Бесплатная электронная книга

УЧУСЬ Android

Free unaffiliated eBook created from
Stack Overflow contributors.

#android

1
1: Android 2
2
2
Examples 3
Android Studio 3
Android Studio 4
/ 4
4
5
Android Studio 5
5
5
- API 6
9
10
15
Android- 15
Android Studio 15
APK 16
Android- IDE 16
16
Android SDK 17
17
18
20
20
21
21
21

22

23

AVD (Android Virtual Device) 24

2: 9-- 30

30

Examples 30

31

31

32

3: ACRA 33

33

33

33

Examples 33

ACRAHandler 33

34

34

4: ADB (Android Debug Bridge) 35

35

35

Examples 35

35

35

36

36

ADB WiFi 39

39

40

, USB- 40

41

() () 41

42

/ Wi-Fi 42
42
IP- 43
Start / stop adb 43
44
ADB 45
(kitkat) 46
: 1 (adb) 46
: 2 () 46
47
47
47
48
49
ADB Linux 49
, An 50
(/ /) 50
51
51
5: adb shell 52
52
52
52
Examples 52
, Android ADB 52
54
API 23+ 54
55
55
chmod 56
/ adb 57
Open Developer 58

« » 58
/ 58
Android 59
6: AdMob 60
60
60
60
Examples 60
60
Build.gradle 60
61
XML 61
61
7: AIDL 63
63
Examples 63
AIDL 63
8: AlarmManager 65
Examples 65
65
65
Android 66
API23 + Doze AlarmManager 66
9: Android Authenticator 68
Examples 68
68
10: Android NDK 71
Examples 71
Android 71
72
makefile, Android.mk 72

ndk 72
11: Android Studio 74
Examples 74
74
75
logcat, 77
/ 78
Android Studio 79
Android Studio 81
Android Studio 82
Android Studio 82
«» 83
84
12: Android Vk Sdk 86
Examples 86
86
13: Android 88
Examples 88
88
14: Android-x86 VirtualBox 90
90
Examples 90
90
SDCARD 90
93
15: API Android Places 97
Examples 97
97
API 98
99
Google 100
PlaceAutocomplete 101

16: API Google 103
..... 103
..... 103
Examples 103
Google Android 103
Google 114
DriveContents 114
..... 115
..... 116
17: API Twitter 117
Examples 117
Twitter 117
18: API 2 119
..... 119
..... 119
Examples 120
TextureView 120
19: API android 130
..... 130
Examples 130
Android 130
API Fingerprint API 131
20: API Google 141
..... 141
Examples 142
Snapshot API 142
Snapshot API 142
Snapshot API 142
Snapshot API 142
API- 143
Fence API 143

Fence API 145
21: AsyncTask 147
147
Examples 147
147
147
: 148
149
AsyncTask 149
150
150
AsyncTask Android 150
Android AsyncTask 151
Android AsyncTask 151
WeakReference 154
155
AsyncTask: 156
THREAD_POOL_EXECUTOR 156
SERIAL_EXECUTOR 156
(1) 158
22: AudioManager 160
Examples 160
160
160
23: AutoCompleteTextView 161
161
Examples 161
, AutoCompleteTextView 161
CustomAdapter, ClickListener 161
: activity_main.xml 161
row.xml 162
strings.xml 162

MainActivity.java ... 162
: People.java ... 163
: PeopleAdapter.java ... 164
24: Bluetooth Bluetooth LE API ... 166
... 166
Examples ... 166
... 166
, Bluetooth ... 166
... 167
Bluetooth ... 167
Bluetooth ... 168
Bluetooth Low Energy ... 170
25: BottomNavigationView ... 175
... 175
... 175
: ... 175
Examples ... 175
... 175
BottomNavigationView ... 176
/ ... 177
3 ... 177
26: BroadcastReceiver ... 179
... 179
Examples ... 179
... 179
BroadcastReceiver ... 180
LocalBroadcastManager ... 180
Bluetooth Broadcast ... 181
... 181
( ) ... 181
... 181
... 182

182

BroadcastReceiver BOOT_COMPLETED 182

LocalBroadcastManager 183

..... 184

..... 185

..... 185

Android 186

27: CardView 188

..... 188

..... 188

..... 189

: 189

Examples 189

CardView 189

CardView 191

Ripple 191

CardView (Lollipo 192

Animate CardView TransitionDrawable 194

28: CleverTap 195

..... 195

..... 195

Examples 195

SDK 195

..... 195

29: ConstraintLayout 196

..... 196

..... 196

..... 197

..... 197

: 197

Examples 198

ConstraintLayout 198

198
30: ConstraintSet 200
200
Examples 200
ConstraintSet ContraintLayout 200
31: ExoPlayer 201
Examples 201
ExoPlayer 201
ExoPlayer 201
Tr 202
32: Facebook SDK Android 203
203
203
Examples 203
Facebook Android 203
Facebook 205
Facebook 206
Facebook / 207
Facebook 208
33: Fastjson 209
209
209
Examples 209
JSON Fastjson 209
Map JSON 211
34: Fastlane 213
213
Examples 213
Fastfile - Crashlytics 213
Fastfile 216
35: FileIO Android 217
217

217

Examples 217

217

217

218

(SD-) 218

« MTP» 219

219

36: FileProvider 221

Examples 221

221

, , . 221

FileProvider 221

URI 222

222

37: Firebase 224

224

224

Firebase - : 224

: 224

Examples 224

Firebase 224

Firebase 225

Firebase 227

Firebase 228

229

Firebase 230

Firebase 232

Firebase 238

Firebase FCM SDK 238

238

Firebase Android- 240
Firebase 240
SDK 240
Firebase: / 242
FCM 243
Firebase 253
38: FloatingActionButton 255
255
255
255
: 256
: 256
Examples 256
FAB 256
FloatingActionButton 257
FloatingActionButton 259
FloatingActionButton 262
39: Genymotion android 263
263
Examples 263
Genymotion, 263
1 - VirtualBox 263
2 - Genymotion 263
3 - Genymotion 263
4 - Genymotion 263
5 - genymotion Android Studio 263
6 - Genymotion Android Studio 264
Google Genymotion 264
40: Google Maps API v2 Android 266
266
266

Examples 266
Google Map 266
Google Map 267
277
MapView: GoogleMap 278
Google 280
SH1- 286
Google (Lite) 287
UISettings 287
SHA1 288
InfoWindow Click Listener 289
291
41: Google Play 292
Examples 292
Google Play 292
Google Play Store 292
42: Gradle Android 294
294
294
294
Gradle for Android - : 295
Examples 295
build.gradle 295
DSL (,) 296
296
DSL 296
296
, 298
signingConfig 298
298
299

, ... 300
..... 300
BuildConfigField 300
ResValue 301
«dependencies.gradle» 303
..... 304
, ... 305
Android Studio build.gradle? 305
..... 306
Gradle 307
..... 308
APK 309
A: keystore.properties. **309**
B: **310**
«version.properties» 311
apk : 312
APK 312
Proguard gradle 312
NDK- Gradle AndroidStudio 313
MyApp / build.gradle **313**
MyApp / app / build.gradle **314**
, ... **314**
gradle 315
«» apk 317
..... 317
..... 317
gradle.properties / buildconfigurations 318
..... 319
..... 320
43: GreenBot EventBus **322**
..... 322

322

Examples....322

Event....322

....322

....323

....323

44: GreenDAO....326

....326

Examples....326

SELECT, INSERT, DELETE, UPDATE....326

GreenDAO 3.X, ...328

GreenDao v3.X....329

45: Gson....332

....332

....332

Examples....333

JSON Gson....333

JSON Gson....335

....335

/ JSON AutoValue Gson....335

JSON Gson....337

Gson 337

Gson JSON- 338

Gson....338

Gson 339

json Gson....340

JSON Gson....341

Gson 342

46: HttpURLConnection....345

....345

....345

Examples....345

HttpURLConnection 345
HTTP GET 346
HTTP GET 347
HttpURLConnection multipart / form-data 347
HTTP POST 350
(POST) HttpURLConnection 351
HttpURLConnection HTTP- 352
..... 355
47: ImageView 356
..... 356
..... 356
..... 356
Examples 356
..... 356
..... 357
ImageView ScaleType - 358
ImageView ScaleType - CenterCrop 361
ImageView ScaleType - CenterInside 362
ImageView ScaleType - FitStart FitEnd 362
ImageView ScaleType - FitCenter 362
ImageView ScaleType - FitXy 362
..... 362
..... 367
MLRoundedImageView.java 368
48: IntentService 370
..... 370
..... 370
Examples 370
IntentService 370
..... 370
IntentService 371
49: Java Android 374
.....

