatDelegate.MODE_NIGHT_NO` : imposta il tema predefinito per la tua app e prende i colori definiti nella directory `res/values` . Si consiglia di utilizzare colori chiari per questo tema.
- `AppCompatDelegate.MODE_NIGHT_YES` : imposta un tema notturno per la tua app e prende i colori definiti nella directory `res/values-night` . Si consiglia di utilizzare i colori scuri per questo tema.

- `AppCompatDelegate.MODE_NIGHT_AUTO` : questo cambia automaticamente i colori dell'app in base all'ora del giorno e ai colori definiti in `values` e `values-night` directory `values-night` .

È anche possibile ottenere lo stato attuale della modalità notte utilizzando il metodo `getDefaultNightMode()` . Per esempio:

```
int modeType = AppCompatDelegate.getDefaultNightMode();
```

Si noti, tuttavia, che l'interruttore del tema non persisterà se si uccide l'app e la si riapre. Se lo fai, il tema ritorna a `AppCompatDelegate.MODE_NIGHT_AUTO` , che è il valore predefinito. Se vuoi che il tema rimanga permanente, assicurati di memorizzare il valore nelle preferenze condivise e carica il valore memorizzato ogni volta che l'app viene aperta dopo che è stata distrutta.

Leggi Tema DayNight (AppCompat v23.2 / API 14+) online:
https://riptutorial.com/it/android/topic/7650/tema-daynight--appcompat-v23-2---api-14plus-

Capitolo 236: Tema, stile, attributo

Examples

Usa il tema personalizzato a livello globale

In themes.xml:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

In AndroidManifest.xml:

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">

    <!-- Activity declarations here -->

</application>
```

Definisci colori primari, scuri primari e accenti

È possibile personalizzare la tavolozza dei colori del tema .

Utilizzo **delle** API **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>
```

Utilizzo della **libreria di supporto Appcompat** (e `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>
```

Usa tema personalizzato per attività

In themes.xml:

```
<style name="MyActivityTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

In 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>
```

Overscroll Color (API 21+)

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorEdgeEffect">@color/my_color</item>
</style>
```

Ripple Color (API 21+)

5.0

L'animazione ripple viene mostrata quando l'utente preme viste cliccabili.

Puoi utilizzare lo stesso colore di ripple utilizzato dalla tua app che assegna il `?android:colorControlHighlight` nelle tue visualizzazioni. Puoi personalizzare questo colore modificando l'attributo `android:colorControlHighlight` nel tuo tema:

Questo colore dell'effetto può essere cambiato:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorControlHighlight">@color/my_color</item>
</style>
```

Oppure, se stai usando un tema materiale:

```
<style name="AppTheme" parent="android:Theme.Material.Light">
    <item name="android:colorControlHighlight">@color/your_custom_color</item>
</style>
```

Light Status Bar (API 23+)

Questo attributo può cambiare lo sfondo delle icone della barra di stato (nella parte superiore dello

schermo) in bianco.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowLightStatusBar">true</item>
</style>
```

Navigazione traslucida e barre di stato (API 19+)

La barra di navigazione (nella parte inferiore dello schermo) può essere trasparente. Ecco il modo per raggiungerlo.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentNavigation">true</item>
</style>
```

La barra di stato (in alto sullo schermo) può essere resa trasparente, applicando questo attributo allo stile:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentStatus">true</item>
</style>
```

Colore barra di navigazione (API 21+)

5.0

Questo attributo viene utilizzato per modificare la barra di navigazione (una, che contiene il pulsante Indietro, Home recenti). Di solito è nero, tuttavia è possibile cambiare il colore.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:navigationBarColor">@color/my_color</item>
</style>
```

Eredità tematica

Quando si definiscono temi, in genere si utilizza il tema fornito dal sistema, quindi le modifiche modificano l'aspetto per adattarlo alla propria applicazione. Ad esempio, questo è il modo in cui il tema `Theme.AppCompat` è ereditato:

```
<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>
```

Questo tema ora ha tutte le proprietà del tema `Theme.AppCompat` standard, ad eccezione di quelli che abbiamo modificato in modo esplicito.

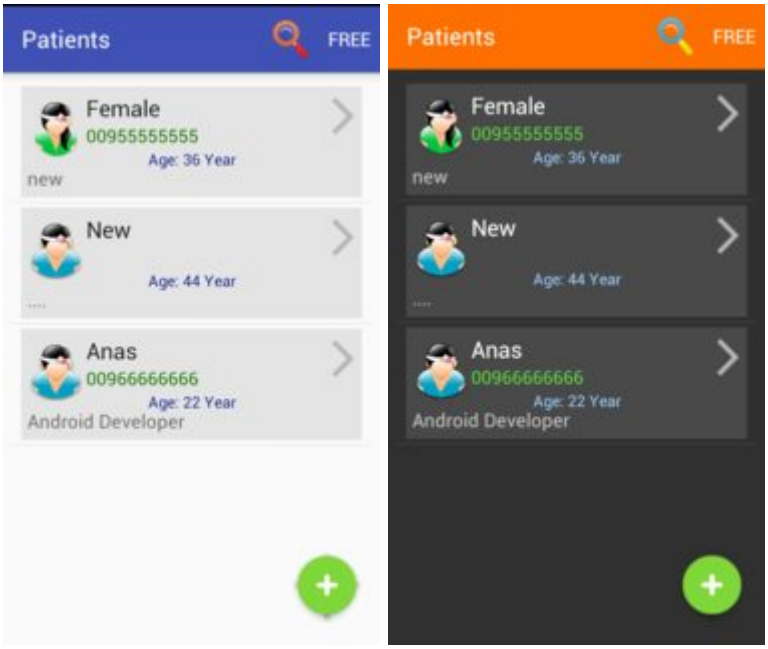
C'è anche una scorciatoia quando si eredita, di solito usata quando si eredita dal proprio tema:

```
<style name="AppTheme.Red">
    <item name="colorAccent">@color/red</item>
</style>
```

Dal momento che ha già `AppTheme.` all'inizio del suo nome, lo eredita automaticamente, senza dover definire il tema `parent` . Ciò è utile quando devi creare stili specifici per una parte (ad esempio, una singola attività) della tua app.

Temi multipli in un'unica app

Utilizzando più di un tema nella tua applicazione Android, puoi aggiungere colori personalizzati a ogni tema, per essere così:



Innanzitutto, dobbiamo aggiungere i nostri temi a `style.xml` questo modo:

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">

</style>

<!--  -->
<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >

</style>
......
```

Sopra puoi vedere **OneTheme** e **TwoTheme** .

Ora vai sul tuo `AndroidManifest.xml` e aggiungi questa linea: `android:theme="@style/OneTheme"` al tag *dell'applicazione* , questo renderà **OneTheme** il tema predefinito:

```
<application
        android:theme="@style/OneTheme"
        ...>
```

Crea un nuovo file xml chiamato `attrs.xml` e aggiungi questo codice:

```
<?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) -->
```

Torna a `style.xml` e aggiungi questi colori con i relativi valori per ciascun 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>
```

Ora hai colori personalizzati per ogni tema, aggiungiamo questi colori alle nostre visualizzazioni.

Aggiungi **custom_blue** color a TextView usando "? Attr /":

Vai alla tua imageView e aggiungi questo colore:

```
<TextView>
    android:id="@+id/txte_view"
    android:textColor="?attr/custom_blue" />
```

Mow possiamo cambiare il tema solo con la singola riga `setTheme(R.style.TwoTheme);` questa linea deve essere prima del metodo `setContentView()` nel metodo `onCreate()` , come questo `Activity.java` :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setTheme(R.style.TwoTheme);
    setContentView(R.layout.main_activity);
    ....
}
```

---

cambia tema per tutte le attività contemporaneamente

Se vogliamo cambiare il tema per tutte le attività, dobbiamo creare una nuova classe denominata MyActivity estende la classe `AppCompatActivity` (o la classe `Activity`) e aggiunge line `setTheme(R.style.TwoTheme);` al metodo **onCreate ()** :

```
public class MyActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (new MySettings(this).isDarkTheme())
            setTheme(R.style.TwoTheme);
    }
}
```

Infine, vai a tutte le tue attività aggiungi fai in modo che tutti estendano la classe base **MyActivity** :

```
public class MainActivity extends MyActivity {
    ....
}
```

Per cambiare il tema, vai su **MyActivity** e modifica `R.style.TwoTheme` sul tema (`R.style.OneTheme` , `R.style.ThreeTheme` ....).

Leggi Tema, stile, attributo online: https://riptutorial.com/it/android/topic/1843/tema--stile--attributo

Capitolo 237: tensorflow

introduzione

TensorFlow è stato progettato pensando alle piattaforme mobili e integrate. Abbiamo codice di esempio e supporto per la build che puoi provare ora per queste piattaforme:

Android iOS Raspberry Pi

Osservazioni

Lavoro apprezzato di MindRocks

Examples

Come usare

Installa Bazel da qui . Bazel è il sistema di compilazione principale per TensorFlow. Ora, modifica il WORKSPACE, possiamo trovare il file WORKSPACE nella directory root del TensorFlow che abbiamo clonato in precedenza.

```
# 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)
```

Come sotto con il nostro percorso sdk e 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)
```

Leggi tensorflow online: https://riptutorial.com/it/android/topic/9991/tensorflow

Capitolo 238: Test dell'interfaccia utente con Espresso

Osservazioni

Caffè espresso

Il cheat dell'Espresso ti aiuterà a scrivere i tuoi test e ciò che vuoi testare:

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Inoltre sempre un buon punto di riferimento è la documentazione ufficiale:

https://google.github.io/android-testing-support-library/docs/espresso/index.html

Suggerimenti avanzati sui video per l'espresso di Google:
https://www.youtube.com/watch?v=isihPOY2vS4

Risoluzione dei problemi

- Quando provi a scorrere, assicurati di chiudere prima la tastiera:

Watchout: non usare la versione "Espresso" non farà nulla se usato al di fuori di una ViewAction. Questo potrebbe non essere ovvio se si ha un'importazione sulla versione ViewAction poiché hanno esattamente lo stesso nome del metodo.

```
ViewActions.closeSoftKeyboard;
Espresso.closeSoftKeyboard();
```

- Quando si eseguono i test insieme in una suite anziché individualmente, tenere presente che l'attività del test precedente potrebbe essere ancora in esecuzione. Non fare affidamento sul fatto che onDestroy () del test precedente venga chiamato prima degli attuali test suResume (). **Si scopre che questo è in realtà un bug** : http://b.android.com/201513

Examples

Preparare l'espresso

Nel file `build.gradle` del modulo dell'app Android aggiungi le dipendenze successive:

```
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'
}
```

Specificare `AndroidJUnitRunner` per il parametro `testInstrumentationRunner` nel file `build.gradle` .

```
android {

  defaultConfig {
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }

}
```

Inoltre, aggiungi questa dipendenza per fornire supporto di simulazione di intenti

```
androidTestCompile 'com.android.support.test.espresso:espresso-intents:2.2.2'
```

E aggiungi questo per il supporto per i test di webview

```
// Espresso-web for WebView support
androidTestCompile 'com.android.support.test.espresso:espresso-web:2.2.2'
```

Crea una classe di test Espresso

Posiziona la prossima classe java in **src / androidTest / java** ed **eseguila** .

```
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
  }

}
```

Apri Chiudi DrawerLayout

```
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);
      }
    };
  }

}
```

Espresso semplice test dell'interfaccia utente

Strumenti di test dell'interfaccia utente

Due strumenti principali oggi utilizzati principalmente per i test dell'interfaccia utente sono Appium ed Espresso.

Appium	Caffè espresso
test blackbox	test bianco / grigio
quello che vedi è ciò che puoi testare	può cambiare il funzionamento interno dell'app e prepararlo per il test, ad esempio salvare alcuni dati nel database o sharedpreferences prima di eseguire il test
utilizzato principalmente per l'integrazione end-to-end e l'intero flusso di utenti	testare la funzionalità di uno schermo e / o flusso
può essere estratto in modo che il test scritto possa essere eseguito	Solo Android

Appium	Caffè espresso
su iOS e Android	
ben supportato	ben supportato
supporta il test parallelo su più dispositivi con griglia di selenio	I test paralleli non sono fuori dalla scatola, plug-in come Spoon esiste fino a quando non viene fuori il vero supporto di Google

Come aggiungere l'espresso al progetto

```
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 Se si utilizzano le librerie di supporto più recenti, le annotazioni ecc. È necessario escludere le versioni precedenti dall'espresso per evitare collisioni:

```
    // 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'
    }
```

Oltre a queste importazioni è necessario aggiungere il correttore di prova di strumentazione Android a build.gradle android.defaultConfig:

```
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
```

Configurazione del dispositivo

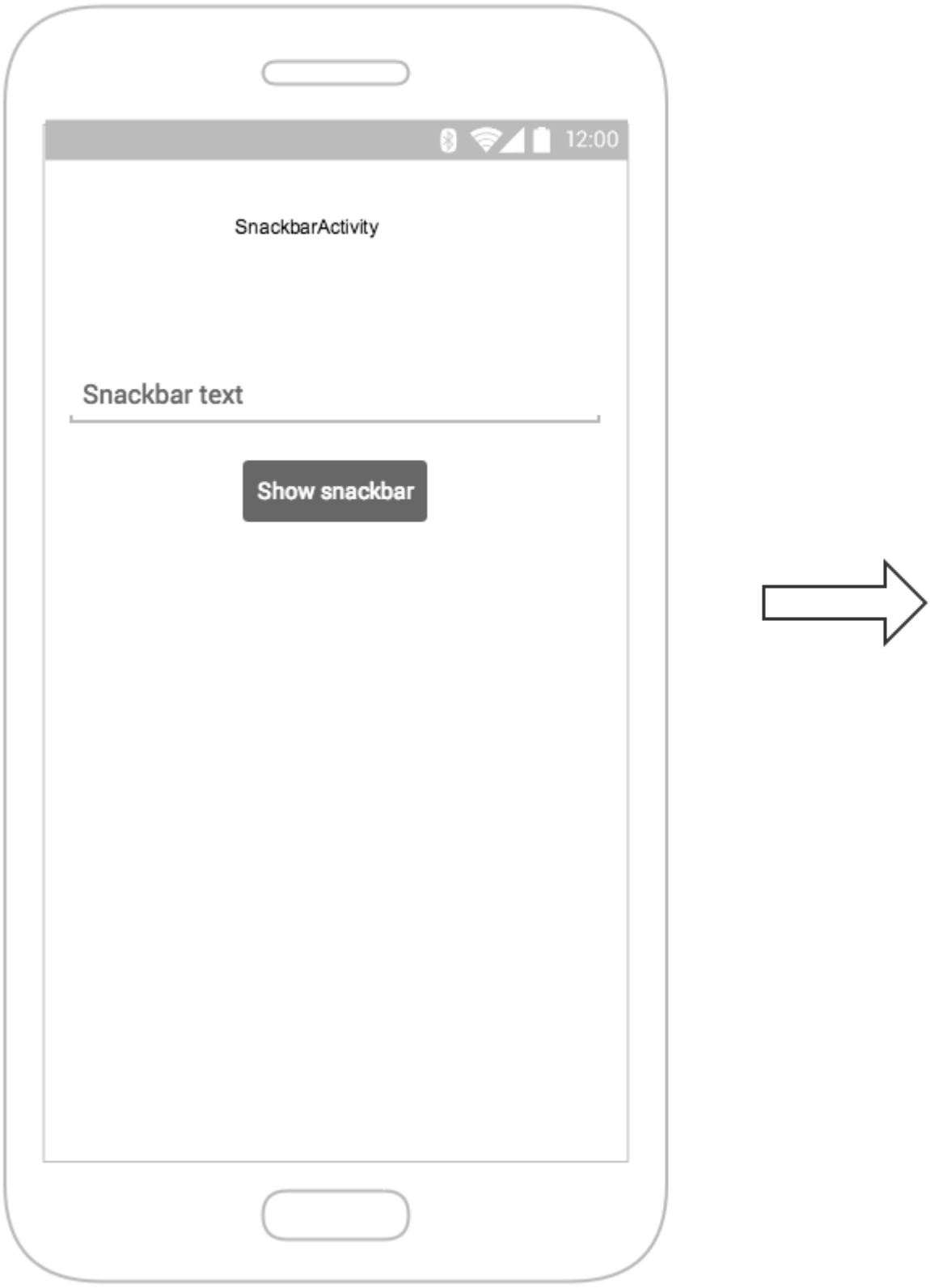
Per i test non a fiocchi si consiglia di impostare le seguenti impostazioni sui dispositivi:

- Opzioni sviluppatore / Disattiva animazioni: riduce la screpolatura dei test
- Opzioni per lo sviluppatore / Rimani sveglio: se hai dispositivi dedicati per i test, questo è utile
- Opzioni dello sviluppatore / Dimensioni del buffer del logger: impostare su un numero più alto se si eseguono suite di test molto grandi sul telefono
- Accessibilità / Ritardo tocco e attesa - lunga per evitare problemi con il picchiettare nell'espresso

Abbastanza una configurazione dal mondo reale ha? Bene, ora, quando è fuori mano, diamo un'occhiata a come impostare un piccolo test

Scrivere il test

Supponiamo che abbiamo la seguente schermata:



Lo schermo contiene:

- campo di immissione del testo - **R.id.textEntry**
- pulsante che mostra la barra degli snack con testo digitato quando si fa clic - **R.id.shownSnackbarBtn**
- snackbar che dovrebbe contenere testo digitato dall'utente - **android.support.design.R.id.snackbar_text**

Ora creiamo una classe che metterà alla prova il nostro flusso:

```
/**
* 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()));
    }
}
```

Come hai notato, ci sono 3-4 cose che potresti notare venire spesso:

onView (withXYZ) <- viewMatchers con loro sei in grado di trovare elementi sullo schermo

perform (click ()) <- viewActions, puoi eseguire azioni sugli elementi che hai precedentemente trovato

check (matches (isDisplayed ())) <- viewAssertions, controlla che vuoi fare sulle schermate che

hai trovato in precedenza

Tutti questi e molti altri possono essere trovati qui: https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/index.html

Ecco fatto, ora è possibile eseguire il test sia facendo clic destro sul nome della classe / test e selezionando Esegui test o con comando:

```
./gradlew connectedFLAVORNAMEAndroidTest
```

Navigazione

```
@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()));
}
```

Si noti che questa è una soluzione alternativa e si scontrerà con altre visualizzazioni che hanno la stessa descrizione del contenuto.

Esecuzione di un'azione su una vista

È possibile eseguire `ViewActions` su una vista utilizzando il metodo perform.
La classe `ViewActions` fornisce metodi di supporto per le azioni più comuni, come:

```
ViewActions.click()
ViewActions.typeText()
ViewActions.clearText()
```

Ad esempio, per fare clic sulla vista:

```
onView(...).perform(click());
onView(withId(R.id.button_simple)).perform(click());
```

È possibile eseguire più di un'azione con una chiamata di esecuzione:

```
onView(...).perform(typeText("Hello"), click());
```

Se la vista su cui stai lavorando si trova all'interno di una `ScrollView` (verticale o orizzontale), considera le azioni precedenti che richiedono la visualizzazione della vista (come `click()` e `typeText()`) con `scrollTo()` . Questo assicura che la vista sia visualizzata prima di procedere all'altra azione:

```
onView(...).perform(scrollTo(), click());
```

Trovare una vista con onView

Con `ViewMatchers` puoi trovare la vista nella gerarchia della vista corrente.