374

Examples....374

Java 8 Retrolambda....374

50: JCodec....377

Examples....377

....377

....377

51: JSON Android org.json....378

....378

....378

Examples....378

JSON....378

JSON....379

JSONArray JSONObject....380

JSON 380

json....381

JsonReader JSON 382

JSON....383

JSON....383

JSON....384

JSON....385

52: Leakcanary....387

....387

....387

Examples....387

Leak Canary Android....387

53: Lint Warnings....388

....388

:....388

Examples....388

: xml-....388

«»....389

LintOptions 389
lint.xml 390
Java XML 391
lint Java 391
XML 391
..... 391
54: Looper 393
..... 393
Examples 393
LooperThread 393
HandlerThread 393
55: LruCache 394
..... 394
Examples 394
..... 394
() 394
(Resouce) 395
56: MediaSession 396
..... 396
..... 396
Examples 396
..... 396
57: MediaStore 399
Examples 399
/ MP3- 399
..... **401**
58: Moshi 403
..... 403
..... 403
Examples 403
JSON Java 403

Java JSON 403
..... 403
59: MVVM () 405
..... 405
Examples 406
MVVM DataBinding 406
60: NavigationView 414
..... 414
: 414
: 414
Examples 414
NavigationView 414
..... 419
..... 420
, DividerItemDecoration 421
61: Notification Channel Android O 423
..... 423
..... 423
..... 423
Examples 423
..... 423
62: OkHttp 430
Examples 430
..... 430
..... 430
..... 430
..... 431
..... 431
..... 432
..... 432
OkHttp 433
63: ORMLite android 434

Examples....434
Android OrmLite SQLite....434
....434
....435
SQLite....437
64: PackageManager....439
Examples....439
....439
....439
....439
PackageManager....440
65: Parcelable....442
....442
....442
Examples....442
....442
Parcelable, Parcelable....443
Enums with Parcelable....444
66: Ping ICMP....446
....446
Examples....446
Ping....446
67: ProGuard - 447
Examples....447
....447
ProGuard 449
( ) 450
....450
ProGuard 451
68: RecyclerView....453
....453

453

.....453

:.....454

.....454

:.....454

Examples.....455

RecyclerView.....455

.....456

RecyclerView.....457

/ RecyclerView.....458

ViewHolders ItemViewType.....461

RecyclerView SearchView.....462

recyclerView.....463

.....465

SortedList.....467

RecyclerView DataBinding.....469

Recycleview.....470

.....471

RecyclerView.....474

69: RecyclerView onClickListeners.....476

Examples.....476

.....476

Kotlin RxJava.....477

Easy OnLongClick OnClick.....478

-.....479

Click Listeners.....481

.....483

RecyclerView Click.....485

70: RecyclerView LayoutManagers.....487

Examples.....487

GridLayoutManager.....487

recyclerview gridlayout.....489

LinearLayoutManager 491
491
491
492
RecyclerView ViewHolder 492
( ) 494
RecyclerView PlaceListAdapter 495
! 495
StaggeredGridLayoutManager 495
71: Renderscript 498
498
Examples 498
498
498
RenderScript 499
RenderScript 499
RilerScript Boilerplate 500
501
501
501
API RenderScript Runtime 502
502
RenderScript Java 504
504
505
506
507
507
509
BlurBitmapTask.java 509
: 511

72: Retrofit2 512
..... 512
..... 512
Examples 512
GET 512
Retrofit2 515
Multipart 516
OkHttp 517
: 517
Retrofit multipart 518
Retrofit2 520
Stetho 523
Retrofit 2 Custom Xml Converter 523
POST GSON 526
URL- XML- 2 528
73: Retrofit2 RxJava 530
Examples 530
Retrofit2 RxJava 530
RxJava 531
: , ... 533
74: RoboGuice 535
Examples 535
..... 535
..... 535
@ContentView 535
@InjectResource 535
@InjectView 536
RoboGuice 536
75: Robolectric 539
..... 539
Examples 539
Robolectric test 539
.....

539

539

SDK 539

540

540

76: SearchView 541

Examples 541

AppView RxBindings 541

543

SearchView 546

77: SensorManager 548

Examples 548

548

549

, , 549

78: SharedPreferences 551

551

551

552

552

552

Examples 552

SharedPreferences 553

554

SharedPreferences 554

SharedPreferences 556

SharedPreferences 556

SharedPreferences Singleton 557

SharedPreferences 561

getPreferences (int) VS getSharedPreferences (String, int) 562

Commit vs. Apply 562

SharedPreferences 563

, , ... 563
pre-Honeycomb StringSet 564
EditTextPreference 566
79: ShortcutManager 567
Examples 567
..... 567
80: SpannableString 568
..... 568
Examples 568
TextView 568
, ... 571
81: SQLite 573
..... 573
..... 573
Examples 573
SQLiteOpenHelper 573
..... 574
onUpgrade () 575
..... 575
, SQLite Android 577
..... 581
..... 582
() 582
SQLite 583
..... 585
..... 587
..... 588
82: SyncAdapter 590
..... 590
Examples 590
..... 590
83: TabLayout 600

Examples 600
TabLayout ViewPager 600
84: TensorFlow 601
601
601
Examples 601
601
85: TextInputLayout 603
603
603
Examples 603
603
603
604
604
TextInputEditText 605
TextInputLayout 605
86: TextView 607
607
607
607
Examples 607
607
TextView 607
Spannable TextView 610
TextView 612
TextView 612
612
613
613
RelativeSizeSpan 615
Pinchzoom TextView 617

Single TextView 618

87: TransitionDrawable 620

Examples 620

..... 620

1. XML 620

2. ImageView XML-, ... 620

3: XML-, onCreate () 620

( ) TransitionDrawable 621

88: Typedef : @IntDef, @StringDef 622

..... 622

Examples 622

IntDef 622

..... 623

89: VectorDrawable AnimatedVectorDrawable 624

Examples 624

Basic VectorDrawable 624

..... 624

..... 625

..... 626

..... 627

AppCompat 629

90: VideoView 631

Examples 631

..... 631

URL VideoView 631

91: ViewFlipper 633

..... 633

Examples 633

ViewFlipper 633

92: ViewPager 635

..... 635

635
Examples 635
Basic ViewPager 635
ViewPager TabLayout 637
ViewPager PreferenceFragment 639
ViewPager 640
ViewPager 641
TabLayout ViewPager 642
TabLayout 642
selected_dot.xml 643
default_dot.xml 643
tab_selector.xml 643
OnPageChangeListener 643
93: WebView 645
645
645
Examples 645
JavaScript WebView - 645
Javascript Java (Android) 645
Java Javascript 647
647
WebView 648
webview logcat 648
Android Chrome 648
USB Android 649
Android- 649
/ Webview 649
94: XMPP 650
Examples 650
XMPP 650
95: Xposed 659
Examples 659

Xposed 659
..... 659
96: Youtube-API 662
..... 662
Examples 662
StandAlonePlayerActivity 662
, YouTubeBaseActivity 662
YoutubePlayerFragment 663
API YouTube Player 667
API YouTube Android 668
97: Zip- android 672
Examples 672
android 672
98: 674
..... 674
Examples 674
..... 674
..... 675
99: TextViews 676
..... 676
Examples 676
..... 676
..... 677
100: 678
Examples 678
ImageView 678
Fade in / out animation 679
TransitionDrawable 680
ValueAnimator 680
ObjectAnimator 681
ViewPropertyAnimator 682
View 682

101: AlertDialog 684
..... 684
Examples 684
... . 684
102: / (PTT, LWP . .) 687
..... 687
Examples 687
Sonim 687
PTT_KEY 687
YELLOW_KEY 687
SOS_KEY 687
GREEN_KEY 687
..... 688
RugGear 688
PTT 688
103: MVP 689
..... 689
..... 689
MVP 689
( ) 689
Examples 690
Presenter (MVP) 690
..... 693
: 694
MVP 695
..... 695
XML activity_login 695
LoginActivity.class 696
ILoginView 698
ILoginPresenter 698

ILoginPresenter.class 698
LoginPresenterCompl.class 698
UserModel 699
UserModel.class 699
IUser.class 700
MVP 700
104: 702
702
Examples 702
702
105: Firebase Realtime 704
704
: 704
Examples 704
Firebase Realtime DataBase 704
705
, Fi 705
1. 706
2: JSON 706
3: 706
4: 707
708
: 708
JSON Firebase 711
712
713
714
106: 716
Examples 716
- 716
107: 718

718
718
718
718
Examples 718
718
108: 720
720
720
720
720
Examples 720
720
109: 722
722
Examples 722
722
724
724
725
726
RecyclerView 727
727
XML- 727
727
728
729
731
DataBinding (int, boolean) 732
733
BindingAdapter 733
110: 735

Examples....735
....735
:....735
1:....735
2:....735
3:....735
4:....736
5:....736
6:....739
() In-App v3....740
111: Retrolambda Android....742
....742
Examples....742
:....742
112: 744
Examples....744
....744
....744
....745
Drawable....746
113: 749
....749
....749
....749
Examples....750
VectorDrawable....750
VectorDrawable xml....751
SVG- VectorDrawable....751
114: Android....754
....754
Examples....755

Android 756
115: SDK 757
..... 757
..... 757
..... 757
Examples 758
SDK 758
116: 759
Examples 759
..... 759
..... 759
..... 759
..... 760
..... 760
117: 761
..... 761
Examples 761
- 761
..... 761
AppWidgetProvider 761
..... 762
/ Android Studio 763
==> ==> ==> 763
118: 766
Examples 766
..... 766
Spinner 766
119: 769
Examples 769
..... 769
..... 770
GetCurrentRealTime 770

120: 772
Examples....772
DatePicker....772
....774
121: RecyclerView....776
....776
....776
....776
....776
....776
:....776
javadoc....776
Examples....777
....777
ItemDecoration....778
RecyclerView....779
DividerItemDecoration....781
ItemOffsetDecoration GridLayoutManager RecycleView....781
122: 783
....783
....783
....784
....784
Examples....784
back-stack....784
Android....785
launchMode....789
:....789
SingleTop:....789
SingleTask:....789
SingleInstance:....789

setContentView....790
....790
:....790
:....791
....791
....791
....792
123: 795
....795
Examples....795
....795
124: 797
....797
....797
Examples....797
....797
....798
DialogFragment....798
DatePickerDialog....801
DatePicker....801
DatePickerDialog....801
AlertDialog Appcompat....802
ListView AlertDialog....803
EditText....804
....805
....806
125: 809
....809
....809
Examples....809
AppCompat....809
....

810

FloatingActionButton (FAB) 812

Material Design 813

TextInputLayout 815

TabLayout 816

RippleDrawable 817

..... 822

..... 826

..... 826

..... 828

..... 829

126: FuseView Android 831

..... 831

Examples 831

hikr, android.view.View 831

127: SQLite ContentValues 840

Examples 840

SQLite 840

..... 840

..... 840

128: 841

Examples 841

..... 841

129: 842

..... 842

..... 842

..... 842

..... 842

Examples 843

..... 843

..... 843

844

.....845

: ?.....845

( ).....846

130: 847

.....847

.....847

.....847

.....847

.....847

Examples.....848

Basic StringRequest GET.....848

.....848

NetworkImageView.....849

JSON.....850

[, auth].....851

.....852

StringRequest POST.....853

Volley HTTP.....854

json 856

JSONArray 858

131: 859

Examples.....859

Android Studio.....859

Android-.....859

.....860

.....860

.....860

132: Android.....862

Examples.....862

api> 19.....862

SoundPool.....863

133: 865
Examples 865
..... 865
134: 866
..... 866
Examples 866
..... 866
..... 867
..... 868
..... 869
..... 869
AsyncTask 870
: 870
: 870
: 870
: 870
AsyncTaskLoader: 871
: 871
..... 872
135: 873
..... 873
Examples 873
View Activity 873
136: 875
..... 875
Examples 875
Bar 875
ProgressBar 875
..... 877
ProgressBar 880
ProgressBar 881

882
882
883
883
884
137: 887
887
Examples 887
- Crashlytics 887
-Crashlytics 887
IDE Fabric 888
ACRA 892
893
894
138: OpenCV Android Studio 896
896
Examples 896
896
139: Google 905
905
905
Examples 905
Google 905
140: Google Android 909
909
Examples 909
google Auth . ( ) 909
Google SignIn 909
141: Android Paypal 911
911
Examples 911

PayPal Android ... 911
142: ... 913
Examples ... 913
- ... 913
143: UIAutomator ... 916
... 916
... 916
Examples ... 916
UIAutomator ... 916
UIAutomatorViewer ... 917
UIAutomator ... 919
144: (I18N L10N) ... 920
... 920
... 920
Examples ... 920
: RTL ... 920
: RTL ... 921
: RTL ... 922
: ... 922
: ... 923
: ... 924
145: Java Java Native (JNI) ... 925
... 925
Examples ... 925
JNI ... 925
Java ... 926
JNI ... 927
146: ... 929
Examples ... 929
... 929
... 929
... 929

929
-930
931
147: Firebase 933
933
Examples 935
URL- Http 935
API AppIndexing 937
148: 940
Examples 940
NetworkOnMainThreadException 940
ActivityNotFoundException 941
OutOfMemoryError 941
DexException 942
UncaughtException 943
943
149: 945
Examples 945
AES 945
150: SparseArray 949
949
949
Examples 950
SparseArray 950
151: 952
Examples 952
952
AndroidManifest.xml 952
954
957
960
, 961

152: SurfaceView 964
..... 964
Examples 964
SurfaceView 964
153: 2 970
..... 970
..... 970
Examples 970
..... 970
..... 972
..... 972
@Subcomponent @Component (dependencies = {...}) 973
2 build.gradle 974
..... 975
154: 2: 977
..... 977
..... 977
Dagger 2 API: 977
: 977
Examples 978
@Module @Singleton 978
..... 978
@Modules @Inject 978
@Component 979
155: 980
Examples 980
, ... 980
..... 980
156: 982
..... 982
Examples 982

inline onClickListener 982
..... 982
click XML 983
..... 983
..... 984
..... 984
..... 984
..... 985
157: Android 990
..... 990
Examples 990
..... 990
AppCompatActivity 990
ViewModel LiveData 991
..... 992
LiveData 994
, ... 995
158: 997
..... 997
..... 997
..... 997
Examples 997
..... 997
159: Layout 999
..... 999
..... 999
Examples 999
..... 999
Layout.Behavior 999
..... 1000
XML 1000
..... 1000

SwipeDismissBehavior 1001
1001
160: 1003
Examples 1003
1003
161: Bitmap 1007
1007
1007
1007
Examples 1007
LRU 1007
162: Android 1009
Examples 1009
1009
Android 1009
«res» 1010
«res» 1011
1012
Android 1015
163: / Android 1019
1019
Examples 1019
DateUtils.formatDateTime () 1019
Android 1019
/ 1019
164: 1021
1021
1021
1021
LayoutParams Layout_ 1021
RelativeLayouts 1022