Per trovare una vista, utilizzare il metodo `onView()` con un visualizzatore di viste che seleziona la vista corretta. I metodi `onView()` restituiscono un oggetto di tipo `ViewInteraction` .

Ad esempio, trovare una vista dal suo `R.id` è semplice come:

```
onView(withId(R.id.my_view))
```

Trovare una vista con un testo:

```
onView(withText("Hello World"))
```

Abbinamenti personalizzati Espresso

Espresso di default ha molti matchers che ti aiutano a trovare le viste che ti servono per fare alcuni controlli o interazioni con loro.

I più importanti possono essere trovati nel seguente cheat sheet:

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Alcuni esempi di abbinamenti sono:

- withId (R.id.ID_of_object_you_are_looking_for);
- withText ("Qualche testo ti aspetti che l'oggetto abbia");
- isDisplayed () <- check è la vista visibile
- doesNotExist () <- controlla che la vista non esista

Tutti questi sono molto utili per l'uso quotidiano, ma se hai delle visioni più complesse scrivendo i tuoi abbinatori personalizzati puoi rendere i test più leggibili e puoi riutilizzarli in posti diversi.

Esistono 2 tipi di matcher più comuni che è possibile estendere: **TypeSafeMatcher** **BoundedMatcher**

L'implementazione di TypeSafeMatcher richiede di verificare l'istanza della vista su cui si sta facendo valere, se è il tipo corretto si abbinano alcune delle sue proprietà a un valore che si è fornito ad un corrispondente.

Ad esempio, digitare safe matcher che convalida una vista dell'immagine che abbia il drawable corretto:

```
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("]");
        }
    }
}
```

L'utilizzo del matcher può essere eseguito in questo modo:

```
  public static Matcher<View> withDrawable(final int resourceId) {
    return new DrawableMatcher(resourceId);
}

 onView(withDrawable(R.drawable.someDrawable)).check(matches(isDisplayed()));
```

Gli abbinamenti limitati sono simili, ma non devi fare il controllo del tipo, ma poiché ciò è fatto automaticamente per te:

```
 /**
 * 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());
        }
    };
}
```

Maggiori informazioni sui matchers possono essere lette su:

http://hamcrest.org/

https://developer.android.com/reference/android/support/test/espresso/matcher/ViewMatchers.html

Espresso in generale

Impostazione Espresso:

```
androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
androidTestCompile 'com.android.support.test:runner:0.5'
```

ViewMatchers : una raccolta di oggetti che implementano `Matcher<? super View>` interface. È possibile passare uno o più di questi al metodo `onView` per individuare una vista all'interno della gerarchia della vista corrente.

ViewActions : una raccolta di `ViewActions` che può essere passata al metodo `ViewInteraction.perform()` (ad esempio, `click()`).

ViewAssertions : una raccolta di `ViewAssertions` che può essere passata al metodo `ViewInteraction.check()` . Nella maggior parte dei casi, si utilizzerà l'asserzione di corrispondenze,

che utilizza un controllo di visualizzazione per affermare lo stato della vista correntemente selezionata.

Foglio di caffè espresso da google

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion);
```

View Matchers

USER PROPERTIES

```
withId(...)
withText(...)
withTagKey(...)
withTagValue(...)
hasContentDescription(...)
withContentDescription(...)
withHint(...)
withSpinnerText(...)
hasLinks()
hasEllipsizedText()
hasMultilineTest()
```

UI PROPERTIES

```
isDisplayed()
isCompletelyDisplayed()
isEnabled()
hasFocus()
isClickable()
isChecked()
isNotChecked()
withEffectiveVisibility(...)
isSelected()
```

OBJECT MATCHER

```
allOf(Matchers)
anyOf(Matchers)
is(...)
```

HIERARCHY

```
withParent(Matcher)
withChild(Matcher)
hasDescendant(Matcher)
isDescendantOfA(Matcher)
hasSibling(Matcher)
isRoot()
```

INPUT

```
supportsInputMethods(...)
hasIMEAction(...)
```

CLASS

```
isAssignableFrom(...)
withClassName(...)
```

ROOT MATCHERS

```
isFocusable()
isTouchable()
isDialog()
withDecorView()
isPlatformPopup()
```

SEE ALSO

Preference matchers

https://riptutorial.com/it/android/topic/3485/test-dell-interfaccia-utente-con-espresso

Capitolo 239: Test dell'interfaccia utente Inter-App con UIAutomator

Sintassi

- Strumentazione getInstrumentation ()
- UIDevice UiDevice.getInstance (Strumentazione di strumentazione)
- booleano UIDevice.pressHome ()
- booleano UIDevice.pressBack ()
- booleano UIDevice.pressRecentApps ()
- void UIDevice.wakeUp ()
- booleano UIDevice.swipe (int startX, int startY, int endX, int endY, int step)
- booleano UIDevice.drag (int startX, int startY, int endX, int endY, int step)
- UIObject2 UIDevice.findObject (By.desc (String contentDesc))
- booleano UIObject2.click ()

Osservazioni

UIAutomator è particolarmente utile per testare le storie degli utenti. Si verificano problemi se gli elementi di visualizzazione non hanno né un *ID risorsa* univoco né un *contenuto-desc* . Nella maggior parte dei casi c'è comunque un modo per completare il test, ciò richiede molto tempo. Se puoi influenzare il codice della tua app, UIAutomator potrebbe essere il tuo strumento di test.

Examples

Prepara il tuo progetto e scrivi il primo test UIAutomator

Aggiungi le librerie richieste nella sezione delle dipendenze del build.gradle del tuo modulo 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'
}
```

Si noti che, naturalmente, le versioni potrebbero differire nel frattempo.

Dopo questa sincronizzazione con le modifiche.

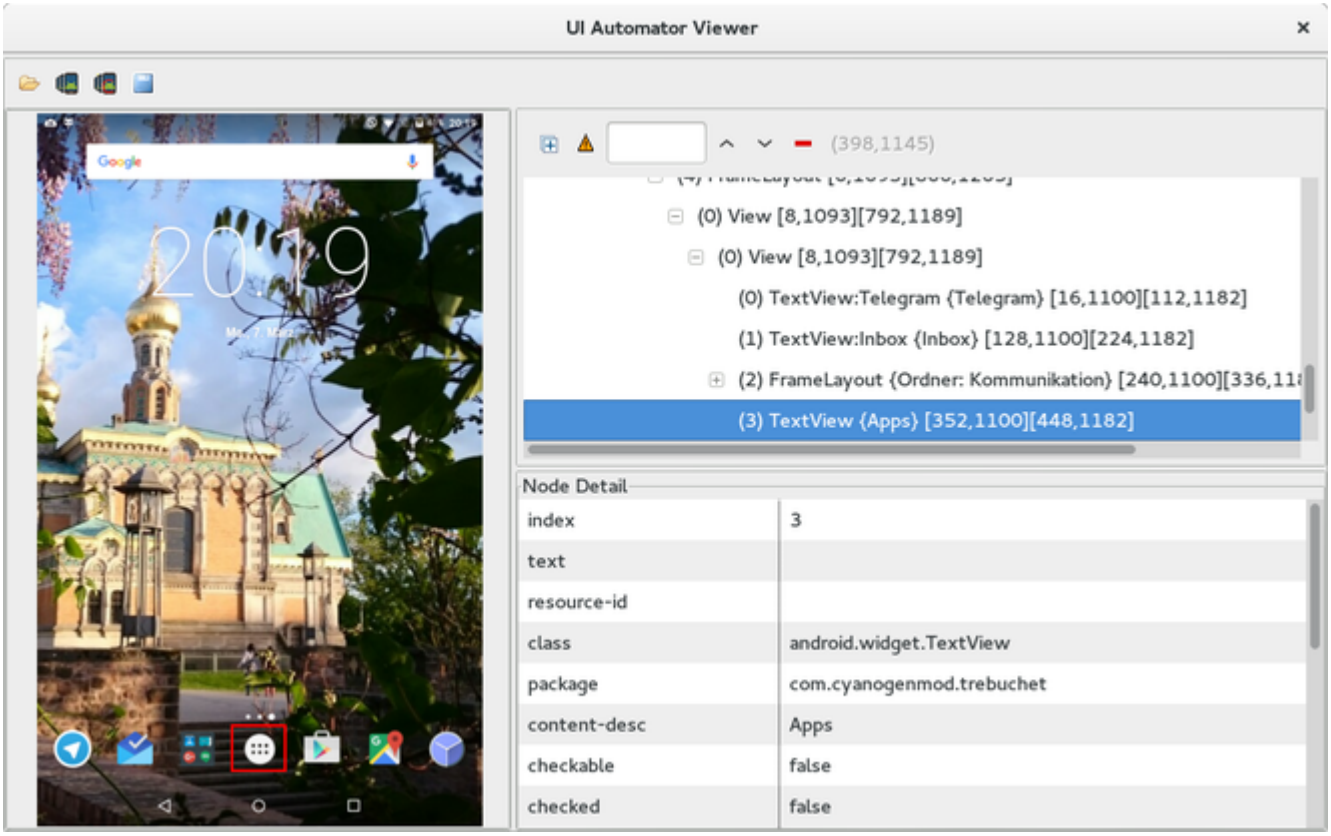
Quindi aggiungi una nuova classe Java nella cartella androidTest:

```
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();
  }
}
```

Facendo un clic destro sulla scheda della classe e su "Esegui" InterAppTest "esegue questo test.

Scrivere test più complessi usando UIAutomatorViewer

Per abilitare la scrittura di più complessi test dell'interfaccia utente è necessario *UIAutomatorViewer* . Lo strumento situato in */ tools /* crea uno screenshot a schermo intero che include i layout delle viste attualmente visualizzate. Guarda l'immagine successiva per avere un'idea di cosa viene mostrato:



Per i test dell'interfaccia utente cerchiamo *id risorsa* , *content-desc* o qualcos'altro per identificare

una vista e utilizzarla nei nostri test.

Il visualizzatore *uiautomator* viene eseguito tramite terminale.

Se ora, ad esempio, vogliamo fare clic sul pulsante delle applicazioni e quindi aprire alcune app e scorrere, ecco come può apparire il metodo di prova:

```
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)
}
```

Creazione di una suite di test dei test di UIAutomator

Mettere insieme i test di UIAutomator in una suite di test è una cosa rapida:

```
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 {}
```

Esegui simile a un singolo test facendo clic su destra ed esegui la suite.

Leggi Test dell'interfaccia utente Inter-App con UIAutomator online:
https://riptutorial.com/it/android/topic/6249/test-dell-interfaccia-utente-inter-app-con-uiautomator

Capitolo 240: Test delle unità in Android con JUnit

Osservazioni

- Vogella: unit test con JUnit
- Annotazioni Junit : java2novice.com
- Assert Class: junit.org
- JUnit Api: tutorialspoint.com
- Anroid che prova i messaggi su Medium.com

Examples

Creazione di test di unità locali

Inserisci qui le tue classi di test: `/src/test/<pkg_name>/`

Esempio di test di classe

```
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
    }
}
```

Abbattersi

```
public class ExampleUnitTest {
  ...
}
```

La classe di test, è possibile creare diverse classi di test e inserirle all'interno del pacchetto di test.

```
@Test
public void addition_isCorrect() {
   ...
}
```

Il metodo di prova, diversi metodi di prova possono essere creati all'interno di una classe di test.

Notare l'annotazione `@Test` .

> L'annotazione di Test dice a JUnit che il metodo pubblico vuoto a cui è collegato può essere eseguito come un caso di test.

Ci sono molte altre annotazioni utili come `@Before` , `@After` ecc. Questa pagina sarebbe un buon punto di partenza.

```
assertEquals(9, c); // This test passes
assertEquals(10, c); //Test fails
```

Questi metodi sono membri della classe `Assert` . Alcuni altri metodi utili sono `assertFalse()` , `assertNotNull()` , `assertTrue` ecc. Ecco una spiegazione dettagliata .

Informazioni sull'annotazione per il test JUnit:

@ Test **:** l'annotazione Test dice a JUnit che il metodo di annullamento pubblico a cui è collegato può essere eseguito come un caso di test. Per eseguire il metodo, JUnit prima costruisce una nuova istanza della classe, quindi richiama il metodo annotato.

@Prima: quando si scrivono i test, è comune scoprire che per diversi test sono necessari oggetti simili creati prima che possano essere eseguiti. `@Before` un metodo public void con `@Before` , il metodo viene eseguito prima del metodo Test.

@After: se si assegnano risorse esterne in un metodo Before è necessario rilasciarle dopo l'esecuzione del test. `@After` un metodo public void con `@After` , il metodo viene eseguito dopo il metodo Test. Tutti i metodi `@After` sono garantiti per l'esecuzione anche se un metodo Before o Test genera un'eccezione

---

Suggerimento Crea rapidamente classi di test in Android Studio

- Posiziona il cursore sul nome della classe per il quale vuoi creare una classe di test.
- Premi Alt + Invio (Windows).
- Seleziona Crea test, premi Invio.
- Selezionare i metodi per cui si desidera creare metodi di prova, fare clic su OK.
- Seleziona la directory in cui desideri creare la classe di test.
- Il gioco è fatto, questo è il tuo primo test.

Suggerimento Esegui facilmente i test in Android Studio

- Fare clic con il tasto destro sul pacchetto.
- Seleziona Esegui "Test in ...
- Tutti i test nel pacchetto verranno eseguiti contemporaneamente.

Spostamento della logica aziendale dai componenti Android

Gran parte del valore dei test delle unità JVM locali deriva dal modo in cui si progetta la propria applicazione. Devi progettarlo in modo tale da poter disgiungere la tua logica di business dai tuoi componenti Android. Ecco un esempio di questo modo utilizzando il modello Model-View-Presenter . Tienilo in pratica implementando una schermata di registrazione di base che richiede solo un nome utente e una password. La nostra app per Android è responsabile della convalida che il nome utente fornito dall'utente non sia vuoto e che la password sia lunga almeno otto caratteri e contenga almeno una cifra. Se il nome utente / password è valido, eseguiamo la nostra chiamata API di iscrizione, altrimenti viene visualizzato un messaggio di errore.

Esempio in cui la logica aziendale è fortemente accoppiata con 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();
        }
    }
}
```

Esempio in cui la logica aziendale è disaccoppiata dal componente Android.

Qui definiamo in una singola classe, LoginContract, che ospiterà le varie interazioni tra le nostre varie classi.

```
public interface LoginContract {
    public interface View {
        performSignUpApiCall(String username, String password);
        displayInvalidCredentialsErrorMessage();
    }
    public interface Presenter {
        void validateUserCredentials(String username, String password);
    }
}
```

Il nostro LoginActivity è per lo più lo stesso eccetto che abbiamo rimosso la responsabilità di dover sapere come convalidare il modulo di iscrizione di un utente (la nostra logica di business). LoginActivity ora farà affidamento sul nostro nuovo LoginPresenter per eseguire la convalida.

```
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);
    }
    ...
}
```

Ora la tua logica aziendale risiederà nella tua nuova classe 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();
        }
    }
}
```

E ora possiamo creare test di unità JVM locali contro la tua nuova classe 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);
    }
}
```

Come puoi vedere, quando abbiamo estratto la nostra logica aziendale da LoginActivity e l'abbiamo inserita nel POJO LoginPresenter. Ora possiamo creare test di unità JVM locali contro la nostra logica aziendale.

Dovrebbe essere notato che ci sono varie altre implicazioni dal nostro cambiamento di architettura, come il fatto che siamo vicini all'adesione ad ogni classe che ha una singola responsabilità, classi aggiuntive, ecc. Questi sono solo effetti collaterali del modo in cui scelgo di fare questo disaccoppiamento tramite lo stile MVP. MVP è solo un modo per farlo, ma ci sono altre alternative che potresti voler considerare come MVVM . Devi solo scegliere il miglior sistema che funzioni per te.

Iniziare con JUnit

Impostare

Per avviare l'unità testando il tuo progetto Android usando JUnit è necessario aggiungere la dipendenza JUnit al progetto ed è necessario creare un set di sorgenti di test che contenga il codice sorgente per i test di unità. I progetti creati con Android Studio spesso includono già la dipendenza JUnit e il set di origini di test

Aggiungi la seguente riga al tuo file `build.gradle` modulo all'interno delle dipendenze `Closure` :

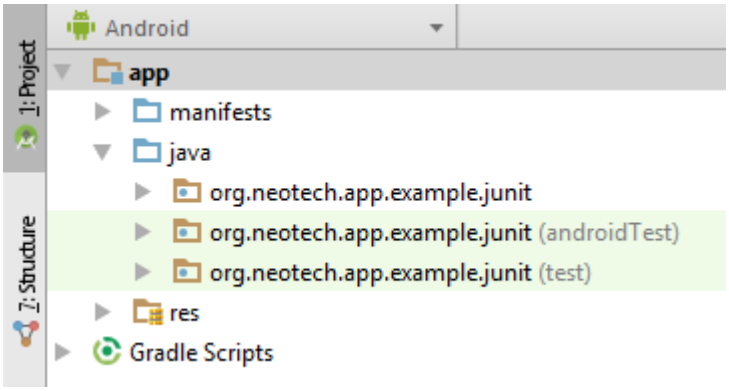
```
testCompile 'junit:junit:4.12'
```

Le classi di test JUnit si trovano in un `test` denominato set di origine. Se questo set di sorgenti non esiste devi creare tu stesso una nuova cartella. La struttura delle cartelle di un progetto predefinito di Android Studio (basato sul gradiente) ha il seguente aspetto:

```
<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)
```

Se il tuo progetto non ha la cartella `/app/src/test` devi crearlo da solo. All'interno della cartella di `test` è anche necessaria una cartella `java` (creala se non esiste). La cartella java nel set sorgente di `test` dovrebbe contenere la stessa struttura del pacchetto del set sorgente `main` .

Se la configurazione è corretta, la struttura del progetto (nella vista Android in Android Studio) dovrebbe essere simile alla seguente:



Nota: non è necessario disporre del `androidTest` origini `androidTest` , questo set di origine si trova spesso nei progetti creati da Android Studio ed è incluso qui per riferimento.

Scrivere un test

1. **Crea una nuova classe all'interno del set di sorgenti di `test` .**
   Fare clic con il tasto destro del mouse sull'origine del test nella vista del progetto, selezionare `New` > `Java class` .
   Il modello di denominazione più utilizzato consiste nell'utilizzare il nome della classe che si testerà con `Test` aggiunto ad esso. Quindi `StringUtilities` diventa `StringUtilitiesTest` .

2. **Aggiungi l'annotazione `@RunWith`**
   L'annotazione `@RunWith` è necessaria per fare in modo che JUnit esegua i test che definiremo nella nostra classe di test. Il JUnit runner predefinito (per JUnit 4) è il `BlockJUnit4ClassRunner` ma invece di usare direttamente questo è più comodo usare l'alias `JUnit4` che è una scorciatoia per il runner JUnit predefinito.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

}
```