Examples 1023
LinearLayout 1023
RelativeLayout 1024
1026
1029
1031
FrameLayout 1032
CoordinatorLayout 1033
CoordinatorLayout Scrolling Behavior 1035
1036
LinearLayout 1038
LayoutParams 1039
165: Android Studio 1043
1043
Examples 1043
Instant Run 1044
1046
Instant Run 1046
166: - 1048
1048
1048
Examples 1050
1050
1050
1051
1052
1052
1052
1053
MediaPlayer 1053
1053
androidstudio 1055

167: **1058**

..... 1058

..... 1058

..... 1058

Examples 1058

..... 1058

..... 1059

..... 1059

1: 1059

2: 1060

! **1061**

, : 1061

168: **1063**

..... 1063

..... 1063

LocationManager **1063**

FusedLocationProviderApi **1064**

..... **1066**

Examples 1073

API 1073

w / LocationRequest **1073**

..... **1075**

LocationManager 1078

LocationManager 1079

..... 1081

..... 1084

BroadcastReceiver 1084

169: Android JUnit **1086**

..... 1086

Examples 1086

..... 1086

1086

1086

Android Studio 1087

Android Studio 1088

- Android 1088

JUnit 1090

1091

1092

1093

1094

1095

170: Dex 1097

1097

1097

dex? 1097

: 1097

: 1097

: 1098

Examples 1099

MultiDexApplication 1099

Multidex, 1099

Multidex 1100

1100

MultiDex 1100

(Gradle Dexcount) 1101

Multidex MultiDexApplication 1101

171: UI- - Android 1103

1103

1103

1103

JUnit: 1103

Appium ... 1103
... 1103
Examples ... 1104
MockWebServer ... 1104
IdlingResource ... 1107
... 1107
... 1107
... 1107
... 1108
JUnit ... 1109
172: Jenkins CI Android- ... 1111
Examples ... 1111
Jenkins Android ... 1111
I: ... 1111
II: Jenkins Android ... 1112
III: Jenkins Job Android- ... 1113
173: OpenGL ES 2.0+ ... 1115
... 1115
Examples ... 1115
GLSurfaceView OpenGL ES 2.0+ ... 1115
GLSL-ES ... 1116
174: ... 1118
... 1118
... 1118
... 1118
Examples ... 1118
... 1118
... 1119
175: ... 1120
... 1120
... 1120

Examples 1120
BottomSheetBehavior Google 1120
1127
1127
BottomSheetDialogFragment 1129
BottomSheetDialog 1129
Open BottomSheet DialogFragment 1130
176: Bluetooth 1131
1131
Examples 1131
BLE 1131
GATT 1132
1132
Gatt 1133
BLE 1134
Gatt 1135
177: 1137
Examples 1137
1137
1137
178: 1139
1139
1139
1139
Examples 1140
ButterKnife 1140
ButterKnife 1142
1142
1142
1142
1143
ViewHolder 1143

1143
1144
1144
ButterKnife 1144
ButterKnife 1145
Android Studio ButterKnife 1146
179: 1148
1148
1148
Examples 1148
1148
1149
(RealmList) 1150
- 1151
1151
1152
Realm RxJava 1152
1153
1153
1154
1154
1155
1156
1156
180: 1157
1157
Examples 1157
1157
1158
181: Shake Android 1160
Examples 1160

- Android- 1160
1161
1161
182: 1162
1162
1162
1162
<intent-filter> 1162
<data> 1163
1163
Examples 1163
1163
1163
1164
HTTP, https 1164
1165
pathPrefix 1165
183: 1167
1167
1167
Examples 1167
1167
1168
1169
184: 1171
1171
Examples 1171
@NonNull Annotation 1171
1171
1172
185: 1175
Examples 1175

Instagram OAuth 1175
186: 1177
1177
1177
Examples 1177
1177
1177
1179
Binder 1179
(AIDL) 1180
1182
187: 1185
Examples 1185
/ 1185
PNG- 1185
ByteStrings Buffers 1186
188: () RangeSeekBar 1187
1187
1187
Examples 1188
7 1188
189: 1189
1189
Examples 1189
ViewHolder 1189
190: Android 1191
Examples 1191
1191
1191
- 1193
191: 1195
1195

Examples 1195
ListView 1195
192: Play Store 1207
Examples 1207
1207
193: Cling Android 1209
Examples 1209
Cling Android- 1209
NAT 1209
194: 1211
1211
1211
Examples 1211
1211
195: 1214
1214
1214
Examples 1214
Picasso Android- 1214
Gradle. 1214
Maven: 1214
1215
1215
1215
Picasso 1217
1217
Picasso 1218
Picasso 1218
Picasso ImageGetter Html.fromHtml 1218
, 1220
196: 1222

1222

Examples 1222

..... 1222

JobService **1222**

JobService AndroidManifest.xml **1222**

..... **1223**

197: **1225**

..... 1225

..... 1225

..... 1225

Loaders **1226**

Examples 1226

AsyncTaskLoader 1226

AsyncTaskLoader 1227

..... 1229

Bundle 1229

198: , ... **1230**

..... 1230

..... 1230

..... 1230

..... 1230

..... 1231

..... 1231

..... 1231

..... 1231

..... 1231

dp dip 1231

..... 1231

Examples 1232

..... 1232

dp sp 1233

Android 1234
199: Wi-Fi 1235
Examples 1235
WEP- 1235
WPA2 1235
1236
200: Android 1239
1239
Examples 1239
1239
build.gradle 1240
201: 1242
Examples 1242
Firebase 1242
1243
202: Firebase 1244
1244
Examples 1244
CloudBase Firebase Android 1244
1245
, , , 1245
1247
1248
203: 1249
Examples 1249
1249
1249
px dp, dp to px 1249
204: 1251
1251
Examples 1251
1251

Paint 1251
..... 1251
..... 1252
Paint 1252
..... 1253
205: 1254
..... 1254
Examples 1254
..... 1254
TextView 1254
206: 1255
Examples 1255
..... 1255
..... 1255
TextView 1255
TextView xml (Java) 1255
..... 1256
..... 1257
..... 1257
Android O 1258
207: 1260
..... 1260
..... 1260
Examples 1260
CursorAdapter 1260
ArrayAdapter 1261
ListView ArrayAdapter 1262
208: 1264
..... 1264
Examples 1264
..... 1264
209: vietnamese Android 1269

Examples ... 1269
: ... 1269
Chuyn chui Ting Vit thành chui không du ... 1269
210: ... **1270**
Examples ... 1270
Google Google ... 1270
... 1271
211: ... **1273**
... 1273
Examples ... 1273
RecyclerView ... 1273
Swipe-to-Refresh ... 1274
212: ... **1275**
Examples ... 1275
... 1275
ConnectivityManager ... 1275
, ... 1275
213: ... **1277**
... 1277
... 1277
... 1277
... 1277
Examples ... 1277
, - ... 1277
Android? ... 1278
... 1278
214: ... **1282**
Examples ... 1282
... 1282
... 1282
215: Android Kotlin ... **1283**
... 1283

1283
Examples 1283
Kotlin 1283
Gradle Kotlin 1284
Kotlin 1286
Java- Kotlin 1288
1288
216: Google 1289
Examples 1289
1289
1289
217: Maven 1292
Examples 1292
.aar Maven 1292
218: .aar Apache Archiva Gradle 1294
Examples 1294
1294
219: Android 1296
Examples 1296
1296
220: RecyclerView 1297
1297
Examples 1297
MainActivity.java 1297
221: Android- 1302
1302
1302
Examples 1302
Canvas SurfaceView 1302
222: API-23 + 1309
1309

1309

Examples 1310

Android 6.0 1310

, URI 1311

1312

PermissionUtil 1314

, , 1315

1317

223: 1318

1318

Examples 1318

Google PlayRecognitionAPI 1318

PathSense 1320

224: Logcat 1323

1323

1323

1324

1324

1324

1324

Examples 1324

logcat 1324

1326

1326

1327

1327

: 1328

: 1328

: 1328

: 1328

: 1329

Logcat 1329
Logcat 1330
1331
Android Studio 1331
1334
225: 1335
Examples 1335
EditTexts 1335
InputType 1337
`inputype` 1339
SoftKeyboard 1340
Custom Edit Text 1341
226: PorterDuff 1344
1344
1344
Examples 1345
PorterDuff ColorFilter 1345
PorterDuff XferMode 1346
() , PorterDuffXfermo 1346
227: 1347
1347
Examples 1349
1349
Android 1350
228: 1351
Examples 1351
1351
1352
1353
1353
1354
1354

1355

«» 1356

Activity / Fragment 1357

strings.xml 1358

..... 1359

..... 1359

, ... 1360

9 1363

9-PATCH ANDROID UI 18 2011 1363

() 1366

strings.xml 1367

229: **1369**

Examples 1369

..... 1369

230: **1372**

Examples 1372

Dummy Sync Stub 1372

231: **1378**

..... 1378

..... 1378

Examples 1378

Glide 1378

..... 1379

ImageView 1379

RecyclerView ListView 1380

(ImageView) 1380

..... 1381

Glide 1382

..... 1383

..... 1383

ImageView 1384

Glide 1384

232: Singleton Toast 1386
..... 1386
..... 1386
..... 1386
..... 1387
Examples 1387
..... 1387
233: 1389
Examples 1389
API 1389
: 1391
234: Overlay ( ) Windows 1392
Examples 1392
..... 1392
WindowManager 1392
SYSTEM_ALERT_WINDOW Android 6.0 1393
235: Android 1394
Examples 1394
! 1394
Java 1394
..... 1394
..... 1394
236: 1396
Examples 1396
..... 1396
..... 1399
..... 1401
CustomView 1405
SVG / VectorDrawable drawableRight 1406
: custom_edit_drawable (prefix-c_d_e) 1406
build.gradle 1406

: c_e_d_compound_view.xml 1406
: attrs.xml 1407
: EditTextWithDrawable.java 1407
: 1408
: activity_main.xml 1408
: MainActivity.java 1408
Touch 1409
237: Android 1410
Examples 1410
1410
1411
, Jitpack.io 1411
238: 1413
1413
Examples 1413
1413
1415
1. 1415
2: 1415
3: 1416
239: - / 1418
Examples 1418
Split Screen, Android Nougat. 1418
240: : , . 1420
1420
1420
Examples 1420
StrictMode . 1420
, , SQLL 1420
241: 1422
1422

1422
Examples 1422
1422
1423
242: (TTS) 1425
Examples 1425
1425
TextToSpeech API 1427
243: DayNight (AppCompat v23.2 / API 14+) 1430
Examples 1430
DayNight 1430
244: , , 1432
Examples 1432
1432
, 1432
1432
Overscroll (API 21+) 1433
(API 21+) 1433
(API 23+) 1433
(API 19+) 1434
(API 21+) 1434
1434
1435
1436
245: 1438
1438
1438
1438
Examples 1438
1438
Espresso 1439
DrawerLayout 1439

1440
UI 1440
1441
1442
1443
1446
1446
onView 1447
1447
1449
EditText 1451
1451
1451
1451
246: 1454
1454
1454
1454
1454
: 1455
Examples 1455
1455
Toast 1455
1456
Toast (Application Wide) 1457
1458
Thread Toast (AsyncTask) 1458
247: 1459
Examples 1459
1459
: 1459
: 1459

, ... 1459
..... 1459
Android Marshmallow Heads Up: 1460
Android KitKat : 1461
Android 6.0 Marshmallow: 1461
Android 4.4.x KitKat: 1462
..... 1463
..... 1464
- 1465
, : 1466
, : 1466
, «Picasso» 1466
..... 1467
«» 1468
248: 1469
..... 1469
Examples 1469
..... 1469
HandlerThreads Threads 1469
..... 1469
( ) 1470
Runnable 1470
HandlerThread 1470
..... 1470
Handler (javax.swing.Timer) 1471
249: 1473
..... 1473
Examples 1473
, ... 1473
250: Android 1474
..... 1474
Examples 1474

1475
TabLayout 1477
251: 1479
1479
1479
1480
1480
1480
, singleTask singleTop 1481
Examples 1481
1481
1482
OriginActivity 1482
DestinationActivity 1482
1484
1484
: 1485
DetailActivity: 1485
, : 1486
URL- 1487
1487
1487
1488
1488
URI URI 1488
1489
CustomTabsIntent Chrome 1490
1490
1491
1492
1492

1493

Google , ...1494

Intent in Activity....1494

....1496

....1496

....1497

....1498

Parcelable....1498

....1500

Activity to Fragment....1501

252: 1503

....1503

Examples....1503

....1503

....1503

253: ADB....1505

Examples....1505

....1505

....1505

apk 1505

254: 1507

Examples....1507

....1507

1. :....1507

2. 1507

3. , ...1508

4. :....1508

5. , ...1508

6. java.util.Observer ( ):....1508

AsyncTask....1508

....1509

....1510

LeakCanary 1511
..... 1512
1: 1515
2: 1515
, , , ... 1518
255: **1520**
Examples 1520
/ 1520
256: **1523**
..... 1523
Examples 1523
..... 1523
..... 1524
257: **1525**
Examples 1525
..... 1525
..... 1525
String 1525
258: **1527**
..... 1527
Examples 1527
+ 1 (786) 1234 5678 1527
259: **1528**
..... 1528
..... 1528
..... 1529
..... **1529**
Examples 1529
newInstance () 1529
backstack 1531
Activity Fragment Bundle 1532
..... 1533

1533
, 1533
1534
1535
260: 1540
1540
1540
Examples 1540
Fresco 1540
OkHttp 3 Fresco 1541
JPEG Fresco DraweeController 1542
261: 1543
1543
1543
Examples 1543
1543
1544
Android: - 1545
SD- (SD) 1550
: 1551
262: 1554
Examples 1554
1554
263: - QR- 1555
1555
Examples 1555
QRCodeReaderView (Zxing) 1555
1555
1555
264: ProGuard? Android? 1557
1557

Examples 1557
proguard 1557
265: 1560
1560
Examples 1560
Singleton Class 1560
1561
1561
266: / 1563
1563
Examples 1563
AES 1563
267: 1565
1565
Examples 1565
1565
AVD Manager 1568
1569
1570
268: 1571
1571
1571
Examples 1571
Android. 1571
1574

Около

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

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

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

глава 1: Начало работы с Android

замечания

Если вы хотите узнать больше о настройке Android Gradle Plugin, просмотрите документацию по `android-gradle` .

Если вас интересуют альтернативные эмуляторы, вы можете посмотреть на Genymotion . Он обеспечивает свободный план и требует меньшего объема оперативной памяти.

Версии

Версия	Уровень API	Код версии	Дата выхода
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

Версия	Уровень API	Код версии	Дата выхода
4,2	17	`JELLY_BEAN_MR1`	2012-11-13
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` (Зефир)	2015-10-05
7,0	24	`N` (Нуга)	2016-08-22
7,1	25	`N_MR1` (Nougat MR1)	2016-10-04
8,0	26	`O` (Предварительный просмотр разработчика 4)	2017-07-24

Examples

Настройка Android Studio

Android Studio - это IDE для разработки Android, которая официально поддерживается и рекомендуется Google. Android Studio поставляется в комплекте с Android SDK Manager , который является инструментом для загрузки компонентов `Android SDK` необходимых для начала разработки приложений.

Установка инструментов Android Studio и `Android SDK` :

1. Загрузите и установите Android Studio .
2. Загрузите последние инструменты SDK Tools и SDK Platform, открыв Android Studio, а затем следуя инструкциям Android SDK Tool Updates . Вы должны установить последние доступные стабильные пакеты.

   Если вам нужно работать над старыми проектами, которые были созданы с использованием старых версий SDK, возможно, вам также придется загрузить эти версии

Начиная с Android Studio 2.2, копия последней версии OpenJDK поставляется вместе с установкой и является рекомендуемым JDK (Java Development Kit) для всех проектов Android Studio. Это устраняет необходимость установки пакета Oracle JDK. Чтобы

использовать пакет SDK, выполните следующие действия:

1. Откройте проект в Android Studio и выберите « **Файл»> «Структура проекта»** в строке меню.
2. На странице **местоположения SDK** и в **местоположении JDK** установите флажок **Использовать встроенный JDK** .
3. Нажмите « **OK»** .