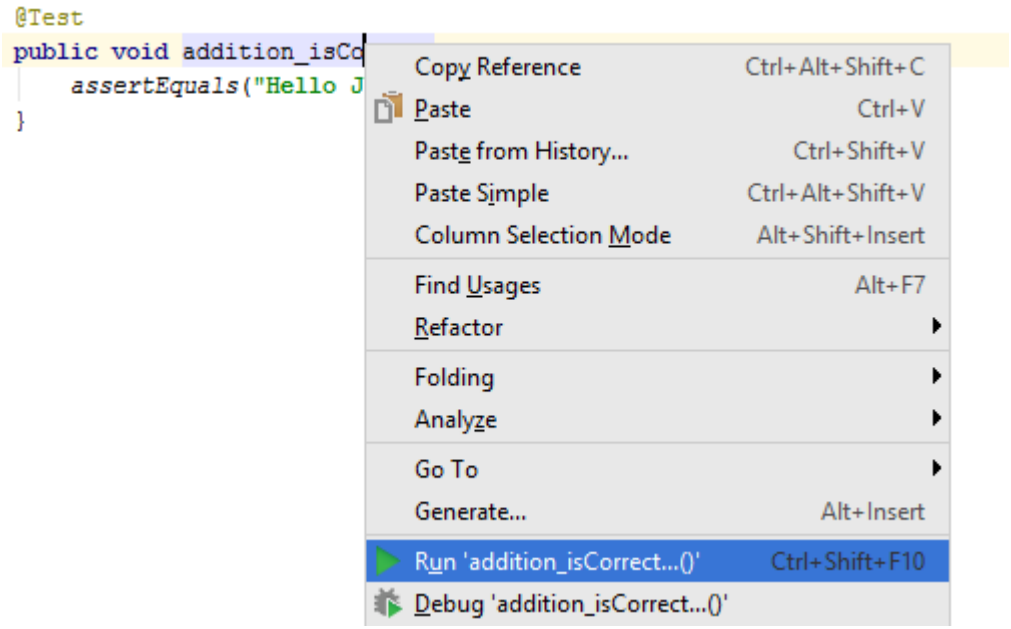
3. **Crea un test**
   Un test unitario è essenzialmente solo un metodo che, nella maggior parte dei casi, non dovrebbe fallire se eseguito. In altre parole, non dovrebbe fare eccezione. All'interno di un metodo di test troverai quasi sempre asserzioni che verificano se sono soddisfatte condizioni specifiche. Se un'asserzione fallisce, genera un'eccezione che causa il fallimento del metodo / test. Un metodo di prova è sempre annotato con l'annotazione `@Test` . Senza questa annotazione, JUnit non eseguirà automaticamente il test.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

    @Test
    public void addition_isCorrect() throws Exception {
        assertEquals("Hello JUnit", "Hello" + " " + "JUnit");
    }
}
```

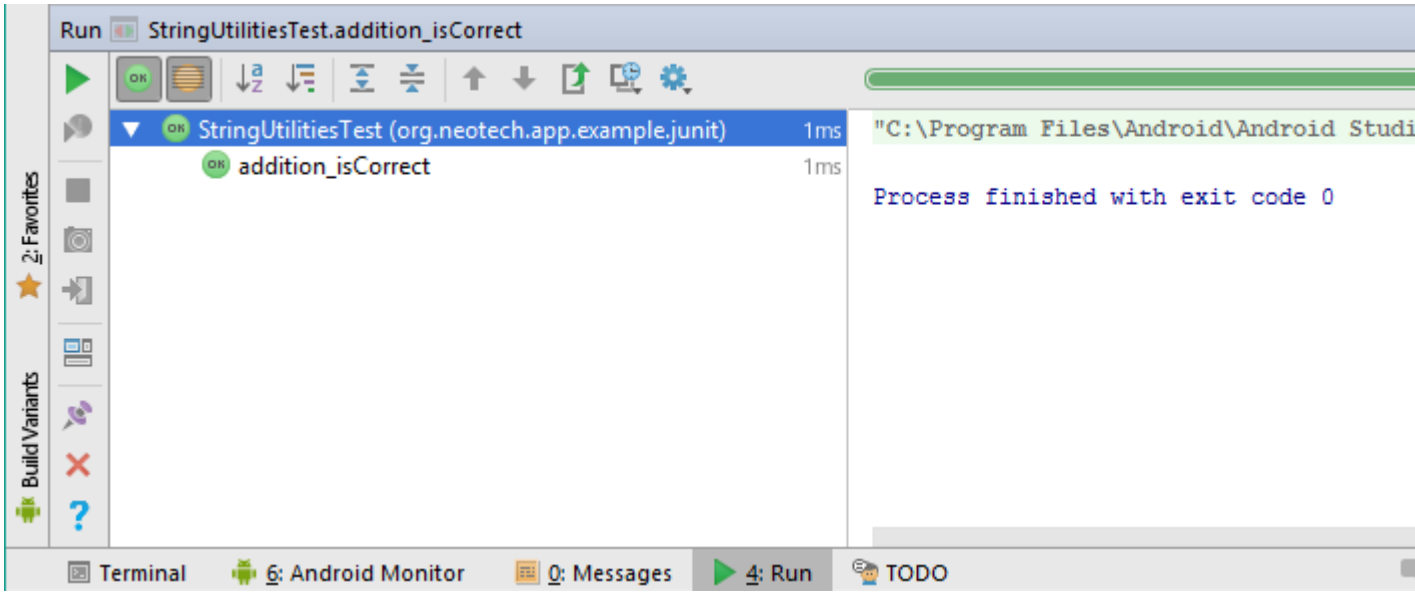
   Nota: a differenza del metodo di denominazione dei metodi standard, i nomi dei metodi di test dell'unità di misura spesso contengono caratteri di sottolineatura.

Esecuzione di un test

1. **Metodo**
   Per eseguire un singolo metodo di prova, è possibile fare clic con il tasto destro sul metodo e fare clic su `Run 'addition_isCorrect()'` o utilizzare la scorciatoia da tastiera `ctrl+shift+f10` .



Se tutto è impostato correttamente, JUnit avvia il metodo e dovresti vedere la seguente interfaccia in Android Studio:



2. **Classe**
   È inoltre possibile eseguire tutti i test definiti in una singola classe, facendo clic con il tasto destro sulla classe nella vista del progetto e facendo clic su `Run 'StringUtilitiesTest '` o utilizzare la scorciatoia da tastiera `ctrl+shift+f10` se si è selezionata la classe nella vista di progetto.

3. **Pacchetto (tutto)**
   Se non si desidera eseguire tutti i test definiti nel progetto o in un pacchetto, è sufficiente fare clic con il pulsante destro del mouse sul pacchetto e fare clic su `Run ...` proprio come si eseguiranno tutti i test definiti in una singola classe.

eccezioni

JUnit può anche essere usato per verificare se un metodo genera un'eccezione specifica per un

dato input.

In questo esempio testeremo se il seguente metodo genera un'eccezione se il formato booleano (input) non è riconosciuto / sconosciuto:

```
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);
    }
}
```

Aggiungendo il parametro `expected` all'annotazione `@Test` , è possibile definire quale eccezione dovrebbe essere generata. Il test unitario fallirà se questa eccezione non si verifica e riuscirà se l'eccezione è effettivamente lanciata:

```
@Test(expected = IllegalArgumentException.class)
public void parseBoolean_parsesInvalidFormat_throwsException(){
    StringUtilities.parseBoolean("Hello JUnit");
}
```

Funziona bene, tuttavia, ti limita a un solo caso di test all'interno del metodo. A volte potresti voler testare più casi con un singolo metodo. Una tecnica spesso utilizzata per superare questa limitazione è l'utilizzo dei blocchi `try-catch` e del metodo `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: alcune persone ritengono che sia una cattiva pratica testare più di un singolo caso all'interno di un test unitario.

Importazione statica

JUnit definisce alcuni metodi `assertEquals` almeno uno per ogni tipo primitivo e uno per gli oggetti è disponibile. Questi metodi sono di default non direttamente disponibili per chiamare e dovrebbero essere chiamati così: `Assert.assertEquals` . Ma poiché questi metodi vengono utilizzati così

spesso, le persone utilizzano quasi sempre un'importazione statica, in modo che il metodo possa essere utilizzato direttamente come se fosse parte della classe stessa.

Per aggiungere un'importazione statica per il metodo `assertEquals` , utilizzare la seguente dichiarazione di importazione:

```
import static org.junit.Assert.assertEquals;
```

È inoltre possibile importare statici tutti i metodi di assert, inclusi `assertArrayEquals` , `assertNotNull` e `assertFalse` ecc. Utilizzando la seguente importazione statica:

```
import static org.junit.Assert.*;
```

Senza importazione statica:

```
@Test
public void addition_isCorrect(){
    Assert.assertEquals(4 , 2 + 2);
}
```

Con l'importazione statica:

```
@Test
public void addition_isCorrect(){
    assertEquals(4 , 2 + 2);
}
```

Leggi Test delle unità in Android con JUnit online: https://riptutorial.com/it/android/topic/3205/test-delle-unita-in-android-con-junit

Capitolo 241: Text to Speech (TTS)

Examples

Base di sintesi vocale

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);
        }
    }
}
```

La lingua da pronunciare può essere impostata fornendo un `Locale` al metodo `setLanguage()` :

```
tts.setLanguage(Locale.CHINESE); // Chinese language
```

Il numero di lingue supportate varia tra i livelli Android. Il metodo `isLanguageAvailable()` può essere usato per verificare se una determinata lingua è supportata:

```
tts.isLanguageAvailable(Locale.CHINESE);
```

Il livello del tono di voce può essere impostato usando il metodo `setPitch()` . Per impostazione predefinita, il valore di altezza è 1.0. Utilizzare valori inferiori a 1.0 per ridurre il livello di intonazione o valori maggiori di 1.0 per aumentare il livello di intonazione:

```
tts.setPitch(0.6);
```

La velocità vocale può essere impostata utilizzando `setSpeechRate()` . La frequenza vocale predefinita è 1.0. La frequenza vocale può essere raddoppiata impostandola su 2.0 o resa metà

impostandola su 0.5:

```
tts.setSpeechRate(2.0);
```

Implementazione di TextToSpeech attraverso le API

Implementazione osservabile a freddo, emette vero quando il motore TTS finisce di parlare, inizia a parlare quando è iscritto. Si noti che il livello 21 dell'API introduce un modo diverso di eseguire il parlato:

```
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;
}
```

}

Leggi Text to Speech (TTS) online: https://riptutorial.com/it/android/topic/3381/text-to-speech--tts-

Capitolo 242: TextInputLayout

introduzione

È stato introdotto TextInputLayout per visualizzare l'etichetta mobile su EditText. EditText deve essere inserito da TextInputLayout per visualizzare l'etichetta mobile.

Osservazioni

`TextInputLayout` è un layout che racchiude un `EditText` (o un discendente) per mostrare un'etichetta mobile quando il suggerimento è nascosto a causa dell'inserimento del testo da parte dell'utente. Addizionalmente, `TextInputLayout` consente di visualizzare un messaggio di errore sotto `EditText` .

Assicurati che la seguente dipendenza venga aggiunta al file `build.gradle` dell'app in dipendenze:

```
compile 'com.android.support:design:25.3.1'
```

Examples

Utilizzo di base

È l'utilizzo di base di `TextInputLayout` .
Assicurati di aggiungere la dipendenza nel file `build.gradle` come descritto nella sezione commenti.

Esempio:

```
<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>
```

Gestione degli errori

È possibile utilizzare `TextInputLayout` per visualizzare i messaggi di errore in base alle linee guida sulla progettazione dei materiali utilizzando i metodi `setError` e `setErrorEnabled` .

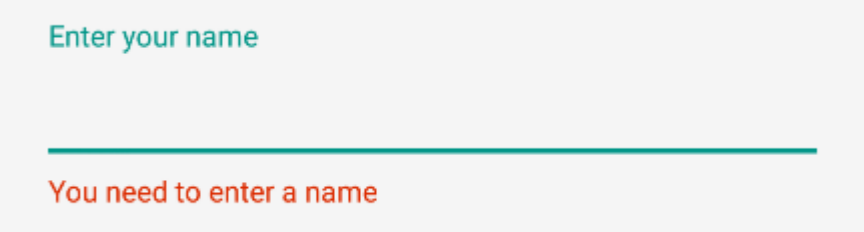
Per mostrare l'errore sotto l'uso di EditText:

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
```

```
til.setErrorEnabled(true);
til.setError("You need to enter a name");
```

Per abilitare l'errore in `TextInputLayout` puoi utilizzare `app:errorEnabled="true"` in xml o `til.setErrorEnabled(true);` come mostrato sopra.

Otterrai:



Aggiungere il conteggio dei caratteri

TextInputLayout ha un contatore di caratteri per un EditText definito al suo interno.
Il contatore verrà visualizzato sotto EditText.

Basta usare i `setCounterEnabled()` e `setCounterMaxLength` :

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setCounterEnabled(true);
til.setCounterMaxLength(15);
```

oppure l' `app:counterEnabled` e `app:counterMaxLength` attributi in xml.

```
<android.support.design.widget.TextInputLayout
    app:counterEnabled="true"
    app:counterMaxLength="15">

    <EditText/>

</android.support.design.widget.TextInputLayout>
```

Commuta password

Con un tipo di password di input, puoi anche abilitare un'icona che può mostrare o nascondere l'intero testo utilizzando l'attributo `passwordToggleEnabled` .

Puoi anche personalizzare lo stesso valore predefinito utilizzando questi attributi:

- `passwordToggleDrawable` : per cambiare l'icona occhio predefinita
- `passwordToggleTint` : per applicare una tinta alla visibilità della password, è possibile disegnare.
- `passwordToggleTintMode` : per specificare la modalità di fusione utilizzata per applicare la tinta di sfondo.

Esempio:

```
<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` è un `EditText` con una correzione aggiuntiva per visualizzare un suggerimento nell'IME durante la modalità 'estrai' .

La **modalità Estrai** è la modalità a cui passa l'editor della tastiera quando si fa clic su un EditText quando lo spazio è troppo piccolo (ad esempio l'orizzontale su uno smartphone).
In questo caso, usando un `EditText` mentre stai modificando il testo puoi vedere che l'IME non ti dà un suggerimento su ciò che stai modificando

`TextInputEditText` corregge questo problema fornendo testo di suggerimento mentre l'IME del dispositivo dell'utente si trova in modalità Estrai.

Esempio:

```
<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>
```

Personalizzazione dell'aspetto di TextInputLayout

È possibile personalizzare l'aspetto di `TextInputLayout` e del relativo `EditText` incorporato definendo gli stili personalizzati nel proprio `styles.xml` . Gli stili definiti possono essere aggiunti come stili o temi al tuo `TextInputLayout` .

Esempio per personalizzare l'aspetto del suggerimento:

`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>
```

Per applicare lo stile, aggiornare TextInputLayout e EditText come segue

```
<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>
```

Esempio per personalizzare il colore dell'accento di `TextInputLayout` . Il colore dell'accento influisce sul colore della linea di base di `EditText` e sul colore del testo per il testo suggerimento mobile:

`styles.xml` :

```
<style name="TextInputLayoutWithPrimaryColor" parent="Widget.Design.TextInputLayout">
    <item name="colorAccent">@color/primary</item>
</style>
```

file di layout:

```
<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>
```

Leggi TextInputLayout online: https://riptutorial.com/it/android/topic/5652/textinputlayout

Capitolo 243: TextView

introduzione

Tutto ciò che riguarda la personalizzazione di TextView in Android SDK

Sintassi

- TextView (contesto di contesto)
- (TextView) findViewById (int id)
- void setText (int resid)
- void setText (CharSequence text) // Puoi usare String come argomento

Osservazioni

Prova ad usarlo nel design xml o programmaticamente.

Examples

Textview con diversi testi

Puoi archiviare diversi Textsizes all'interno di una Textview con Span

```
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)
```

Personalizzazione di 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);
    }
}
```

Disposizione:

```
<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>
```

Attar:

```
<?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 programmatico:

```
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 spannabile

Un `TextView` utilizzabile in Android può essere utilizzato per evidenziare una particolare porzione di testo con un colore, uno stile, una dimensione e / o un evento di clic diversi in un singolo widget `TextView` .

Considera che hai definito un `TextView` come segue:

```
TextView textview=findViewById(R.id.textview);
```

Quindi puoi applicare diverse evidenziazioni ad esso come mostrato di seguito:

- **Colore spendibile:** per impostare un colore diverso su una porzione di testo, è possibile utilizzare `ForegroundColorSpan` , come mostrato nell'esempio seguente:

  ```
  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 );
  ```

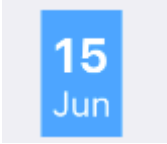
  Uscita creata dal codice sopra:

- **Carattere spannabile:** per impostare una diversa dimensione del carattere su una porzione di testo, è possibile utilizzare un `RelativeSizeSpan` , come mostrato nell'esempio seguente:

  ```
  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 );
  ```

  Uscita creata dal codice sopra:



- **Carattere tipografico:** per impostare un carattere tipografico diverso su una porzione di testo, è possibile utilizzare un `TypefaceSpan` personalizzato, come mostrato nell'esempio seguente:

```
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 );
```

Tuttavia, per rendere funzionante il codice precedente, la classe `CustomTypefaceSpan` deve essere derivata dalla classe `TypefaceSpan` . Questo può essere fatto come segue:

```
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 immagine

Android consente ai programmatori di posizionare le immagini su tutti e quattro gli angoli di un `TextView` . Ad esempio, se stai creando un campo con `TextView` e allo stesso tempo vuoi mostrare che il campo è modificabile, in genere gli sviluppatori posizioneranno un'icona di modifica vicino a quel campo. Android ci offre un'interessante opzione chiamata **drawable composto** per 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" />
```

Puoi impostare il drawable su qualsiasi lato di `TextView` come segue:

```
android:drawableLeft="@drawable/edit"
android:drawableRight="@drawable/edit"
android:drawableTop="@drawable/edit"
android:drawableBottom="@drawable/edit"
```

L'impostazione del drawable può essere ottenuta anche programmaticamente nel modo seguente:

```
yourTextView.setCompoundDrawables(leftDrawable, rightDrawable, topDrawable, bottomDrawable);
```

L'impostazione di uno qualsiasi dei parametri `setCompoundDrawables()` a `setCompoundDrawables()` su `null` rimuoverà l'icona dal lato corrispondente di `TextView` .

Barrato TextView

Barrato attraverso l'intero testo

```
String sampleText = "This is a test strike";
textView.setPaintFlags(tv.getPaintFlags()| Paint.STRIKE_THRU_TEXT_FLAG);
textView.setText(sampleText);
```

Uscita: ~~questo è un colpo di prova~~

Barrato solo alcune parti del testo

```
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);
```

Uscita: ~~questo~~ è un colpo di prova

Personalizzazione di temi e stili

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>
```

Rendi RelativeSizeSpan allineato in alto

Al fine di rendere `RelativeSizeSpan` allineato verso l'alto, è possibile derivare una classe personalizzata dalla classe `SuperscriptSpan` . Nell'esempio seguente, la classe derivata è denominata `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" />
```

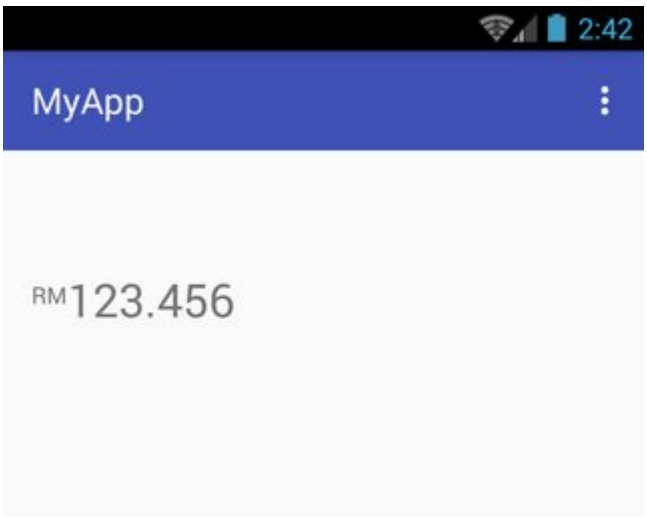
MainActivity.java:

```
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 );
    }
}
```

Schermata di riferimento:



Pinchzoom su 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;
    }
}
```

Single TextView con due colori diversi

Il testo colorato può essere creato passando il testo e il nome del colore del carattere alla seguente funzione:

```
private String getColoredSpanned(String text, String color) {
    String input = "<font color=" + color + ">" + text + "</font>";
    return input;
}
```

Il testo colorato può quindi essere impostato su `TextView` (o anche su un `Button` , `EditText` , ecc.) Utilizzando il codice di esempio seguente.