Настройка Android Studio

Android Studio предоставляет доступ к двум файлам конфигурации через меню « **Справка»** :

- studio.vmoptions : Настроить параметры виртуальной машины Java Java (JVM), такие как размер кучи и размер кеша. Обратите внимание, что на машинах Linux этот файл можно назвать *studio64.vmoptions* , в зависимости от вашей версии Android Studio.
- idea.properties : Настройте свойства Android Studio, такие как путь к папке плагинов или максимальный размер поддерживаемого файла.

Изменить / добавить тему

Вы можете изменить его как свое предпочтение. `File->Settings->Editor->Colors & Fonts->` и выберите тему. Также вы можете загружать новые темы из http://color-themes.com/. После того, как вы загрузили файл `.jar.zip` , перейдите в `File -> Import Settings...` и выберите загруженный файл.

Компиляция приложений

Создайте новый проект или откройте существующий проект в Android Studio и нажмите зеленую кнопку воспроизведения  на верхней панели инструментов, чтобы запустить его. Если он серый, вам нужно подождать секунду, чтобы Android Studio могла правильно индексировать некоторые файлы, прогресс которых можно увидеть в нижней строке состояния.

Если вы хотите создать проект из оболочки, убедитесь, что у вас есть файл `local.properties` , который создается Android Studio автоматически. Если вам нужно создать проект без Android Studio, вам потребуется строка, начинающаяся с `sdk.dir=` а затем путь к установке SDK.

Откройте оболочку и зайдите в каталог проекта. Введите `./gradlew aR` и нажмите enter. `aR` - это ярлык для `assembleRelease` , который загрузит все зависимости для вас и создаст приложение. Окончательный файл APK будет находиться в `ProjectName/ModuleName/build/outputs/apk` и будет называться `ModuleName-release.apk` .

Настройка Android Studio

Начните с настройки Android Studio и затем откройте ее. Теперь вы готовы сделать свое первое Android-приложение!

> **Примечание.** Данное руководство основано на Android Studio 2.2, но процесс в других версиях в основном одинаков.

Настроить свой проект

Основная конфигурация

Вы можете запустить новый проект двумя способами:

- Нажмите « `Start a New Android Studio Project` с экрана приветствия.
- Перейдите к `File` → `New Project` если у вас уже открыт проект.

Затем вам нужно описать свое приложение, заполнив некоторые поля:

1. **Имя приложения** - это имя будет показано пользователю.

   Пример: `Hello World` . Вы всегда можете изменить его позже в файле `AndroidManifest.xml` .

2. **Company Domain** - это определитель имени пакета вашего проекта.

   Пример: `stackoverflow.com` .

3. **Имя пакета** (aka `applicationId`) - это *полное* имя пакета проекта.

   Он должен следовать за *реверсивным именем имени домена* (aka *Reverse DNS*): *домен верхнего уровня* **.** *Домен компании* **.** [*Сегмент компании* **.**] *Имя приложения* .

   Пример: `com.stackoverflow.android.helloworld` или `com.stackoverflow.helloworld` . Вы всегда можете изменить свой *applicationId* , переопределив его в файле gradle .

   Не используйте префикс по умолчанию «com.example», если вы не намерены отправлять свое приложение в Google Play Store. Имя пакета будет вашим уникальным **приложением** в Google Play.

4. **Местоположение проекта** - это каталог, в котором будет сохранен ваш проект.

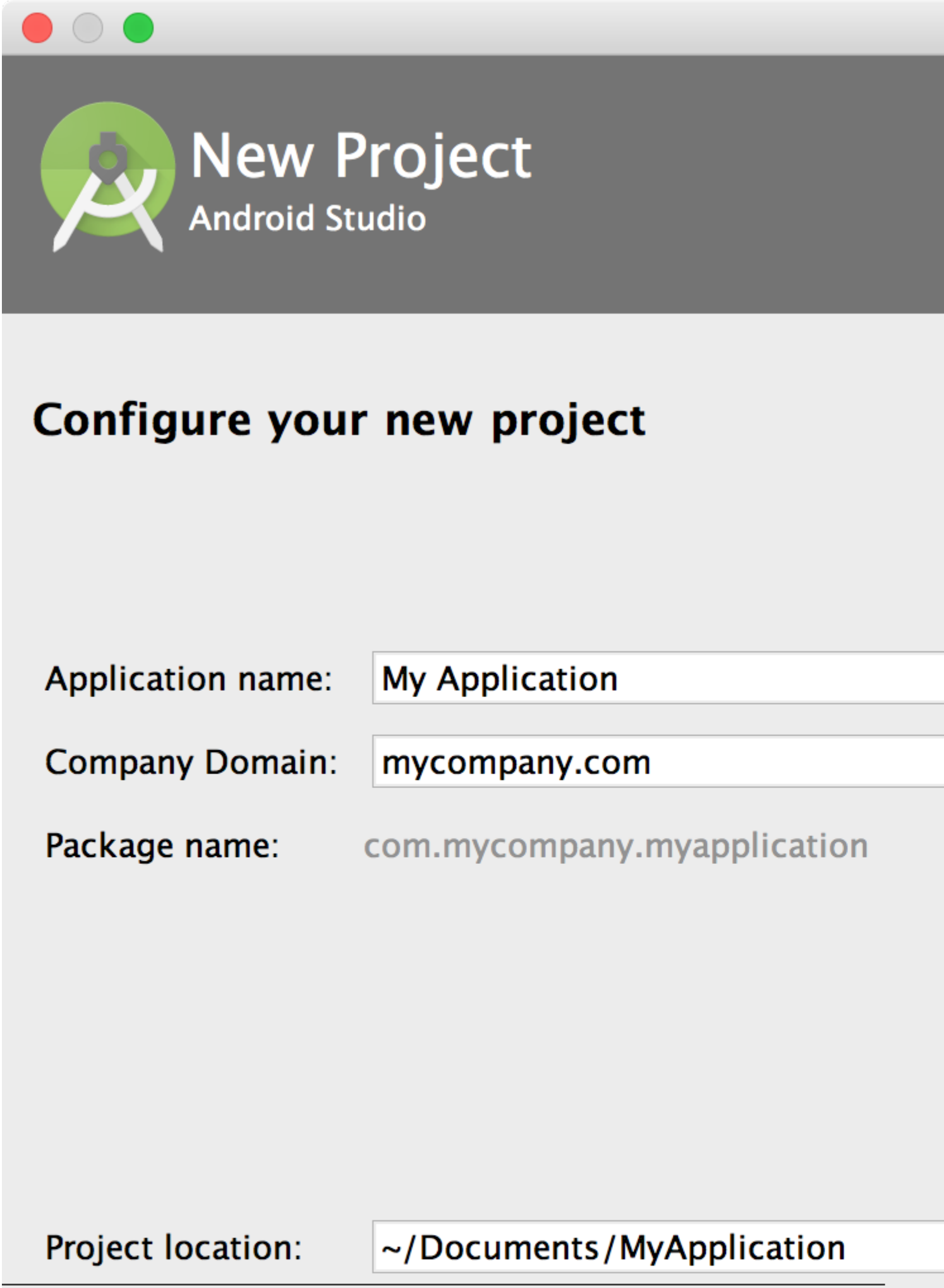
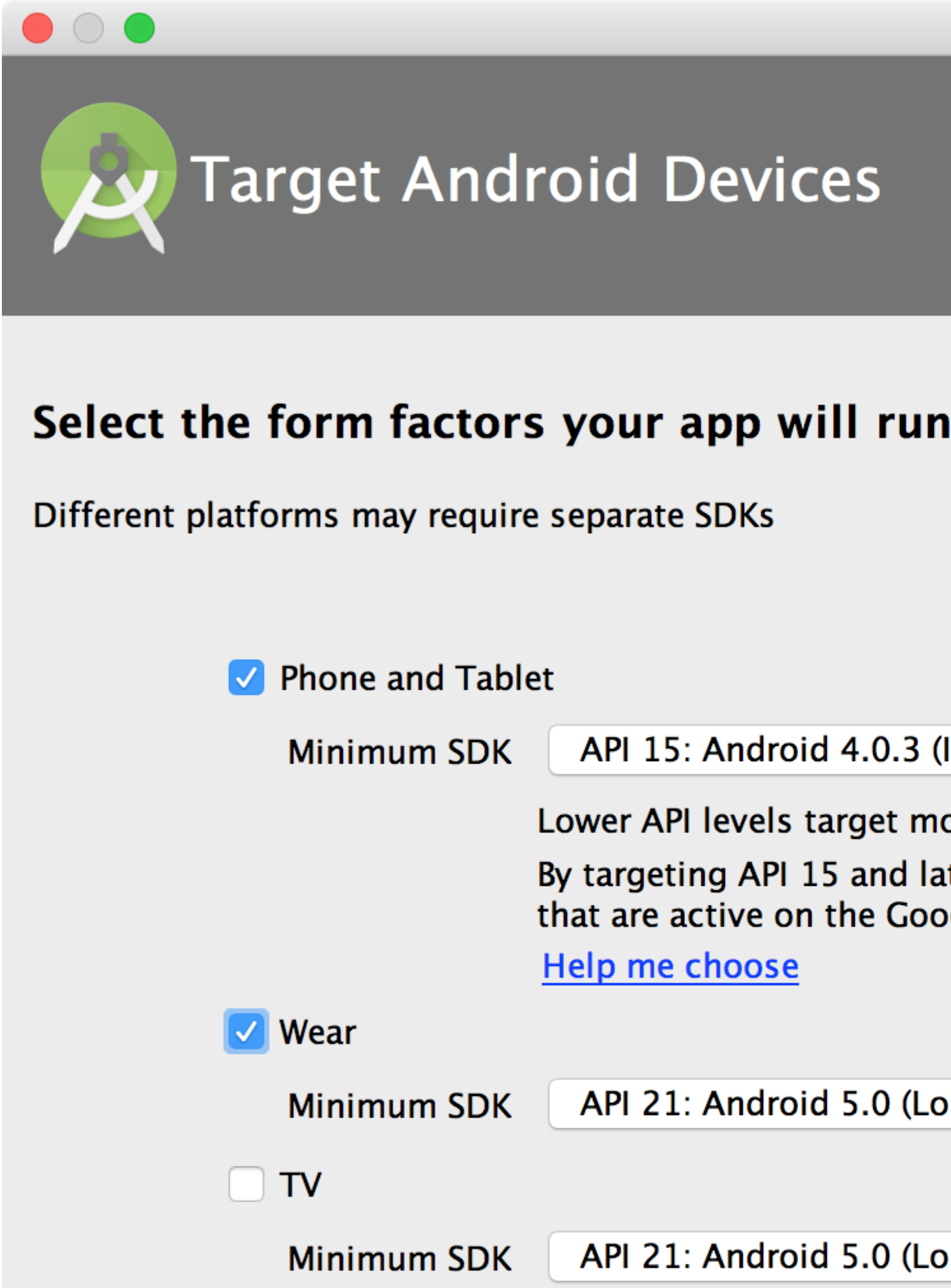