Innanzitutto, definire un `TextView` come segue:

```
TextView txtView = (TextView)findViewById(R.id.txtView);
```

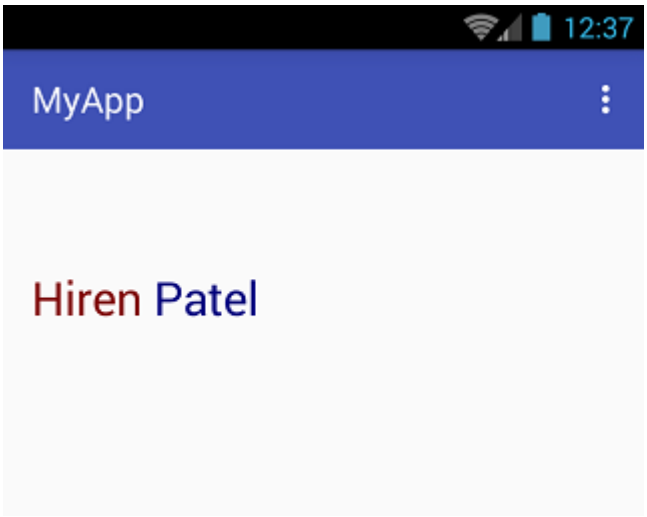
Quindi, crea un testo di colore diverso e assegnalo alle stringhe:

```
String name = getColoredSpanned("Hiren", "#800000");
String surName = getColoredSpanned("Patel","#000080");
```

Infine, imposta le due stringhe di colore diverso su `TextView` :

```
txtView.setText(Html.fromHtml(name+" "+surName));
```

Schermata di riferimento:



Leggi TextView online: https://riptutorial.com/it/android/topic/4212/textview

Capitolo 244: Time Utils

Examples

Converti formato data in millisecondi

Per convertire la tua data in formato gg / MM / aaaa in millisecondi chiami questa funzione con dati come 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();
}
```

Questo metodo converte millisecondi in data Formato timbro temporale:

```
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;
}
```

Questo metodo consente di convertire determinati giorni, mesi e anni in millisecondi. Sarà molto utile quando si usa `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();
}
```

Restituirà millisecondi dalla data

```
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;
}
```

Restituirà la data corrente

```
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 Fornisce numeri di supporto per il formato data Pattern per data

Per controllare entro un periodo

Questo esempio aiuterà a verificare che il tempo specificato sia entro un periodo o meno.

Per controllare l'ora è possibile, possiamo usare la classe ***DateUtils***

```
boolean isToday = DateUtils.isToday(timeInMillis);
```

Per controllare l'ora è entro una settimana,

```
private static boolean isWithinWeek(final long millis) {
    return System.currentTimeMillis() - millis <= (DateUtils.WEEK_IN_MILLIS -
DateUtils.DAY_IN_MILLIS);
}
```

Per controllare l'ora è entro un anno,

```
private static boolean isWithinYear(final long millis) {
    return System.currentTimeMillis() - millis <= DateUtils.YEAR_IN_MILLIS;
}
```

Per controllare l'ora è entro un numero di giorno del giorno incluso oggi,

```
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 è **android.text.format.DateUtils**

GetCurrentRealTime

Questo calcola il tempo corrente del dispositivo e aggiunge / sottrae la differenza tra tempo reale e 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;
}
```

ottenere fuso orario basato su UTC.

```
public static TimeZone getUTCTimeZone() {
    return TimeZone.getTimeZone("GMT");
}
```

Leggi Time Utils online: https://riptutorial.com/it/android/topic/7138/time-utils

Capitolo 245: Tocca Eventi

Examples

Come variare tra gli eventi di tocco del gruppo di visualizzazione figlio e padre

1. Il `onTouchEvents()` per i gruppi di viste nidificate può essere gestito dal `boolean` onInterceptTouchEvent .

Il valore predefinito per `OnInterceptTouchEvent` è false.

`onTouchEvent` del genitore viene ricevuto prima del figlio. Se `OnInterceptTouchEvent` restituisce false, invia l'evento di movimento lungo la catena al gestore `OnTouchEvent` del bambino. Se restituisce true, i genitori gestiranno l'evento touch.

Tuttavia ci possono essere casi in cui vogliamo che alcuni elementi figlio gestiscano `OnTouchEvent` e alcuni siano gestiti dalla vista genitore (o possibilmente il genitore del genitore).

Questo può essere gestito in più di un modo.

2. Un modo in cui un elemento figlio può essere protetto da `OnInterceptTouchEvent` del genitore è implementando requestDisallowInterceptTouchEvent .

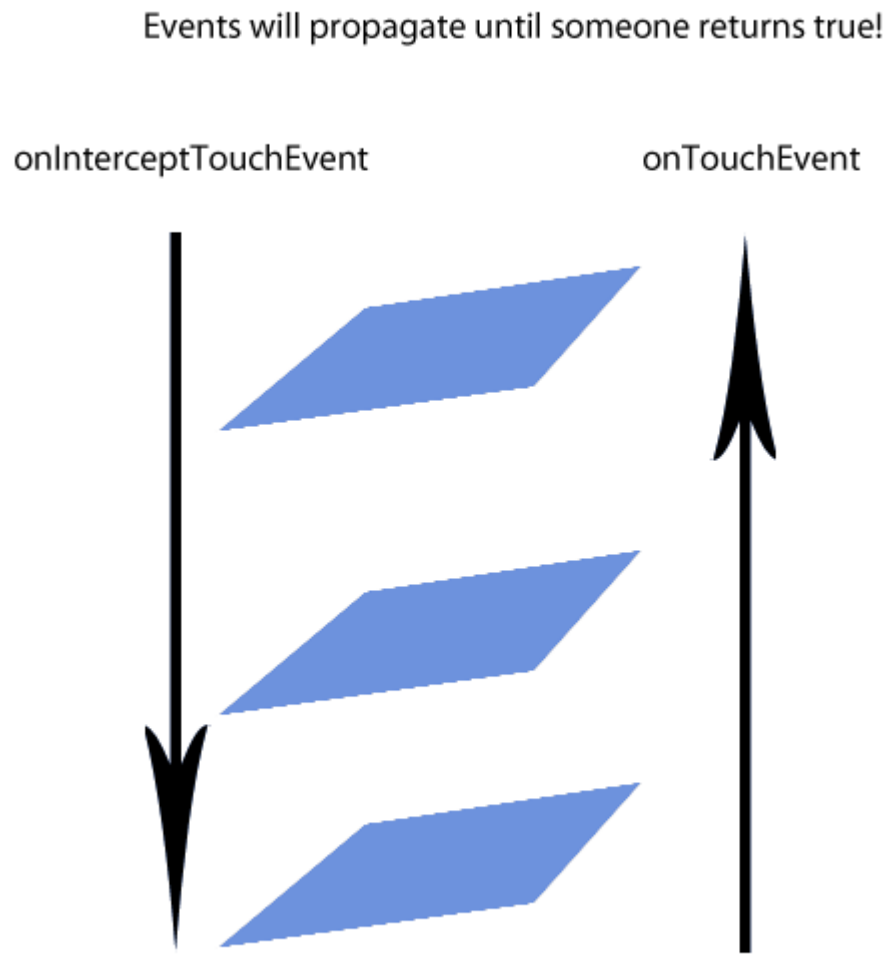
   public void requestDisallowInterceptTouchEvent (boolean disallowIntercept)

Ciò impedisce a qualsiasi delle viste principali di gestire `OnTouchEvent` per questo elemento, se l'elemento ha abilitati i gestori di eventi.

3.

Se `OnInterceptTouchEvent` è falso, verrà valutato l'elemento `OnTouchEvent` dell'elemento `OnTouchEvent` . Se si dispone di metodi all'interno degli elementi figlio che gestiscono i vari eventi di tocco, qualsiasi gestore di eventi correlato disabilitato restituirà OnTouchEvent al genitore.

Questa risposta:
Una visualizzazione di come passa la propagazione degli eventi tattili:
`parent -> child|parent -> child|parent -> child views.`



Per gentile concessione da qui

4. Un altro modo restituisce valori diversi da `OnInterceptTouchEvent` per il genitore.

Questo esempio è tratto da Gestione degli eventi di tocco in un ViewGroup e dimostra come intercettare l' `OnTouchEvent` del bambino quando l'utente sta scorrendo.

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;
}
```

Questo è un codice dallo stesso link che mostra come creare i parametri del rettangolo attorno al tuo 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);
}
```

Leggi Tocca Eventi online: https://riptutorial.com/it/android/topic/7167/tocca-eventi

Capitolo 246: Traccia audio

Examples

Genera tono di una frequenza specifica

Per riprodurre un suono con un tono specifico, dobbiamo prima creare un suono sinusoidale. Questo viene fatto nel modo seguente.

```
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
}
```

Ora dobbiamo configurare AudioTrack per giocare secondo il buffer generato. È fatto nel modo seguente

```
AudioTrack audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC,
                    sampleRate, AudioFormat.CHANNEL_OUT_MONO,
                    AudioFormat.ENCODING_PCM_16BIT, buffer.length,
                    AudioTrack.MODE_STATIC);
```

Scrivi il buffer generato e riproduci la traccia

```
audioTrack.write(buffer, 0, buffer.length);
audioTrack.play();
```

Spero che questo ti aiuti :)

Leggi Traccia audio online: https://riptutorial.com/it/android/topic/9155/traccia-audio

Capitolo 247: TransitionDrawable

Examples

Aggiungi transizione o Dissolvenza incrociata tra due immagini.

Passaggio 1: crea una transizione drawable in XML

Salva questo file `transition.xml` nella cartella `res/drawable` del tuo progetto.

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/image1"/>
    <item android:drawable="@drawable/image2"/>
</transition>
```

Image1 e image2 sono le due immagini che vogliamo passare e dovrebbero essere inserite anche nella cartella `res/drawable` .

Passaggio 2: aggiungi il codice per ImageView nel tuo layout XML per visualizzare il precedente drawable.

```
<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>
```

Passaggio 3: accedi alla transizione XML selezionabile nel metodo onCreate () della tua attività e avvia la transizione nell'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);
        }
    });
}
```

Animare le viste con il colore di sfondo (cambia colore) 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);
}
```

Leggi TransitionDrawable online: https://riptutorial.com/it/android/topic/6088/transitiondrawable

Capitolo 248: Transizioni di elementi condivise

introduzione

Qui puoi trovare esempi di transizione tra `Activities` o `Fragments` utilizzando un elemento condiviso. Un esempio di questo comportamento è l'app Google Play Store che traduce l'icona di un'app dall'elenco alla visualizzazione dettagli dell'app.

Sintassi

- transaction.addSharedElement (sharedElementView, "targetTransitionName");
- fragment.setSharedElementEnterTransition (new CustomTransaction ());

Examples

Transizione elemento condivisa tra due frammenti

In questo esempio, uno dei due diversi `ImageViews` dovrebbe essere tradotto da `ChooserFragment` a `DetailFragment` .

Nel layout `ChooserFragment` sono necessari gli attributi unique `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" />
```

Nel `ChooserFragments` di classe, abbiamo bisogno di passare la `View` che è stato cliccato e un ID al genitore `Activity` wich sta gestendo la sostituzione dei frammenti (abbiamo bisogno del ID di sapere quale immagine risorsa da mostrare nella `DetailFragment`). Come passare le informazioni a un'attività principale in dettaglio è sicuramente coperto in un'altra documentazione.

```
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);
        }
     }
});
```

Nel `DetailFragment` , `ImageView` dell'elemento condivisa deve anche unica `transitionName` attributo.

```
<ImageView
    android:id="@+id/image_shared"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:transitionName="sharedImage" />
```

Nel metodo `onCreateView()` di `DetailFragment` , dobbiamo decidere quale risorsa immagine deve essere mostrata (se non lo facciamo, l'elemento condiviso sparirà dopo la transizione).

```
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;
}
```

L' `Activity` principale sta ricevendo i callback e gestisce la sostituzione dei frammenti.

```
@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();
}
```

Per non dimenticare - la `Transition` stessa. Questo esempio sposta e ridimensiona l'elemento condiviso.

```
@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class DetailsTransition extends TransitionSet {

    public DetailsTransition() {
        setOrdering(ORDERING_TOGETHER);
        addTransition(new ChangeBounds()).
            addTransition(new ChangeTransform()).
            addTransition(new ChangeImageTransform());
    }

}
```

Leggi Transizioni di elementi condivise online:
https://riptutorial.com/it/android/topic/8933/transizioni-di-elementi-condivise

Capitolo 249: UI Lifecycle

Examples

Salvataggio dei dati sul taglio della 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);
    }
}
```

Spiegazione

Quindi, cosa sta succedendo qui?

Il sistema Android cercherà sempre di cancellare più memoria possibile. Quindi, se la tua attività è in background e un'altra attività in primo piano richiede la sua condivisione, il sistema Android chiamerà `onTrimMemory()` sulla tua attività.

Ma ciò non significa che tutte le tue proprietà dovrebbero svanire. Quello che dovresti fare è salvarli in un oggetto Bundle. Gli oggetti bundle sono gestiti molto meglio dalla memoria. All'interno di un fascio ogni oggetto è identificato da una sequenza di testo univoca - nell'esempio sopra la variabile del valore intero `mArg` è tenuta sotto il nome di riferimento `EXAMPLE_ARG` . E quando l'attività viene ricreata estrai i tuoi vecchi valori dall'oggetto Bundle invece di ricrearli da zero

Leggi UI Lifecycle online: https://riptutorial.com/it/android/topic/3440/ui-lifecycle

Capitolo 250: Universal Image Loader

Osservazioni

Esempi di URI accettabili:

```
"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

Inizializza Universal Image Loader

1. Aggiungi la seguente dipendenza al file *build.gradle* :

   ```
   compile 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
   ```

2. Aggiungi le seguenti autorizzazioni al file *AndroidManifest.xml* :

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

3. Inizializza Universal Image Loader. Questo deve essere fatto prima del primo utilizzo:

   ```
   ImageLoaderConfiguration config = new ImageLoaderConfiguration.Builder(this)
       // ...
       .build();
   ImageLoader.getInstance().init(config);
   ```

   Le opzioni di configurazione complete possono essere trovate qui .

Utilizzo di base

1. Carica un'immagine, decodificala in una bitmap e visualizza la bitmap in un `ImageView` (o in qualsiasi altra vista che implementa l'interfaccia `ImageAware`):

   ```
   ImageLoader.getInstance().displayImage(imageUri, imageView);
   ```

2. Carica un'immagine, decodificala in una bitmap e riporta la bitmap a un callback:

   ```
   ImageLoader.getInstance().loadImage(imageUri, new SimpleImageLoadingListener() {
   ```

```
    @Override
    public void onLoadingComplete(String imageUri, View view, Bitmap loadedImage) {
        // Do whatever you want with the bitmap.
    }
});
```

3. Carica un'immagine, decodificala in una bitmap e restituisci la bitmap in modo sincrono:

```
Bitmap bmp = ImageLoader.getInstance().loadImageSync(imageUri);
```

Leggi Universal Image Loader online: https://riptutorial.com/it/android/topic/2760/universal-image-loader

Capitolo 251: Unzip File in Android

Examples

Unzip file

```
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;}
```

Leggi Unzip File in Android online: https://riptutorial.com/it/android/topic/3927/unzip-file-in-android

Capitolo 252: URL di richiamata

Examples

Esempio di URL di callback con Instagram OAuth

Uno dei casi d'uso degli *URL* di *callback* è OAuth. Facciamo questo con un login Instagram: se l'utente inserisce le proprie credenziali e fa clic sul pulsante *Accedi* , Instagram convaliderà le credenziali e restituirà un `access_token` . Abbiamo bisogno di `access_token` nella nostra app.

Affinché la nostra app sia in grado di ascoltare tali collegamenti, è necessario aggiungere un URL di richiamata alla nostra `Activity` . Possiamo farlo aggiungendo un `<intent-filter/>` alla nostra `Activity` , che reagirà a tale URL di callback. Supponiamo che il nostro URL di richiamata sia `appSchema://appName.com` . Poi bisogna aggiungere le seguenti righe al tuo desiderato `Activity` nel file *Manifest.xml:*

```
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.BROWSABLE"/>
<data android:host="appName.com" android:scheme="appSchema"/>
```

Spiegazione delle righe sopra:

- `<category android:name="android.intent.category.BROWSABLE"/>` rende l'attività di destinazione autorizzata a essere avviata da un browser Web per visualizzare i dati a cui fa riferimento un collegamento.
- `<data android:host="appName.com" android:scheme="appSchema"/>` specifica il nostro schema e host del nostro URL di callback.
- Tutte insieme, queste linee faranno aprire l' `Activity` specifica ogni volta che l'URL di richiamata viene richiamato in un browser.

Ora, per ottenere il contenuto dell'URL nella tua `Activity` , devi sovrascrivere il metodo `onResume()` come segue:

```
@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.
}
```

Ora hai recuperato `access_token` da Instagram, utilizzato in vari endpoint API di Instagram.

Leggi URL di richiamata online: https://riptutorial.com/it/android/topic/4790/url-di-richiamata

Capitolo 253: VectorDrawable e AnimatedVectorDrawable

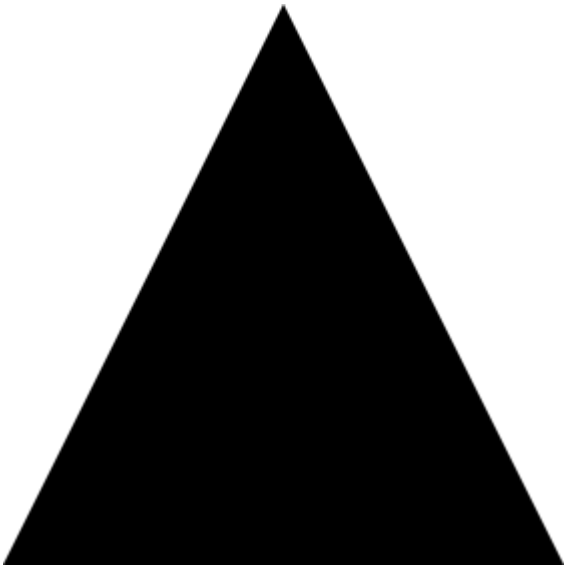
Examples

Basic VectorDrawable

Un `VectorDrawable` deve consistere in almeno un tag `<path>` definisce 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>
```

Questo produrrebbe un triangolo nero:



utilizzando

Un `<clip-path>` definisce una forma che funge da finestra, consentendo solo a parti di un `<path>` di mostrare se si trovano all'interno della forma `<clip-path>` e di tagliare il 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>
```

In questo caso il `<path>` produce un triangolo nero, ma il `<clip-path>` definisce una forma quadrata più piccola, consentendo solo a una parte del triangolo di mostrare:

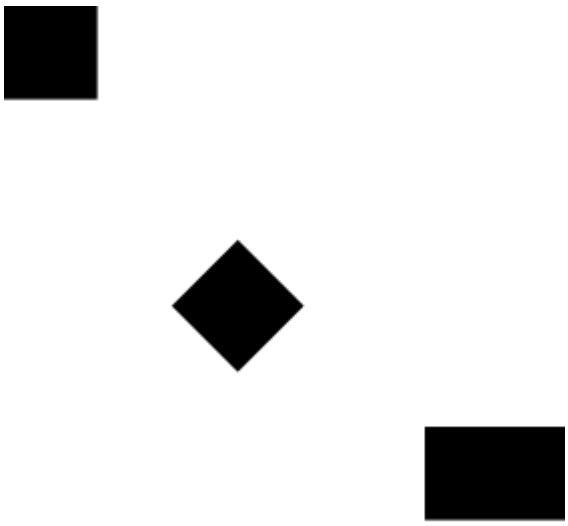
tag

Un tag `<group>` consente di ridimensionare, ruotare e posizionare uno o più elementi di un `VectorDrawable` da regolare:

```
<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>
```

Il codice di esempio sopra contiene tre tag `<path>` identici, tutti che descrivono i quadrati neri. Il primo quadrato non è corretto. Il secondo quadrato è avvolto in un tag `<group>` che lo sposta e lo ruota di 45 °. Il terzo quadrato è avvolto in un tag `<group>` che lo sposta e lo allunga orizzontalmente del 50%. Il risultato è il seguente:



Un tag `<group>` può contenere più tag `<path>` e `<clip-path>` . Può anche contenere un altro `<group>` .

Basic AnimatedVectorDrawable

Un `AnimatedVectorDrawable` richiede almeno 3 componenti:

- Un `VectorDrawable` che verrà manipolato
- Un `objectAnimator` che definisce quale proprietà modificare e come
- `AnimatedVectorDrawable` stesso che collega `objectAnimator` a `VectorDrawable` per creare l'animazione

Di seguito viene creato un triangolo che passa dal colore nero al rosso.

The `VectorDrawable` , filename: `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>
```

L' `objectAnimator` , nomefile: `color_change_animator.xml`

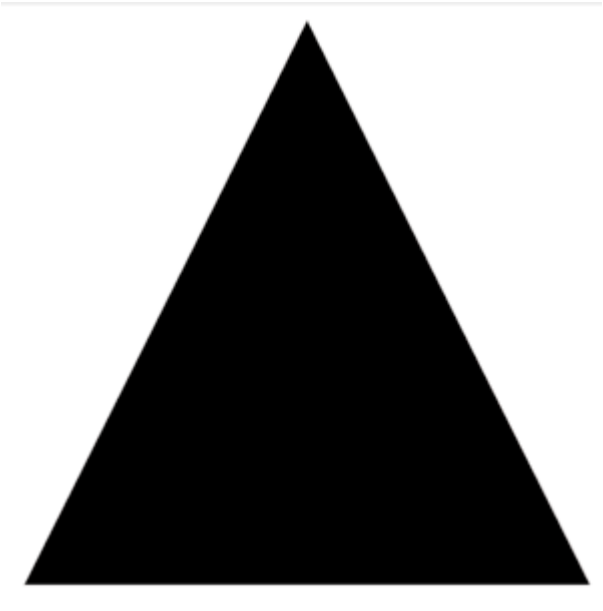
```
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:propertyName="fillColor"
    android:duration="2000"
    android:repeatCount="infinite"
    android:valueFrom="@android:color/black"
    android:valueTo="@android:color/holo_red_light"/>
```

`AnimatedVectorDrawable` , filename: `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>
```

Nota che `<target>` specifica `android:name="triangle"` che corrisponde al `<path>` in `VectorDrawable` . Un `VectorDrawable` può contenere più elementi e `android:name` proprietà `android:name` viene utilizzata per definire quale elemento viene scelto come target.

Risultato:



Uso di tratti

L'uso del tratto SVG semplifica la creazione di un vettore disegnato con una lunghezza del tratto unificata, come da linee guida sulla progettazione dei materiali :

> I pesi di colpo coerenti sono fondamentali per unificare la famiglia di icone del sistema

in generale. Mantieni una larghezza di 2 dpi per tutte le istanze di tratto, comprese le curve, gli angoli e i tratti interni ed esterni.

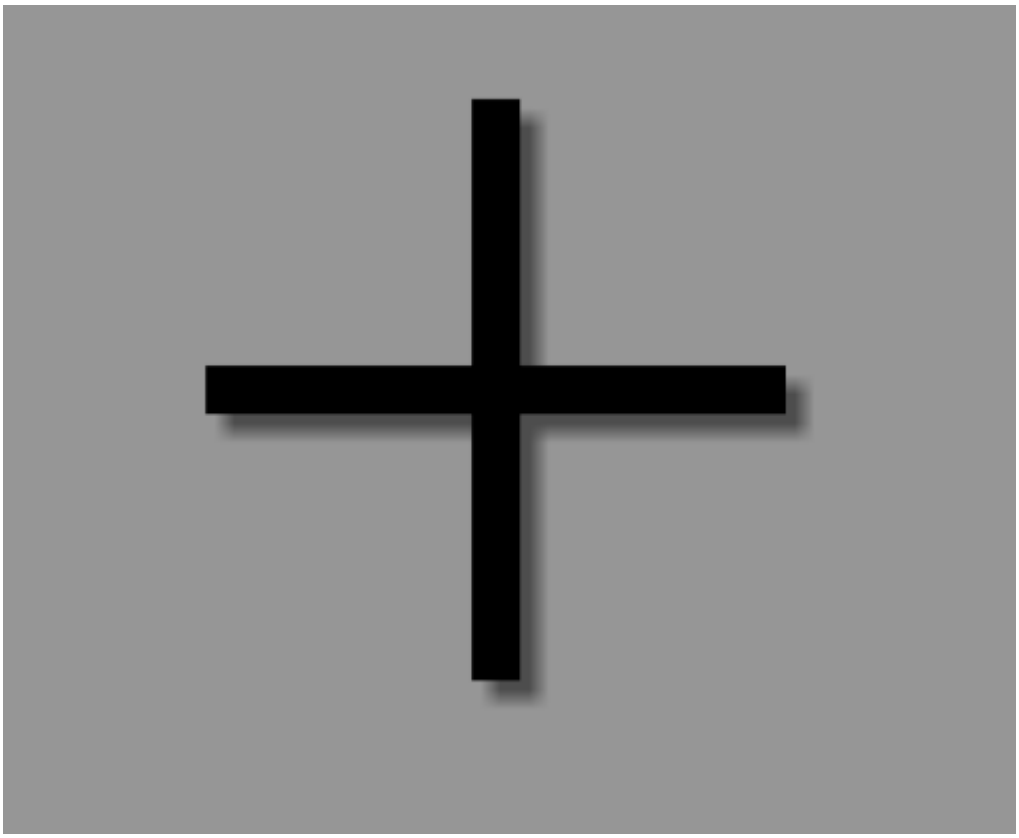
Quindi, ad esempio, questo è il modo in cui crei un segno "più" usando i tratti:

```
<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` definisce il colore del tratto.
- `strokeWidth` definisce la larghezza (in dp) del tratto (in questo caso 2dp, come suggerito dalle linee guida).
- `pathData` è dove descriviamo la nostra immagine SVG:
- `M12,0` sposta il "cursore" nella posizione 12,0
- `V24` crea una linea verticale nella posizione 12, 24

ecc., consultare la documentazione SVG e questo utile tutorial "SVG Path" di w3schools per ulteriori informazioni sui comandi specifici del percorso.

Di conseguenza, abbiamo ottenuto questo segno più senza fronzoli:



Ciò è **particolarmente utile** per creare un oggetto `AnimatedVectorDrawable` , dal momento che ora si sta lavorando con un singolo tratto con una lunghezza unificata, invece di un percorso altrimenti complicato.

Compatibilità vettoriale tramite AppCompat

Alcuni prerequisiti nel `build.gradle` per i vettori funzionano fino all'API 7 per VectorDrawables e API 13 per AnimatedVectorDrawables (con alcune avvertenze attualmente):

```
//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'
}
```

Nel tuo `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" />
```

Leggi VectorDrawable e AnimatedVectorDrawable online:
https://riptutorial.com/it/android/topic/1627/vectordrawable-e-animatedvectordrawable

Capitolo 254: Verifica la connettività Internet

introduzione

Questo metodo è utilizzato per verificare se il Wi-Fi è connesso o meno.

Sintassi

- isNetworkAvailable (): per verificare se Internet è disponibile sul dispositivo

Parametri

Parametro	Dettaglio
Contesto	Un riferimento al contesto delle attività

Osservazioni

Se connesso a Internet, il metodo restituirà true o false.

Examples

Verifica se il dispositivo è connesso a Internet

Aggiungi le autorizzazioni di rete richieste al file manifest dell'applicazione:

```
<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;
}
```

Come controllare la forza della rete in 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
            }
        }
```

Come controllare la forza della rete

Per verificare la forza esatta in decibel usa questo-

```
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
        }
    }
```

Per verificare il tipo di rete usa questa classe-

```
 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 "";
        }
    }

}
```

Leggi Verifica la connettività Internet online: https://riptutorial.com/it/android/topic/3918/verifica-la-connettivita-internet

Capitolo 255: Versioni Android

Osservazioni

Nome	Versione Android	Data di rilascio	API-livello	Build.VERSION_CODES
Torta d'angelo (alfa)	1.0	23 settembre 2008	1	BASE
Battenberg (Beta)	1.1	9 febbraio 2009	2	BASE_1_1
Cupcake	1.5	30 aprile 2009	3	CUPCAKE
Ciambella	1.6	15 settembre 2009	4	CIAMBELLA
pasticcino	2.0	26 ottobre 2009	5	ECLAIR
	2.0.1	3 dicembre 2009	6	ECLAIR_0_1
	2.1	12 gennaio 2010	7	ECLAIR_MR1
Froyo	2.2	20 maggio 2010	8	FROYO
Pan di zenzero	2.3	6 dicembre 2010	9	PAN DI ZENZERO
	2.3.3	9 febbraio 2011	10	GINGERBREAD_MR1
Favo	3.0	22 febbraio 2011	11	FAVO
	3.1	10 maggio 2011	12	HONEYCOMB_MR2
	3.2	15 luglio 2011	13	HONEYCOMB_MR1
Panino gelato	4.0	19 ottobre	14	PANINO GELATO

Nome	Versione Android	Data di rilascio	API-livello	Build.VERSION_CODES
		2011		
	4.0.3	16 dicembre 2011	15	ICE_CREAM_SANDWICH_MR1
Jelly Bean	4.1	9 luglio 2012	16	JELLY BEAN
	4.2	13 novembre 2012	17	JELLY_BEAN_MR1
	4.3	24 luglio 2013	18	JELLY_BEAN_MR2
KitKat	4.4	31 ottobre 2013	19	KITKAT
		25 luglio 2014	20	KITKAT_WATCH
Lecca-lecca	5.0	17 ottobre 2014	21	LECCA-LECCA
	5.1	9 marzo 2015	22	LOLLIPOP_MR1
Marshmallow	6.0	5 ottobre 2015	23	M
Torrone	7.0	22 agosto 2016	24	N
	7.1.1	5 dicembre 2016	25	N_MR1

Examples

Controllo della versione Android sul dispositivo in fase di esecuzione

`Build.VERSION_CODES` è un'enumerazione dei codici di versione SDK attualmente noti.

Per eseguire in modo condizionale il codice in base alla versione Android del dispositivo, utilizzare l'annotazione `TargetApi` per evitare errori Lint e controllare la versione di build prima di eseguire il codice specifico per il livello API.

Ecco un esempio di come utilizzare una classe introdotta in API-23, in un progetto che supporta livelli API inferiori 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);
        //.....................
    }
}
```

Leggi Versioni Android online: https://riptutorial.com/it/android/topic/3264/versioni-android

Capitolo 256: Versioni di Project SDK

introduzione

Un'applicazione Android deve essere eseguita su tutti i tipi di dispositivi. Ogni dispositivo può avere una versione diversa su Android in esecuzione su di esso.

Ora, ogni versione di Android potrebbe non supportare tutte le funzionalità richieste dalla tua app, quindi durante la creazione di un'app è necessario tenere a mente la versione minima e massima di Android.

Parametri

Parametro	Dettagli
Versione SDK	La versione dell'SDK per ciascun campo è l'intero livello dell'SDK API del rilascio di Android. Ad esempio, Froyo (Android 2.2) corrisponde al livello API 8. Questi numeri interi sono definiti anche in `Build.VERSION_CODES` .

Osservazioni

Esistono quattro versioni SDK rilevanti in ogni progetto:

- `targetSdkVersion` è l'ultima versione di Android che hai testato contro.

  Il framework utilizzerà `targetSdkVersion` per determinare quando abilitare determinati comportamenti di compatibilità. Ad esempio, l'API di targeting di livello 23 o superiore ti consentirà di accedere al modello delle autorizzazioni di runtime .

- `minSdkVersion` è la versione minima di Android supportata dall'applicazione. Gli utenti che eseguono versioni di Android precedenti a questa versione non saranno in grado di installare l'applicazione o visualizzarla nel Play Store.

- `maxSdkVersion` è la versione massima di Android supportata dall'applicazione. Gli utenti che eseguono versioni di Android più recenti di questa versione non saranno in grado di installare l'applicazione o visualizzarla nel Play Store. Questo in genere non dovrebbe essere utilizzato poiché la maggior parte delle applicazioni funzionerà su versioni più recenti di Android senza ulteriori sforzi.

- `compileSdkVersion` è la versione di Android SDK con cui verrà compilata la tua applicazione. In genere dovrebbe essere l'ultima versione di Android rilasciata pubblicamente. Questo definisce quali API puoi accedere quando scrivi il tuo codice. Non puoi chiamare i metodi introdotti nel livello API 23 se la tua `compileSdkVersion` è impostata su 22 o inferiore.

Examples

Definizione delle versioni dell'SDK del progetto

Nel tuo file `build.gradle` del modulo principale (**app**), definisci il tuo numero di versione minimo e di destinazione.

```
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
    }
}
```

Leggi Versioni di Project SDK online: https://riptutorial.com/it/android/topic/162/versioni-di-project-sdk

Capitolo 257: Vibrazione

Examples

Iniziare con le vibrazioni

Grant Permission Vibration

prima di iniziare il codice dell'attrezzo, devi aggiungere l'autorizzazione nel manifest Android:

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

Importa libreria vibrazione

```
import android.os.Vibrator;
```

Ottieni un'istanza di Vibrator from Context

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

Controlla il dispositivo ha un vibratore

```
void boolean isHaveVibrate(){
    if (vibrator.hasVibrator()) {
        return true;
    }
    return false;
}
```

Vibrazione a tempo indeterminato

usando la *vibrazione (lungo [], int repeat)*

```
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);
```

Modelli di vibrazione

È possibile creare modelli di vibrazione passando una serie di long, ognuno dei quali rappresenta una durata in millisecondi. Il primo numero è il ritardo di inizio. Ogni voce dell'array si alterna tra vibrazione, sospensione, vibrazione, sospensione, ecc.

Il seguente esempio dimostra questo modello:

- vibrare 100 millisecondi e dormire 1000 millisecondi
- vibrare 200 millisecondi e dormire 2000 millisecondi

```
long[] pattern = {0, 100, 1000, 200, 2000};
```

Per fare in modo che il pattern si ripeta, passare l'indice nell'array del modello in cui iniziare la ripetizione o `-1` per disabilitare il ripetersi.

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(pattern, -1); // does not repeat
vibrator.vibrate(pattern,  0); // repeats forever
```

Stop Vibrazione

Se vuoi smettere di vibrare, per favore chiama:

```
vibrator.cancel();
```

Vibra per una volta

usando la *vibrazione (lunghi millisecondi)*

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(500);
```

Leggi Vibrazione online: https://riptutorial.com/it/android/topic/3359/vibrazione

Capitolo 258: VideoView

Examples

VideoView Crea

Trova VideoView in Activity e aggiungi video al suo interno.

```
VideoView videoView = (VideoView) .findViewById(R.id.videoView);
videoView.setVideoPath(pathToVideo);
```

Inizia a riprodurre video.

```
videoView.start();
```

Definisci VideoView nel file di layout XML.

```
<VideoView
    android:id="@+id/videoView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center" />
```

Riproduci video dall'URL con l'uso di 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;
        }
});
```

Leggi VideoView online: https://riptutorial.com/it/android/topic/8962/videoview

Capitolo 259: VideoView ottimizzata

introduzione

La riproduzione di un video utilizzando una `VideoView` video che estende `SurfaceView` all'interno di una riga di un `ListView` sembra funzionare inizialmente, finché l'utente non tenta di scorrere l'elenco. Non appena l'elenco inizia a scorrere, il video diventa nero (a volte viene visualizzato in bianco). Continua a giocare in background ma non puoi vederlo più perché rende il resto del video come una scatola nera. Con la VideoView ottimizzata personalizzata, i video verranno riprodotti in scroll nel `ListView` proprio come i nostri Instagram, Facebook, Twitter.

Examples

VideoView ottimizzato in ListView

Questa è la `VideoView` personalizzata che devi avere nel tuo pacchetto.

Layout VideoView personalizzato:

```
<your.packagename.VideoView
    android:id="@+id/video_view"
    android:layout_width="300dp"
    android:layout_height="300dp" />
```

Codice per `VideoView` ottimizzato personalizzato:

```
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);
    }
}
```

Aiuto da questo repository gitub . Anche se ha alcuni problemi come è stato scritto 3 anni fa sono riuscito a risolverli da solo come scritto sopra.

Leggi VideoView ottimizzata online: https://riptutorial.com/it/android/topic/10638/videoview-ottimizzata

Capitolo 260: ViewFlipper

introduzione

Un `ViewFlipper` è un `ViewAnimator` che passa da due o più viste che sono state aggiunte ad esso. Viene mostrato un solo bambino alla volta. Se richiesto, `ViewFlipper` può capovolgere automaticamente tra ogni bambino a intervalli regolari.