New Project
Android Studio
Configure your new project
Application name:
My Application
Company Domain:
mycompany.com
Package name:
com.mycompany.myapplication
Project location:
~/Documents/MyApplication

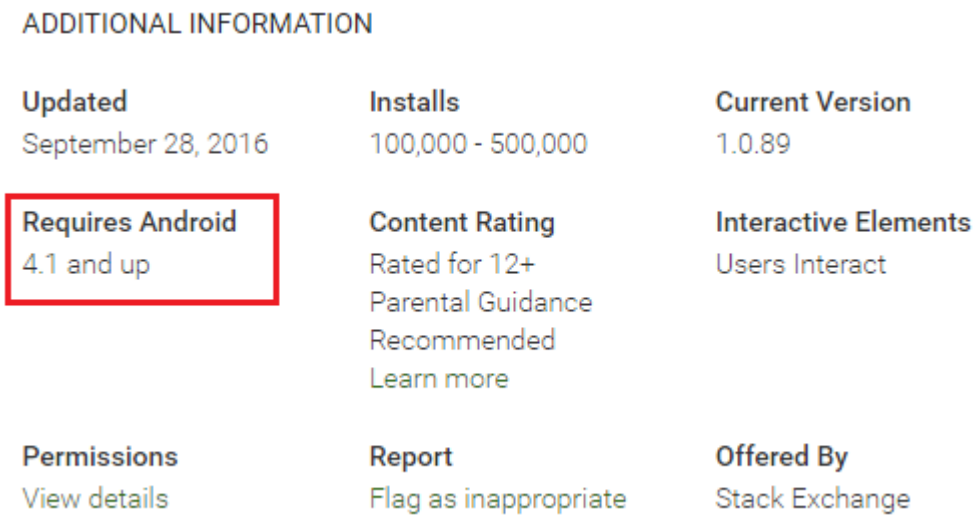
Диаграмма текущих дистрибутивов версий Android, отображаемая при нажатии кнопки «Справка».

В окне «Платформа Android Platform» показано распределение мобильных устройств, работающих с каждой версией Android, как показано на рисунке 2. Нажмите на уровень API, чтобы просмотреть список функций, представленных в соответствующей версии Android. Это поможет вам выбрать минимальный уровень API, который обладает всеми функциями, которые нужны вашим приложениям, поэтому вы можете охватить как можно больше устройств. Затем нажмите « **OK»** .

Теперь выберите, какие платформы и версия Android SDK будет поддерживать приложение.


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

нижняя граница для вашего приложения. Это один из сигналов, которые использует `Google Play Store` для определения того, на каких устройствах может быть установлено приложение. Например, приложение Stack Exchange поддерживает Android 4.1+.



Android Studio сообщит вам (приблизительно), какой процент устройств будет поддерживаться с учетом заданного минимального SDK.

> Более низкие уровни API нацелены на большее количество устройств, но имеют меньше доступных функций.

При принятии решения о **минимальном SDK** вы должны учитывать статистику Dashboards , которая даст вам информацию о версиях устройств, которые посетили Google Play Store по всему миру на прошлой неделе.

С: Панели мониторинга на веб-сайте разработчика Android.

Добавить деятельность

Теперь мы собираемся выбрать действие по умолчанию для нашего приложения. В Android `Activity` представляет собой единый экран, который будет представлен пользователю.

Приложение может размещать несколько действий и перемещаться между ними. В этом примере выберите « `Empty Activity` и нажмите «Далее».