Examples

ViewFlipper con scorrimento dell'immagine

File XML:

```
<ViewFlipper
    android:id="@+id/viewflip"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:layout_weight="1"
    />
```

Codice 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);
    }
}
```

Leggi ViewFlipper online: https://riptutorial.com/it/android/topic/9032/viewflipper

Capitolo 261: ViewPager

introduzione

ViewPager è un gestore di layout che consente all'utente di capovolgere a sinistra ea destra tra le pagine di dati. Viene spesso utilizzato insieme a Fragment, che è un modo conveniente per fornire e gestire il ciclo di vita di ciascuna pagina.

Osservazioni

Una cosa importante da notare sull'utilizzo di ViewPager è che ci sono due diverse versioni di `FragmentPagerAdapter` e `FragmentStatePagerAdapter` .

Se si utilizza `android.app.Fragment` Frammenti nativi con un FragmentPagerAdapter o FragmentStatePagerAdapter, è necessario utilizzare le versioni di libreria di supporto V13 della scheda, ovvero `android.support.v13.app.FragmentStatePagerAdapter` .

Se si sta utilizzando la libreria di supporto `android.support.v4.app.Fragment` Fragments con FragmentPagerAdapter o FragmentStatePagerAdapter, è necessario utilizzare le versioni della libreria di supporto v4 dell'adattatore, ad esempio `android.support.v4.app.FragmentStatePagerAdapter` .

Examples

Utilizzo di base ViewPager con frammenti

Un `ViewPager` consente di mostrare più frammenti in un'attività che può essere esplorata ruotando verso sinistra o verso destra. Un `ViewPager` deve essere alimentato con Views o Fragments usando un `PagerAdapter` .

Esistono tuttavia due implementazioni più specifiche che troverete più utili in caso di utilizzo di Fragments che sono `FragmentPagerAdapter` e `FragmentStatePagerAdapter` . Quando un frammento deve essere istanziato per la prima volta, `getItem(position)` sarà chiamato per ogni posizione che ha bisogno di istanziare. Il metodo `getCount()` restituirà il numero totale di pagine in modo che `ViewPager` sappia quanti frammenti devono essere mostrati.

Sia `FragmentPagerAdapter` che `FragmentStatePagerAdapter` conservano una cache dei Frammenti che il `ViewPager` dovrà mostrare. Per impostazione predefinita, `ViewPager` tenterà di memorizzare un massimo di 3 frammenti che corrispondono al frammento attualmente visibile e quelli accanto a destra e a sinistra. Anche `FragmentStatePagerAdapter` manterrà lo stato di ciascuno dei tuoi frammenti.

Sappi che entrambe le implementazioni presuppongono che i tuoi frammenti manterranno le loro posizioni, quindi se mantieni un elenco dei frammenti invece di avere un numero statico di essi come puoi vedere nel metodo `getItem()` , dovrai creare una sottoclasse di `PagerAdapter` e

sovrascrive almeno i metodi `instantiateItem()` , `destroyItem()` e `getItemPosition()` .

Basta aggiungere un ViewPager nel layout come descritto nell'esempio di base :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>
    <android.support.v4.view.ViewPager
        android:id="@+id/vpPager">
    </android.support.v4.view.ViewPager>
</LinearLayout>
```

Quindi definire l'adattatore che determinerà il numero di pagine esistenti e il frammento da visualizzare per ciascuna pagina dell'adattatore.

```
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

Se stai usando `android.app.Fragment` devi aggiungere questa dipendenza:

```
compile 'com.android.support:support-v13:25.3.1'
```

Se stai usando `android.support.v4.app.Fragment` devi aggiungere questa dipendenza:

```
compile 'com.android.support:support-fragment:25.3.1'
```

ViewPager con TabLayout

Un `TabLayout` può essere utilizzato per una navigazione più semplice.
È possibile impostare le schede per ciascun frammento nell'adattatore utilizzando il metodo `TabLayout.newTab()` , ma esiste un altro metodo più conveniente e più semplice per questa attività, che è `TabLayout.setupWithViewPager()` .

Questo metodo si sincronizza creando e rimuovendo le schede in base al contenuto dell'adattatore associato a `ViewPager` ogni volta che viene chiamato.
Inoltre, imposterà una richiamata in modo che ogni volta che l'utente gira la pagina, verrà selezionata la scheda corrispondente.

Basta definire un layout

```
<?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>
```

Quindi implementa `FragmentPagerAdapter` e applicalo a `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

Fino a poco tempo fa, l'utilizzo di `android.support.v4.app.FragmentPagerAdapter` avrebbe impedito l'utilizzo di `PreferenceFragment` come uno dei frammenti utilizzati in FragmentPagerAdapter.

Le ultime versioni della libreria di supporto v7 ora includono la classe `PreferenceFragmentCompat` , che funzionerà con ViewPager e la versione v4 di FragmentPagerAdapter.

Esempio di frammento che estende `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) {

    }
}
```

Ora puoi usare questo frammento in una sottoclasse `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;
        }
    }

    // .......

}
```

Aggiunta di un ViewPager

Assicurati che la seguente dipendenza venga aggiunta al file `build.gradle` dell'app in dipendenze:

```
compile 'com.android.support:support-core-ui:25.3.0'
```

Quindi aggiungi `ViewPager` al tuo layout delle attività:

```
<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
```

Quindi definisci il tuo `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;
    }
}
```

Infine, imposta `ViewPager` nella tua attività:

```
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 indicatore di punti

Tutto ciò di cui abbiamo bisogno sono: ViewPager , TabLayout e 2 drawable per punti selezionati e predefiniti.

In primo luogo, dobbiamo aggiungere `TabLayout` al nostro layout dello schermo e collegarlo a `ViewPager` . Possiamo farlo in due modi:

TabLayout nidificato in 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>
```

> In questo caso, `TabLayout` verrà automaticamente collegato a `ViewPager` , ma `TabLayout` sarà accanto a `ViewPager` , non su di lui.

TabLayout separato

```
<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"/>
```

> In questo caso, possiamo mettere `TabLayout` ovunque, ma dobbiamo collegare `TabLayout` con `ViewPager` a livello di `ViewPager`

```
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 volta creato il nostro layout, dobbiamo preparare i nostri punti. Quindi creiamo tre file: `selected_dot.xml` , `default_dot.xml` e `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>
```

Ora abbiamo bisogno di aggiungere solo 3 linee di codice a `TabLayout` nel nostro layout xml e il gioco è fatto.

```
app:tabBackground="@drawable/tab_selector"
app:tabGravity="center"
app:tabIndicatorHeight="0dp"
```

Imposta OnPageChangeListener

Se è necessario ascoltare le modifiche alla pagina selezionata, è possibile implementare il listener `ViewPager.OnPageChangeListener` sul 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
    }
});
```

Leggi ViewPager online: https://riptutorial.com/it/android/topic/692/viewpager

Capitolo 262: Visualizzazione di annunci Google

Examples

Impostazione annunci di base

Dovrai aggiungere quanto segue alle tue dipendenze:

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

e poi metti questo nello stesso file.

```
apply plugin: 'com.google.gms.google-services'
```

Successivamente dovrai aggiungere informazioni rilevanti nel tuo strings.xml.

```
<string name="banner_ad_unit_id">ca-app-pub-####/####</string>
```

Successivamente, fai un'anteprima dove vuoi e personalizzala come qualsiasi altra 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>
```

E, ultimo ma non meno importante, lancia questo nel tuo 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);
```

Se hai incollato la copia esattamente ora dovresti avere un piccolo banner pubblicitario. È sufficiente posizionare più annunci pubblicitari ovunque sia necessario per ulteriori informazioni.

Aggiunta di annunci interstiziali

Gli annunci interstiziali sono annunci a schermo intero che coprono l'interfaccia della loro app host. Generalmente vengono visualizzati in punti di transizione naturali nel flusso di un'app, ad esempio tra le attività o durante la pausa tra i livelli di un gioco.

Assicurati di disporre delle autorizzazioni necessarie nel file `Manifest` :

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

1. Vai al tuo account AdMob .
2. Clicca sulla scheda **Monetizza** .
3. Seleziona o crea l'app e scegli la piattaforma.
4. Seleziona Interstitial e indica il nome di un'unità pubblicitaria.
5. Una volta creata l'unità pubblicitaria, puoi notare l'ID dell'unità pubblicitaria sul dashboard. Ad esempio: ca-app-pub-00000000000/000000000
6. Aggiungi dipendenze

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

Questo dovrebbe essere sul fondo.

```
apply plugin: 'com.google.gms.google-services'
```

Aggiungi l' **ID unità pubblicitaria** al file `strings.xml`

```
<string name="interstitial_full_screen">ca-app-pub-00000000/00000000</string>
```

Aggiungi ConfigChanges e meta-dati al tuo manifest:

```
<activity
            android:name="com.google.android.gms.ads.AdActivity"

android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScree

            android:theme="@android:style/Theme.Translucent" />
```

e

```
<meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
```

Attività:

```
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();
        }
    }

}
```

AdActivity mostrerà ora un annuncio a schermo intero.

Leggi Visualizzazione di annunci Google online:
https://riptutorial.com/it/android/topic/5984/visualizzazione-di-annunci-google

Capitolo 263: Visualizzazione elenco

introduzione

ListView è un gruppo di visualizzazione che raggruppa diversi elementi da un'origine dati come una matrice o un database e li visualizza in un elenco a scorrimento. I dati sono associati con listview utilizzando una classe Adapter.

Osservazioni

`ListView` è un gruppo di viste che visualizza un elenco di elementi scorrevoli.
Gli elementi dell'elenco vengono automaticamente inseriti nell'elenco utilizzando un `Adapter` che estrae il contenuto da un'origine come una query dell'array o del database e converte ciascun risultato di un elemento in una vista inserita nell'elenco.

Quando il contenuto del layout è dinamico o non predeterminato, è possibile utilizzare un layout che sottoclassi `AdapterView` per popolare il layout con le visualizzazioni in fase di esecuzione. Una sottoclasse della classe `AdapterView` utilizza un `Adapter` per associare i dati al relativo layout.

Prima di utilizzare `ListView` è necessario controllare anche gli esempi di `RecyclerView` .

Examples

Filtro 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);
     }
});
```

Diciamo che la tua query verrà eseguita ogni volta che l'utente digita 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) {

        }
    });
```

Custom ArrayAdapter

Per impostazione predefinita, la classe ArrayAdapter crea una vista per ogni elemento dell'array chiamando `toString()` su ciascun elemento e posizionando il contenuto in un TextView.

Per creare una vista complessa per ciascun elemento (ad esempio, se si desidera un ImageView per ciascun elemento dell'array), estendere la classe ArrayAdapter e sovrascrivere il metodo `getView()` per restituire il tipo di visualizzazione desiderato per ciascun elemento.

Per esempio:

```
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 di base con un ArrayAdapter

Per impostazione predefinita, `ArrayAdapter` crea una vista per ciascun elemento dell'array chiamando `toString()` su ciascun elemento e posizionando il contenuto in una `TextView` .

Esempio:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, myStringArray);
```

dove `android.R.layout.simple_list_item_1` è il layout che contiene un `TextView` per ogni stringa nell'array.

Quindi chiama semplicemente `setAdapter()` sul tuo `ListView` :

```
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

Per utilizzare qualcosa di diverso da TextViews per la visualizzazione dell'array, ad esempio,

ImageViews, o per avere alcuni dati oltre a risultati `toString()` , riempire le viste, sovrascrivere `getView(int, View, ViewGroup)` per restituire il tipo di visualizzazione desiderato. Controlla questo esempio .

Leggi Visualizzazione elenco online: https://riptutorial.com/it/android/topic/4226/visualizzazione-elenco

Capitolo 264: WebView

introduzione

WebView è una vista che visualizza pagine Web all'interno dell'applicazione. Da questo puoi aggiungere il tuo URL.

Osservazioni

Non dimenticare di aggiungere l'autorizzazione nel file manifest di Android

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

Examples

Dialoghi di avvisi JavaScript in WebView - Come farli funzionare

Per impostazione predefinita, WebView non implementa finestre di dialogo di avviso JavaScript, ad es. `alert()` non farà nulla. Per rendere necessario innanzitutto abilitare JavaScript (ovviamente ..), e quindi impostare un `WebChromeClient` per gestire le richieste di finestre di avviso dalla pagina:

```
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);
    }
});
```

Qui, eseguiamo l'override su `onJsAlert` , quindi eseguiamo la chiamata alla super implementazione, che ci fornisce una finestra di dialogo standard di Android. Puoi anche utilizzare tu stesso il messaggio e l'URL, ad esempio se desideri creare una finestra di dialogo personalizzata o se desideri registrarli.

Comunicazione da Javascript a Java (Android)

Attività 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
    }
}
```

Pagina Web, chiamata Javascript

```
<script>
...
Android.jsCallback();
...
Android.jsCallback('hello test');
...
</script>
```

Suggerimento extra

Passando in una struttura dati complessa, una possibile soluzione è usare JSON.

```
Android.jsCallback('{ "fake-var" : "fake-value", "fake-array" : [0,1,2] }');
```

Sul lato Android usa il tuo parser JSON preferito, ovvero: JSONObject

Comunicazione da Java a Javascript

Esempio di base

```
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!')");
   }
}
```

Esempio di dialer aperto

Se la pagina Web a contiene un numero di telefono, è possibile effettuare una chiamata utilizzando il dialer del telefono. Questo codice controlla l'url che inizia con tel: quindi fai l'intento di aprire il dialer e puoi effettuare una chiamata al numero di telefono cliccato:

```
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;
}
```

Risoluzione dei problemi di WebView stampando i messaggi della console o tramite debug remoto

Stampa dei messaggi della console webview su logcat

Per gestire i messaggi della `console` dalla pagina Web, è possibile eseguire l'override di `onConsoleMessage` in `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;
    }
}
```

E impostalo nella tua attività o frammento:

```
webView.setWebChromeClient(new ChromeClient());
```

Quindi questa pagina di esempio:

```
<html>
<head>
    <script type="text/javascript">
        console.log('test message');
    </script>
</head>
<body>
</body>
</html>
```

scriverà il log 'test message' su logcat:

> WebView: messaggio di prova sample.html: 4

`console.info()` , `console.warn()` e `console.error()` sono supportati anche da chrome-client.

Debugging remoto di dispositivi Android con Chrome

Puoi eseguire il debug remoto dell'applicazione basata su Web dal tuo desktop Chrome.

Abilita il debug USB sul tuo dispositivo Android

Sul tuo dispositivo Android, apri Impostazioni, trova la sezione Opzioni sviluppatore e abilita il debug USB.

Connetti e scopri il tuo dispositivo Android

Aprire la pagina in chrome seguente pagina: chrome: // inspect / # devices

Dalla finestra di dialogo Inspect Devices, seleziona il tuo dispositivo e premi **inspect** . Una nuova istanza di Chrome's DevTools si apre sul tuo computer di sviluppo.

Linee guida e descrizione più dettagliate di DevTools sono disponibili su developers.google.com

Apri file locale / Crea contenuto dinamico in 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" />
```

Carica i dati in 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");
```

Leggi WebView online: https://riptutorial.com/it/android/topic/153/webview

Capitolo 265: widget

Osservazioni

SDV

Examples

Dichiarazione manifesta -

Dichiarare la classe `AppWidgetProvider` nel file `AndroidManifest.xml` dell'applicazione. Per esempio:

```
<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>
```

Metadati

Aggiungi i metadati AppWidgetProviderInfo in `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>
```

Classe AppWidgetProvider

Il callback `AppWidgetProvider` più importante è **`onUpdate()`** . Viene chiamato ogni volta che viene aggiunto 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()` viene chiamato quando il widget viene posizionato o ridimensionato.

`onDeleted(Context, int[])` viene chiamato quando il widget viene cancellato.

Due widget con diversa dichiarazione di layout

1. Dichiarare due ricevitori in un file manifest:

```
<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. Crea due layout
    - `@xml/widget_1x1`
    - `@xml/widget_2x2`
3. Dichiarare la sottoclasse `UVMateWidget2x2` dalla classe `UVMateWidget` con un comportamento esteso:

```
package au.com.aershov.uvmate;

import android.content.Context;
```

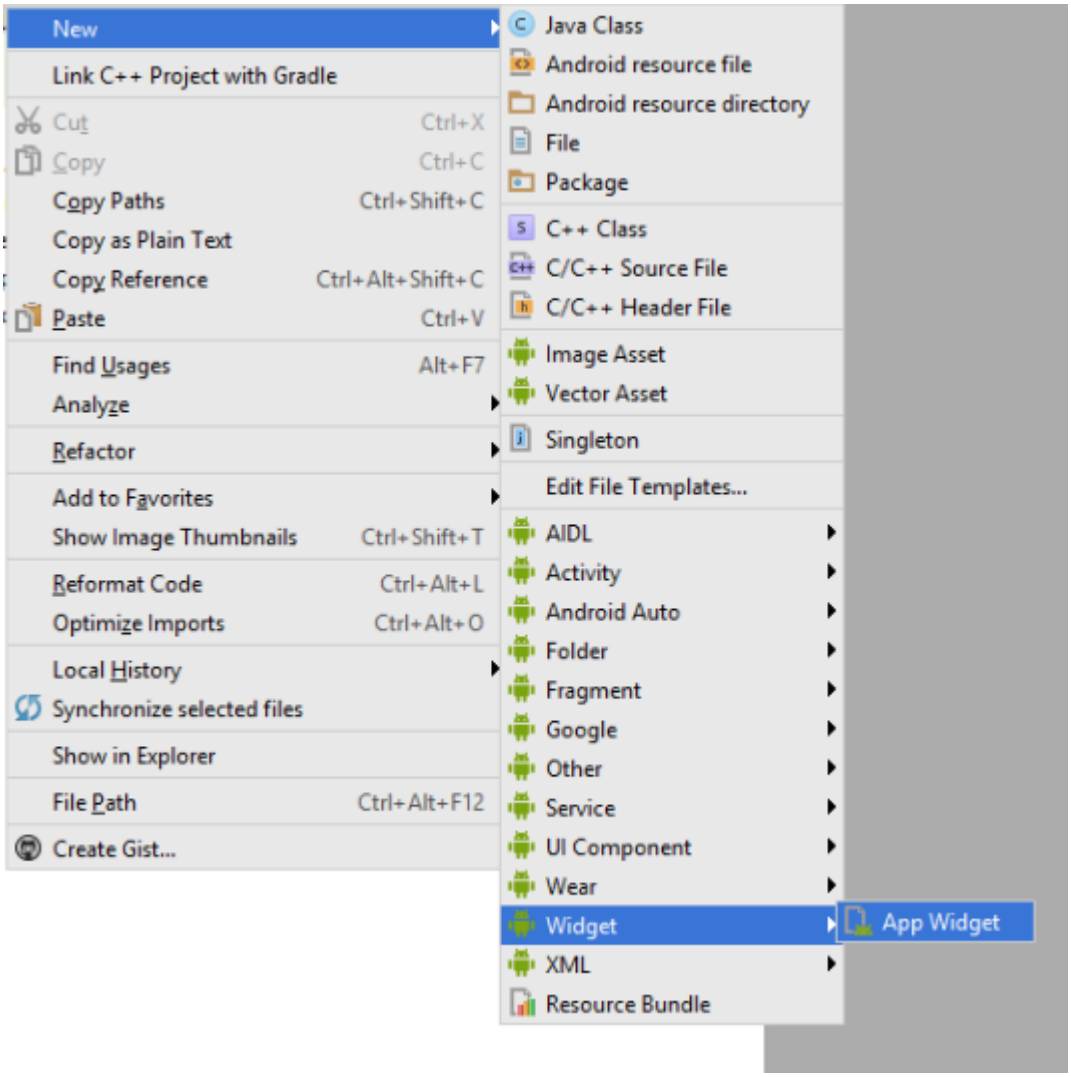
```
import android.widget.RemoteViews;

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);
    }
}
```

Crea / integri il widget di base con Android Studio

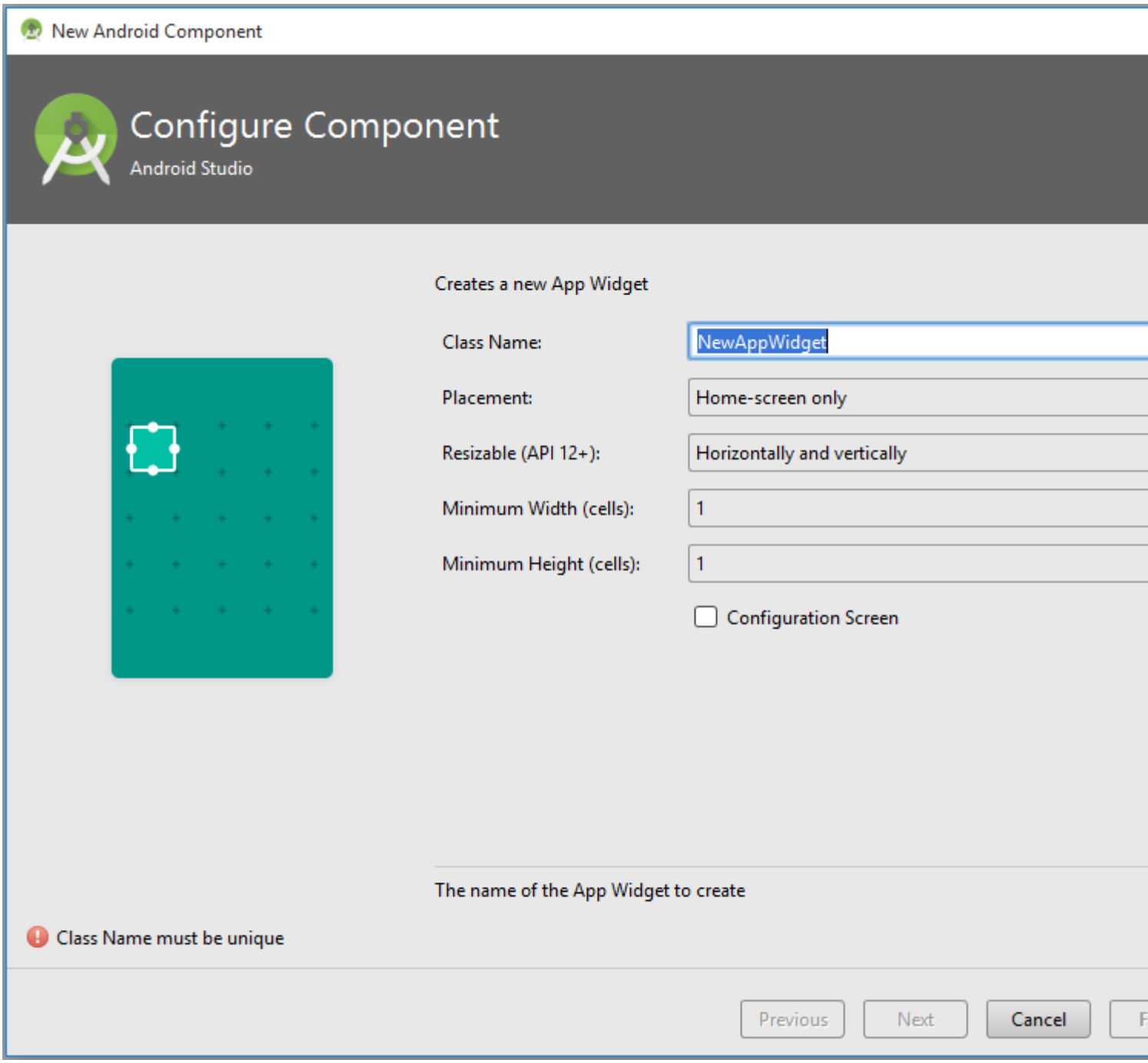
L'ultima versione di Android Studio creerà e integrerà un widget di base per la tua applicazione in 2 passaggi.

Proprio sulla tua applicazione ==> Nuovo ==> Widget ==> Widget app

.



Mostrerà una schermata come sotto e riempirà i campi



E 'fatto.

Creerà e integrerà un widget HelloWorld di base (tra cui file di layout, file di metadati, dichiarazione nel file manifest ecc.) Alla tua applicazione.

Leggi widget online: https://riptutorial.com/it/android/topic/2812/widget

Capitolo 266: XMPP registra login e chat semplici esempi

Examples

XMPP registra login e chat di esempio di base

Installa openfire o qualsiasi server di chat nel tuo sistema o sul server. Per maggiori dettagli clicca qui.

Crea un progetto Android e aggiungi queste librerie in 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'
```

Quindi crea una classe xmpp dallo scopo della connessione 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();
    }
}

}
```

Infine, aggiungi questa attività:

```
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();
    }

}
```

Assicurati che il permesso di internet sia aggiunto nel tuo file manifest.

Leggi XMPP registra login e chat semplici esempi online:
https://riptutorial.com/it/android/topic/6747/xmpp-registra-login-e-chat-semplici-esempi

Capitolo 267: Xposed

Examples

Creazione di un modulo Xposed

Xposed è un framework che ti permette di agganciare chiamate di metodo ad altre app. Quando esegui una modifica decompilando un APK, puoi inserire / modificare i comandi direttamente dove vuoi. Tuttavia, sarà necessario ricompilare / firmare l'APK in seguito e sarà possibile distribuire solo l'intero pacchetto. Con Xposed, puoi inserire il tuo codice prima o dopo i metodi, o sostituire completamente interi metodi. Sfortunatamente, puoi installare Xposed solo su dispositivi rooted. Dovresti utilizzare Xposed ogni volta che desideri manipolare il comportamento di altre app o del sistema Android principale e non vuoi passare attraverso la seccatura di decompilare, ricompilare e firmare gli APK.

Innanzitutto, crei un'app standard senza un'attività in Android Studio.

Quindi devi includere il seguente codice nel tuo *build.gradle* :

```
repositories {
    jcenter();
}
```

Successivamente, aggiungi le seguenti dipendenze:

```
provided 'de.robv.android.xposed:api:82'
provided 'de.robv.android.xposed:api:82:sources'
```

Ora devi inserire questi tag all'interno del tag *dell'applicazione* trovato in *AndroidManifest.xml,* così Xposed riconosce il tuo modulo:

```
<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: sostituire sempre **82** con l' ultima versione di Xposed .

Agganciare un metodo

Crea una nuova classe implementando `IXposedHookLoadPackage` e implementa il metodo `handleLoadPackage` :

```
public class MultiPatcher implements IXposedHookLoadPackage
{
    @Override
    public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
    {

    }
}
```

All'interno del metodo, controlli `loadPackageParam.packageName` come nome del pacchetto dell'app che vuoi collegare:

```
@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }
}
```

Ora puoi agganciare il tuo metodo e manipolarlo prima che venga eseguito il codice o dopo:

```
@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");
        }
    });
}
```

Leggi Xposed online: https://riptutorial.com/it/android/topic/4627/xposed

Capitolo 268: Youtube-API

Osservazioni

1. Prima di tutto, devi scaricare l'ultimo jar dal seguente link https://developers.google.com/youtube/android/player/downloads/
2. Devi includere questo jar nel tuo progetto. Copia e incolla questo jar nella cartella libs e non dimenticare di aggiungerlo nelle dipendenze dei file gradle {file di compilazione ('libs / YouTubeAndroidPlayerApi.jar')}
3. Hai bisogno di una chiave API per accedere alle API di YouTube. Segui questo link: https://developers.google.com/youtube/android/player/register per generare la tua chiave API.
4. Pulisci e costruisci il tuo progetto. Ora sei pronto per utilizzare YouTubeAndroidPlayerApi Per la riproduzione di un video di YouTube, devi avere un ID video corrispondente ad esso in modo da poterlo riprodurre su YouTube. Ad esempio: https://www.youtube.com/watch?v=B08iLAtS3AQ , B08iLAtS3AQ è id video che devi riprodurre su youtube.

Examples

Avvio di StandAlonePlayerActivity

1. Avvia attività di player standalone

```
     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);
```

Attività che estende 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());
    }

}
```

YoutubePlayerFragment in portrait Activty

Il codice seguente implementa un semplice YoutubePlayerFragment. Il layout dell'attività è

bloccato in modalità verticale e quando l'orientamento cambia o l'utente fa clic su schermo intero su YoutubePlayer si trasforma in lansscape con il pop-up di YouTube che riempie lo schermo. YoutubePlayerFragment non ha bisogno di estendere un'attività fornita dalla libreria di Youtube. È necessario implementare YouTubePlayer.OnInitializedListener per poter inizializzare il programma YouTubePlayer. Quindi la classe della nostra attività è la seguente

```
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 può essere aggiunto al layout xaml dell'attività come seguito

```
<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"/>

    </LinearLayout>
  </ScrollView>

</LinearLayout>
```

Infine, è necessario aggiungere i seguenti attributi nel file Manifest all'interno del tag dell'attività

```
android:configChanges="keyboardHidden|orientation|screenSize"
android:screenOrientation="portrait"
```

YouTube Player API

Ottenere la chiave API Android:

Per prima cosa è necessario ottenere l'impronta digitale SHA-1 sulla macchina utilizzando il comando java keytool. Esegui il comando sottostante in cmd / terminale per ottenere l'impronta digitale 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);
    }
}
```

Ora crea il file `Config.java` . Questo file contiene la chiave sviluppatore dell'API di Google Console e l'ID video di 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";
}
```

file 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" />
```

Consumo dell'API dati di YouTube su Android

Questo esempio ti guiderà su come ottenere i dati della playlist utilizzando l'API dei dati di YouTube su Android.

Impronta digitale SHA-1

Per prima cosa devi ottenere un'impronta digitale SHA-1 per la tua macchina. Esistono vari metodi per recuperarlo. Puoi scegliere qualsiasi metodo fornito in questo Q & A.

Console dell'API di Google e chiave YouTube per Android

Ora che hai un'impronta digitale SHA-1, apri la console dell'API di Google e crea un progetto. Vai a questa pagina e crea un progetto usando quel tasto SHA-1 e abilita l'API dei dati di YouTube. Ora otterrai una chiave. Questa chiave verrà utilizzata per inviare richieste da Android e

recuperare i dati.

Gradle part

Dovrai aggiungere le seguenti righe al tuo file Gradle per l'API dei dati di YouTube:

```
compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
```

Per poter utilizzare il client nativo di YouTube per inviare richieste, dobbiamo aggiungere le seguenti righe in 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 seguente configurazione deve essere aggiunta in Gradle per evitare conflitti:

```
configurations.all {
    resolutionStrategy.force 'com.google.code.findbugs:jsr305:3.0.2'
}
```

Di seguito viene mostrato come *apparirà* finalmente *gradle.build* .

build.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:+'
}
```

Ora arriva la parte di Java. Poiché utilizzeremo `HttpTransport` per il networking e `GsonFactory` per convertire JSON in POJO, non abbiamo bisogno di altre librerie per inviare richieste.

Ora voglio mostrare come ottenere playlist tramite l'API di YouTube fornendo gli ID della playlist. Per questa attività userò `AsyncTask` . Per capire come richiediamo i parametri e per comprendere il flusso, consulta l' API dei dati di 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;
    }
}
```

L'attività asincrona sopra riportata restituirà un'istanza di `PlaylistListResponse` che è una classe build-in di YouTube SDK. Ha tutti i campi richiesti, quindi non dobbiamo creare noi stessi POJO.

Infine, nel nostro `MainActivity` dovremo fare quanto segue:

```
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);
    }
}
```

Leggi Youtube-API online: https://riptutorial.com/it/android/topic/7587/youtube-api

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare 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	Accesso ai database SQLite utilizzando la classe ContentValues	Adil Saiyad, emecas, honk
3	Account e AccountManager	gaara87, systemovich
4	ACRA	Zarul Izham, Zoe
5	ADB (Android Debug Bridge)	3VYZkz7t, adao7000, Ahmad Aghazadeh, Amit Thakkar, AndroidMechanic, Anirudh Sharma, Anup Kulkarni, auval, Barend, Blackbelt, Burak Day, Charuಬ, 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

6	Adb shell	3VYZkz7t, Ahmad Aghazadeh, auval, Burak Day, Fabio, fyfyone Google, Hannoun Yassir, Natali, Pavel Durov, R. Zagórski, sukumar, Yury Fedorov
7	AdMob	Carlos Borau, honk, RamenChef, Sukrit Kumar, Zarul Izham, Zoe
8	Affresco	Alexander Oprisnik, Daniel Nugent, honk, Nilesh Singh, Zarul Izham
9	Aggiunta di un FuseView a un progetto Android	Tudor Luca
10	AIDL	Krishnakanth
11	AlarmManager	Daniel Nugent, devnull69, Greg T, honk, TR4Android
12	AlertDialog Box animato	krunal patel, Thomas Easo
13	Android Java Native Interface (JNI)	Doron Yakovlev-Golani, Muthukrishnan Rajendran, samgak
14	Android Vk Sdk	alexey polusov
15	Android-x86 in VirtualBox	Daniel Nugent, Enrique de Miguel
16	animatori	Aryan, Bartek Lipinski, Blundering Philosopher, Brenden Kromhout, Charuක, Daniel Nugent, Eixx, Hiren Patel, Lewis McGeary, Piyush, TR4Android, Uriel Carrillo, Yury Fedorov
17	Annotazioni Typedef: @IntDef, @StringDef	Gabriele Mariotti, hardik m, mmBs, Pongpat
18	API Android Places	busradeniz, honk, Karan Razdan, Murali
19	API di Awareness di Google	Dus, honk, Willie Chalmers III
20	API di Google Drive	Christlin Joseph, honk
21	API di Google Maps v2 per 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
22	API di impronte digitali in Android	Doron Yakovlev-Golani, RamenChef, user01232
23	API di Twitter	Mahmoud Ibrahim

24	Architettura MVP	Atif Farrukh, Harish Gyanani, honk, Jon Adams, Magesh Pandian, N J, zmingchun
25	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
26	Attività	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
27	AudioManager	honk, Nicolai Weitkemper
28	Autenticatore Android	4444, kRiZ
29	AutoCompleteTextView	Harish Gyanani, Jon Adams, Ricardo Vieira, Vivek Mishra
30	Autorizzazioni di runtime in 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
31	Autosizing TextViews	honk, Priyank Patel
32	Avvertenze sui pelucchi	ben75, Daniel Nugent, Gabriele Mariotti, GensaGames, R. Zagórski, rekire, SuperBiasedMan
33	Barra di avanzamento	Gabriele Mariotti, Hiren Patel, mpkuth, Sanoop, shtolik
34	Bitmap Cache	Lokesh Desai
35	Bluetooth e Bluetooth LE API	antonio, Jon Adams, Lukas, Myon, Pavel Durov, R. Zagórski, Reaz Murshed, V-PTR, WMios
36	Bluetooth Low Energy	Roberto Betancourt

37	BottomNavigationView	Abdul Wasae, Daniel Nugent, Gabriele Mariotti, guik, Pankaj Kumar, Pratik Butani, Priyank Patel, RamenChef, rciovati, Stephen Leppik, sud007
38	BroadcastReceiver	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
39	Camera 2 API	ChemicalFlash, devnull69, RamenChef, webo80
40	Canale di notifica Android O	Lokesh Desai
41	Caratteri personalizzati	Daniel Nugent, Erik Ghonyan, Gabriele Mariotti, Hiren Patel, honk, kit, Nougat Lover, Simon Schubert, Stanojkovic, Sujith Niraikulathan
42	CardView	Carlos, Dan, Er. Kaushik Kajavadara, Gabriele Mariotti, Kaushik, Nougat Lover, RamenChef, S.R, Sneh Pandya, Somesh Kumar, Stephen Leppik, sud007, WarrenFaith, Yury Fedorov
43	Caricamento efficace di bitmap	iDevRoids
44	caricatore	chandsie, g4s8, jefry jacky, Marcus Becker, RamenChef, Stephen Leppik
45	Cattura di screenshot	Ayush Bansal, Daniel Nugent, honk, Onik, sushant kumar, W0rmH0le
46	CleverTap	Jordan, judepereira
47	Colori	Carlos Borau, Dalija Prasnikar, Daniel Nugent, Erfan Mowlaei, Jon Adams, N J, Sujith Niraikulathan
48	Coltello da burro	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
49	Come conservare le password in modo sicuro	honk, Jaggs
50	Come usare SparseArray	honk, Robert Banyai
51	Componenti di	DeKaNszn

	architettura Android	
52	Compressione dell'immagine	Hiren Patel, mnoronha
53	Configurazione di Jenkins CI per progetti Android	honk, Ichthyocentaurs
54	Connessioni Wi-Fi	4444, AndroidMechanic, Daniel Nugent, gus27
55	ConstraintLayout	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
56	ConstraintSet	Pratik Butani
57	Contesto	Will Evers
58	Conto alla rovescia	privatestaticint
59	Controlla la connessione dati	sukumar, Suresh Kumar
60	Conversazione di testo in testo	Hitesh Sahu, honk, RamenChef, Stephen Leppik
61	Converti la stringa vietnamita in una stringa inglese Android	1SStorm
62	CoordinatorLayout and Behaviors	Adarsh Ashok, Gabriele Mariotti, honk, RamenChef, Stephen Leppik
63	corsia di sorpasso	Gokhan Arik
64	Cos'è ProGuard? Cosa si usa in Android?	Ayush Bansal, Daniel Nugent, Ghanshyam Sharma, Pratik Butani
65	Cose Android	Fabio, honk
66	Costruire le app compatibili con le versioni precedenti	Jon Adams, mnoronha, RamenChef, SoroushA
67	Crash Reporting Tools	Ajit Singh, Charuಎ, Ekin, Gabriele Mariotti, Ishita Sinha, Jason Bourne, Madhukar Hebbar, pRaNaY
68	Crea ROM	honk, Pradumn Kumar Mahanta

	personalizzate per Android	
69	Crea una classe Singleton per il messaggio Toast	Emad, Ishan Fernando
70	Creare le tue librerie per le applicazioni Android	EpicPandaForce, honk, mnoronha
71	Creazione della schermata Splash	honk, Kiran Benny Joseph, Zoe
72	Creazione di finestre sovrapposte (sempre in primo piano)	honk, mnoronha, NitZRobotKoder, Rupali, Sujith Niraikulathan
73	Creazione di viste personalizzate	AndroidMechanic, Barend, Bartek Lipinski, Charuಎ, 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	Criteri della modalità rigorosa: uno strumento per catturare l'errore nel tempo di compilazione.	Shekhar
75	Crittografia / decrittografia dei dati	honk, HoseinIT, Robert
76	Crostini	Adam Ratzman, adao7000, Aida Isay, Amit, Andrew Brooke, AndroidMechanic, Avijit Karmakar, Bartek Lipinski, cdeange, Charuಎ, Daniel Nugent, Erik Ghonyan, Gabriele Mariotti, Lewis McGeary, LordSidious, Lukas, mpkuth, MrSalmon, RamenChef, Rohit Arya, Sammy T, saurav, SoroushA, sukumar, Vicky, Vucko
77	Data / ora localizzate in Android	Geert, honk, mnoronha
78	Decorazioni RecyclerView	Barend, David Medenjak, Gabriele Mariotti, Muthukrishnan Rajendran, Peter Gordon, RamenChef, Stephen Leppik, Yasin Kaçmaz
79	Definire il valore del passo (incremento) per RangeSeekBar personalizzato	Romu Dizzy

80	Design dei materiali	Akash Patel, Aleksandar Stefanović, Alex Chengalan, AndroidMechanic, Anirudh Sharma, ankit dassor, Bartek Lipinski, Bulwinkel, cascal, Charuಬ, 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
81	Dialogo	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
82	Dipingere	Nicolas Maltais
83	Disegni vettoriali	Priyank Patel, ShahiM
84	Disegno su tela con SurfaceView	davidgiga1993
85	Dividi schermo / Attività multischermo	Vishal Puri
86	Doze Mode	Daniel Nugent, Fabio, honk, NitZRobotKoder, RamenChef, Rosário Pereira Fernandes, Rupali
87	drawable	alanv, B001, Daniel Nugent, Greg T, Hiren Patel, Jinesh Francis, Nick Cardoso, TR4Android
88	eccezioni	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
89	Email di convalida	Hiren Patel, honk, iravul, Nicolas Maltais
90	Emulatore	Ahmad Aghazadeh, Dan Hulme, fyfyone Google, honk, rekire, Rubin Nellikunnathu, ThomasThiebaud

91	Esegui istantaneamente in Android Studio	AndroidMechanic, Daniel Nugent, ridsatrio, Zoe
92	EventBus di GreenRobot	CaseyB, Daniel Nugent, Hamed Momeni, RamenChef
93	ExoPlayer	Hamed Gh
94	Facebook SDK per Android	Akeshwar Jha, AndiGeeky, Community, Daniel Nugent, honk, Zarul Izham
95	Fastjson	KeLiuyue
96	Fatturazione in-app	Hussein El Feky, Pro Mode, Zoe
97	File zip in Android	Adnan
98	FileIO con Android	h22, sun-solar-arrow
99	FileProvider	Joost Verbraeken, pedros
100	Filo	Daniel Nugent, PRIYA PARASHAR, RamenChef
101	Firebase	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
102	Firebase Cloud Messaging	Gabriele Mariotti, shikhar bansal, Shubham Shukla, Zarul Izham
103	Firebase Crash Reporting	AndiGeeky, Gabriele Mariotti, honk, RamenChef, Stephen Leppik, Zarul Izham
104	Firebase Realtime DataBase	Aawaz Gyawali, drulabs, Gabriele Mariotti, honk, Md. Ali Hossain, RamenChef, Sneh Pandya, Stephen Leppik, yennsarah, Zarul Izham
105	Firma la tua app per Android per la versione	Gabriele Mariotti, M M
106	FloatingActionButton	Ahmad Aghazadeh, Charuක, Daniel Nugent, Gabriele Mariotti, mattfred, RamenChef, Shinil M S, Stephen Leppik
107	Fogli inferiori	Daniel Nugent, Gabriele Mariotti, Magesh Pandian, MiguelHincapieC, RamenChef, Stephen Leppik, sud007, Zarul Izham
108	Formattare stringhe	Beena, Daniel Nugent, gaara87, Greg T, Michele, RamenChef, Suresh Kumar

109	Formattazione dei numeri di telefono con pattern.	Pedro Varela
110	Fornitore di contenuti	Andrew Siplas, cdeange, Daniel Nugent, Dinesh Choudhary, Lewis McGeary, RamenChef
111	frammenti	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
112	Genymotion per Android	Atef Hares, Harish Gyanani
113	Gestione degli eventi di tocco e movimento	honk, Zoe
114	Gestire i collegamenti profondi	Doron Yakovlev-Golani, Harsh Sharma, mnoronha, Tanis.7x
115	Google Play Store	dakshbhatt21, Daniel Nugent, reVerse
116	Gradle per 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, Charuꜱ, 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, ɬolɐəz əɥɬ qoq, Vlonjat Gashi, Yassie, yuku, Yury Fedorov
117	GreenDAO	Allan Pereira, Carl Poole, Grundy, MiguelHincapieC, R. Zagórski, RamenChef, Stephen Leppik
118	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
119	handler	Daniel Nugent, Floern, Hasif Seyd, Lewis McGeary, Mike

		Scamell, Muhammed Refaat, Sweeper, Tomik, TR4Android
120	HttpURLConnection	Aleks G, Daniel Nugent, Duan Bressan, honk, KDeogharkar, marshmallow, Shantanu Paul, Simone Carletti
121	Il file manifest	John Snow, Jon Adams, kit, mayojava, Menasheh
122	ImageView	Ahmad Aghazadeh, Ali Sherafat, Chip, Daniel Nugent, DanielDiSu, Dinesh, Gabriele Mariotti, Harish Gyanani, kit, lax1089, Pratik Butani, Squidward, Sup
123	Immagini 9-Patch	Knossos, Nissim R, Tomik
124	Impaginazione in RecyclerView	Muhammad Younas
125	Indicizzazione dell'app Firebase	shalini, tynn
126	Iniziare con OpenGL ES 2.0+	MarGenDo
127	Installazione di app con ADB	Ahmad Aghazadeh, fyfyone Google, Laurel, Xaver Kapeller
128	Integra Google Accedi	AndiGeeky, RamenChef, Tot Zam
129	Integrare OpenCV in Android Studio	MashukKhan, RamenChef, ssimm
130	Integrazione del gateway Paypal per Android	A-Droid Tech
131	Integrazione di accesso Google su Android	jagapathi
132	Intenti impliciti	Blundering Philosopher, Daniel Nugent, mnoronha, Pratik Butani, SoroushA
133	Intento	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, Charuಠ, 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
134	IntentService	Anax, Daniel Nugent, honk, JonasCz, TRINADH KOYA, Yashaswi Maharshi
135	interfacce	appersiano, Brenden Kromhout, Daniel Nugent, RediOne1
136	Internazionalizzazione e localizzazione (I18N e L10N)	Ankur Aggarwal
137	Jackson	KeLiuyue
138	Java su Android	Eugen Martynov
139	JCodec	Adhikari Bishwash
140	JSON in 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
141	layout	a.ch., Adarsh Ashok, Adinia, AesSedai101, Ahmad Aghazadeh, Aleksandar Stefanović, ankit dassor,

		Aurasphere, Bartek Lipinski, Björn Kechel, bjrne, Brenden Kromhout, Charuක, 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
142	Leakcanary	Rakshit Nawani, tynn
143	Lettura codice a barre e QR code	FlyingPumba
144	Library Dagger 2: Iniezione delle dipendenze nelle applicazioni	Er. Kaushik Kajavadara, honk
145	Libreria di associazione dati	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
146	Localizzazione con risorse in Android	AndroidMechanic, electroid, Fabio, Gubbel, Harish Gyanani, honk, Jinesh Francis, mpkuth, RamenChef, USKMobility
147	Looper	tynn
148	LRUCache	Daniel Nugent, honk, LordSidious, RamenChef, Stephen Leppik
149	Macchina fotografica e galleria	Ahmad Aghazadeh, carvaq, Daniel Nugent, Hiren Patel, johnrao07, RediOne1, Squidward, Yasin Kaçmaz
150	Media Player	Ahmad Aghazadeh, Carlos Vázquez Losada, hello_world, Makille, R. Zagórski, Redman
151	MediaSession	Disk Crasher, honk, KuroObi, RamenChef
152	MediaStore	Daniel Nugent, honk, RamenChef, Uttam Panchasara
153	Memorizzazione di file in Archiviazione interna ed esterna	Amit Vaghela, Andrew Brooke, AnV, Daniel Nugent, Gabriele Mariotti, Nickan B, Uttam Panchasara

154	Menu	Bhargavi Yamanuri, Chip, Daniel Nugent, Hi I'm Frogatto, honk, Iman Hamidi
155	Metriche di visualizzazione del dispositivo	Daniel Nugent, Hiren Patel, Talha, W3hri
156	Miglioramento delle finestre di dialogo degli avvisi	Adil Saiyad, honk
157	Miglioramento delle prestazioni di Android utilizzando i font icona	Beto Caldas, honk, Neeraj
158	Modalità PorterDuff	Adarsh Ashok, AndroidMechanic, Knossos, PhilLab, S.D., Vasily Kabunov
159	Modelli di progettazione	Adhikari Bishwash, honk, Steve.P
160	Modifica il testo	Daniel Nugent, Gabriele Mariotti, Kaushik NP, Muthukrishnan Rajendran, Rubin Nellikunnathu, Yousha Aleayoub
161	Modifiche all'orientamento	EmmanuelMess, k3b, Ricardo Vieira, y.feizi
162	Modo rapido per impostare Retrolambda su un progetto Android.	anatoli, Md. Ali Hossain
163	Moshi	Blundell
164	Multidex e il limite del metodo Dex	Adarsh Ashok, Ben, bigbaldy, cdeange, Daniel Nugent, Gabriele Mariotti, Mike, Pongpat, R. Zagórski, Shirane85
165	MVVM (Architettura)	Daniel W., RamenChef, Stephen Leppik
166	NavigationView	Adam Lear, akshay, Charuᔕ, Daniel Nugent, Gabriele Mariotti, Kedar Tendolkar, petrumo, RamenChef, rekire, SANAT, Sevle, Stephen Leppik, sud007
167	NDK Android	Alex, astuter, Doron Yakovlev-Golani, Flayn, Onik, samgak , still_learning, Täg, thiagolr
168	notifiche	alexey polusov, bricklore, Da-Jin C, Daniel Nugent, Dus, gbansal, Jeeter, piotrek1543, RediOne1, Rupali, TR4Android, weston
169	OkHttp	A-Droid Tech, Daniel Nugent, Gabriele Mariotti, Gubbel,

		noob, Rohit Arya, Vucko, Zarul Izham
170	Okio	Adhikari Bishwash
171	ORMLite in Android	Manos
172	Ottenere calcolato Visualizzare le dimensioni	mnoronha, stkent
173	Ottenere i nomi dei font di sistema e usare i font	Adil Saiyad, honk
174	Ottimizzazione del kernel Android	honk, Sneh Pandya
175	Ottimizzazione delle prestazioni	honk, Jonas Köritz
176	Otto Event Bus	gus27, tynn
177	PackageManager	FredMaggiowski, Hi I'm Frogatto, Muthukrishnan Rajendran, Piyush, Squidward
178	Parcelable	Alex Sullivan, Andrei T, HoseinIT, Nick Cardoso
179	Perdite di 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
180	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
181	Ping ICMP	Carl Poole
182	planata	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
183	Port Mapping utilizzando	Shinil M S

	la libreria Cling in Android	
184	Posizione	Alex Chengalan, Aryan, BadCash, Daniel Nugent, Hiren Patel, Mahmoud Ibrahim, MidasLefko, Pablo Baxter, RamenChef, Stephen Leppik
185	Processore di annotazione	krishan
186	Programmazione Android con Kotlin	Gian Patrick Quintana, Govinda Paliwal, Oknesif, Zarul Izham
187	Programmazione del lavoro	RamenChef, reflective_mind
188	ProGuard: offuscamento e riduzione del codice	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
189	Pubblica il file .aar su Apache Archiva con Gradle	Marian Klühspies
190	Pubblica su Play Store	Carlos Borau, Fabio, mnoronha, Zoe
191	Pubblica una libreria per i repository di Maven	Farid
192	Pugnale 2	Aurasphere, Cabezas, David Medenjak, EpicPandaForce, honk, mattfred, Tomik
193	Pulsante	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
194	Pulsante hardware Eventi / Intenti (PTT, LWP, ecc.)	JensV
195	Raccoglitori di data e ora	adalPaRi, Brenden Kromhout, Daniel Nugent, Harish Gyanani, Ironman, Milad Nouri, RediOne1, Rohan Arora
196	raffica	2943, Ankur Aggarwal, Endzeit, Harsh Dalwadi, herrmartell, honk, Jon Adams, Pablo Baxter, RamenChef, Rubin

		Nellikunnathu, Rucha Bhatt, sameera lakshitha, Stephen Leppik, VISHWANATH N P
197	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, Charuज, 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
198	RecyclerView e LayoutManagers	4444, BalaramNayak, Felix Edelmann, Gabriele Mariotti, Kayvan N, MidasLefko, RamenChef, Stephen Leppik
199	RecyclerView onClickListeners	abhishesh, Braj Bhushan Singh, Bryan, FromTheSeventhSky, fuwaneko, Gabriele Mariotti, honk, RamenChef, Smit.Satodia, Stephen Leppik
200	Registrazione e utilizzo di 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
201	Regno	bdash, Dan, EpicPandaForce, Hi I'm Frogatto, iurysza, null pointer, RamenChef, Stephen Leppik, sukumar
202	RenderScript	Ankit Popli, Dalija Prasnikar, Froyo, honk, Rucha Bhatt, Xaver Kapeller
203	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

204	Retrofit2 con RxJava	Anand Singh, gaara87, GurpreetSK95, Lukas, mrtuovinen, R. Zagórski, Zarul Izham
205	Riconoscimento di attività	Pablo Baxter
206	Rileva evento scossa in Android	N-JOY, tynn, Xiaozou
207	Rilevamento dei gesti	mpkuth
208	risorse	biddulph.r, Brenden Kromhout, Charuක, 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
209	RoboGuice	AndroidRuntimeException, Lewis McGeary, Rajesh
210	Robolectric	Blundell, g4s8
211	Schermi di supporto con diverse risoluzioni, dimensioni	Eduardo, Guilherme Torres Castro, kalan, mpkuth, Onur, ppeterka
212	Scorri per aggiornare	Chirag SolankI, Daniel Nugent, Gabriele Mariotti, Malek Hijazi, RamenChef, Stephen Leppik
213	Scrittura di test dell'interfaccia utente - Android	Atif Farrukh, Daniel Nugent, Gabriele Mariotti, honk, Jon Adams, originx
214	SearchView	Daniel Nugent, Dmide, Hiren Patel, RamenChef, Stephen Leppik, sud007
215	Secure SharedPreferences	Christlin Joseph
216	SensorManager	honk, Simon, TDG
217	Servizio	adao7000, AndroidMechanic, Apoorv Parmar, BadCash, B-GangsteR, Daniel Nugent, g4s8, Hiren Patel, JonasCz, Lazai, Lucas Paolillo, Michael Spitsin, Nougat Lover, rakeshdas, Vinícius Barros
218	SharedPreferences	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
219	ShortcutManager	g4s8, Sukrit Kumar
220	Sicurezza	xDragonZ
221	Sincronizzazione dei dati con l'adattatore di sincronizzazione	Arpit Gandhi, mnoronha
222	Smart card	shadygoneinsane
223	Snack bar	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
224	SpannableString	S.R
225	Spinner	AndroidMechanic, Anonsage, Daniel Nugent, Vishwesh Jainkuniya
226	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
227	Strumenti Attributi	Dalija Prasnikar, Gabriele Mariotti, Harsh Sharma, Kayvan N, TR4Android
228	Studio Android	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
229	Suono e supporti Android	johnrao07, Muhammad Umair Shafique, Squidward
230	Sviluppo di giochi Android	Zoe

231	SyncAdapter esegue periodicamente la sincronizzazione dei dati	Bhargavi Yamanuri
232	TabLayout	Daniel Nugent, Willie Chalmers III
233	Tastiera	Hiren Patel, Kayvan N
234	Tema DayNight (AppCompat v23.2 / API 14+)	Ishita Sinha
235	Tema, stile, attributo	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
236	tensorflow	Pratik Butani
237	Test dell'interfaccia utente con Espresso	Daniel Nugent, Gabriele Mariotti, Jason Robinson, Michael Vescovo, Milad Faridnia, N J, RamenChef, Víctor Albertos
238	Test dell'interfaccia utente Inter-App con UIAutomator	Timo Bähr
239	Test delle unità in Android con JUnit	abhi, Andre Perkins, AndroidMechanic, Eugen Martynov, honk, Lewis McGeary, N J, Namnodorel, Patrick Dattilio, Rolf ツ
240	Text to Speech (TTS)	Ahmad Aghazadeh, honk, Jordan, Lukas, nibarius, Peter Taylor, RamenChef, Stephen Leppik
241	TextInputLayout	Adarsh Ashok, BrickTop, Gabriele Mariotti, Hi I'm Frogatto, RamenChef, Shashanth, Sneh Pandya, Stephen Leppik
242	TextView	Beena, Daniel Nugent, Eyad Mhanna, gaara87, Gabriele Mariotti, Hiren Patel, honk, keno, Michele, Sohail Zahid, Sujith Niraikulathan, sun-solar-arrow
243	Time Utils	Burhanuddin Rashid, Mukesh Kumar Swami, Muthukrishnan Rajendran
244	Tocca Eventi	Yvette Colomb
245	Traccia audio	Ayush Bansal
246	TransitionDrawable	S.R, Yogesh Umesh Vaity
247	Transizioni di elementi	noongiya95

	condivise	
248	UI Lifecycle	Daniel Nugent, Dinesh Choudhary, Floern, Lewis McGeary, orelzion, R. Zagórski, Sergey Glotov
249	Universal Image Loader	Greg T, honk, Jon Adams, priyankvex, Stephen Leppik
250	Unzip File in Android	Arth Tilva, Daniel Nugent, mnoronha
251	URL di richiamata	Atif Farrukh, honk, RamenChef, Stephen Leppik
252	VectorDrawable e AnimatedVectorDrawable	Ahmad Aghazadeh, Aleksandar Stefanović, gaara87, honk, Lewis McGeary, RamenChef, Stephen Leppik
253	Verifica la connettività Internet	AndiGeeky, Bill, Daniel Nugent, gbansal, Ichigo Kurosaki, Jon Adams, sukumar, TameHog, Yousha Aleayoub
254	Versioni Android	4444, AndroidMechanic, athor, BooleanCheese, Dalija Prasnikar, Daniel Nugent, Fildor, Gabriele Mariotti, H. Pauwelyn, Matt, RediOne1, tynn
255	Versioni di Project SDK	Arnav M., Ranveer, Tanis.7x
256	Vibrazione	cdeange, Zerntrino
257	VideoView	iravul, Sashabrava
258	VideoView ottimizzata	Chip
259	ViewFlipper	Anita Kunjir, Daniel Nugent, honk
260	ViewPager	Adarsh Ashok, Adrián Pérez, Daniel Nugent, Gabriele Mariotti, Moustachauve, RamenChef, RediOne1, Rucha Bhatt, Sneh Pandya, Stephen Leppik, Usman, ZeroOne
261	Visualizzazione di annunci Google	Egek92, RamenChef, ReverseCold, Stephen Leppik, Zarul Izham
262	Visualizzazione elenco	A.A., brainless, Daniel Nugent, Diti, Douglas Drumond, Fabian Tamp, Gabriele Mariotti, Hiren Patel, Mr.7, Ruben Pirotte, Saeed-rz, shaonAshraf, Squidward
263	WebView	Amod Gokhale, Daniel Nugent, g4s8, j2ko, jasonlam604, JonasCz, Mohammad Yahia, ppeterka, Prakash Bala, shtolik, Squidward, Sukrit Kumar, sukumar
264	widget	4444, Alex Ershov, Daniel Nugent, Don Chakkappan, Imdad, nenofite, sun-solar-arrow
265	XMPP registra login e	4444, RamenChef, Saveen

	chat semplici esempi	
266	Xposed	Medusalix
267	Youtube-API	abhishesh, Giannis, honk, MashukKhan, Zarul Izham, Zeeshan Shabbir