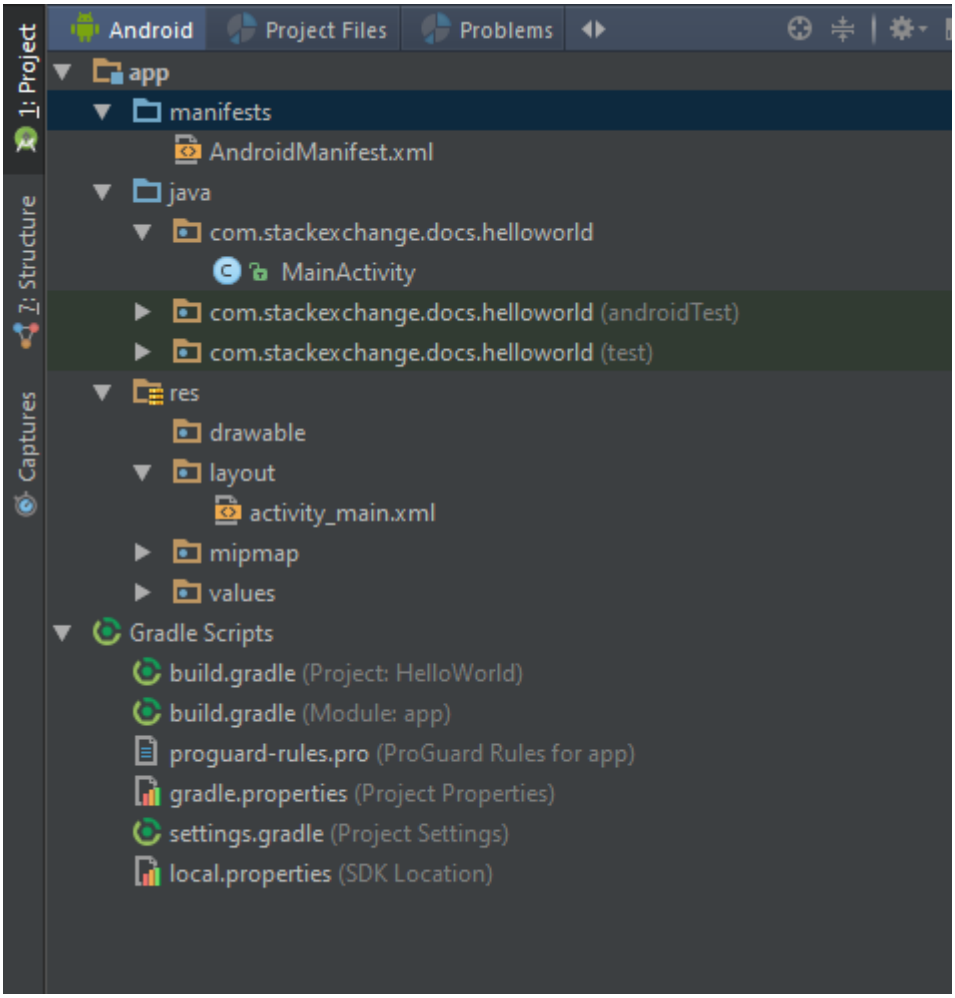
Здесь, если хотите, вы можете изменить название операции и макета. Хорошей практикой является сохранение `Activity` как суффикса для имени действия, а `activity_` в качестве префикса для имени макета. Если оставить их по умолчанию, Android Studio будет генерировать для нас деятельность, называемую `MainActivity` , и файл макета, называемый `activity_main` . Теперь нажмите « `Finish` .

Android Studio создаст и настроит наш проект, который может занять некоторое время в зависимости от системы.

Проверка проекта

Чтобы понять, как работает Android, давайте посмотрим на некоторые из файлов, которые были созданы для нас.

На левой панели Android Studio мы можем увидеть структуру нашего приложения для Android .



Сначала откройте `AndroidManifest.xml` , дважды щелкнув по нему. Файл манифеста Android

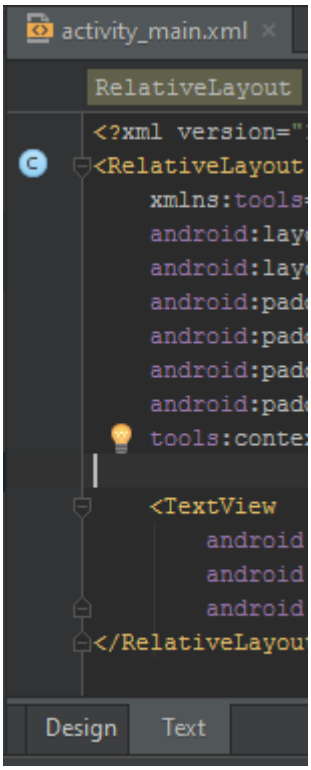
описывает некоторые основные сведения об Android-приложении. Он содержит декларацию о нашей деятельности, а также некоторые более продвинутые компоненты.

Если для приложения требуется доступ к функции, защищенной разрешением, она должна заявить, что для этого требуется это разрешение с элементом `<uses-permission>` в манифесте. Затем, когда приложение установлено на устройстве, установщик определяет, следует ли предоставлять запрашиваемое разрешение, проверяя полномочия, которые подписали сертификаты приложения, а в некоторых случаях и спрашивают пользователя. Приложение также может защищать свои собственные компоненты (действия, службы, широковещательные приемники и контент-провайдеры) с разрешениями. Он может использовать любые разрешения, определенные Android (перечисленные в android.Manifest.permission) или объявленные другими приложениями. Или он может определить свой собственный.

```
<?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>
```

Затем откройте файл `activity_main.xml` который находится в `app/src/main/res/layout/` . Этот файл содержит декларации для визуальных компонентов нашей MainActivity. Вы увидите визуального дизайнера. Это позволяет перетаскивать элементы на выбранный макет.

Вы также можете переключиться на конструктор макетов xml, нажав «Текст» внизу Android Studio, как показано здесь:




```
<?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>
```

Вы увидите виджет под названием `TextView` внутри этого макета, а свойство `android:text` установлено в «Hello World!». Это блок текста, который будет показан пользователю при запуске приложения.

Вы можете больше узнать о макетах и атрибутах .

Затем давайте посмотрим на `MainActivity` . Это код Java, который был создан для `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);
    }
}
```

Как определено в нашем Android-манифестах, `MainActivity` будет запускаться по умолчанию, когда пользователь запустит приложение `HelloWorld` .

Наконец, откройте файл с именем `build.gradle` расположенный в `app/` .
Android Studio использует систему сборки **Gradle** для компиляции и сборки приложений и библиотек 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'
}
```

Этот файл содержит информацию о версии сборки и вашего приложения, и вы также можете использовать ее для добавления зависимостей к внешним библиотекам. Пока что не будем вносить никаких изменений.

Целесообразно всегда выбирать последнюю версию, доступную для зависимостей:

- buildToolsVersion : 26.0.0
- com.android.support:appcompat-v7 : 26.0.0 (июль 2017 г.)
- firebase : 11.0.4 (август 2017)

compileSdkVersion

`compileSdkVersion` - это ваш способ сказать *Gradle,* какую версию Android SDK нужно скомпилировать. Использование нового Android SDK - это требование использовать любой из новых API, добавленных на этом уровне.

Следует подчеркнуть, что изменение вашей команды `compileSdkVersion` не изменяет поведение во время выполнения. Хотя при изменении вашей `compileSdkVersion` могут появляться новые предупреждения / ошибки `compileSdkVersion` , ваша `compileSdkVersion` не включена в ваш APK: она используется исключительно во время компиляции.

Поэтому настоятельно рекомендуется всегда компилировать последнюю версию SDK. Вы получите все преимущества новых проверок компиляции существующего кода, избегайте новых устаревших API и будьте готовы использовать новые API.

minSdkVersion

Если `compileSdkVersion` устанавливает новейшие API-интерфейсы для вас, `minSdkVersion` является *нижней границей* для вашего приложения. `minSdkVersion` является одним из сигналов, которые использует Google Play Store для определения того, какие из устройств пользователя могут быть установлены.

Он также играет важную роль во время разработки: по умолчанию lint запускается против вашего проекта, предупреждая вас, когда вы используете какие-либо API выше вашего `minSdkVersion` , помогая вам избежать проблемы во время выполнения попытки вызова API, который не существует. Проверка версии системы во время выполнения является распространенным методом при использовании API только в новых версиях платформы.

targetSdkVersion

`targetSdkVersion` - это основной способ, с помощью которого Android обеспечивает передовую совместимость, не применяя изменения поведения, если не будет обновлена `targetSdkVersion` . Это позволяет вам использовать новые API до того, как они будут работать с изменениями поведения. Обновление для установки последней версии SDK должно быть приоритетным для каждого приложения. Это не означает, что вам нужно использовать каждую новую введенную функцию и не слепо обновлять `targetSdkVersion` без тестирования.

targetSDKVersion - это версия Android, которая является верхним пределом доступных инструментов. Если targetSDKVersion меньше 23, приложение не нуждается в запросе разрешений во время выполнения для экземпляра, даже если приложение запускается на API 23+. TargetSDKVersion **не** предотвращает запуск версий Android над выбранной

версией Android.

Дополнительную информацию о плагине Gradle можно найти:

- Основной пример
- Введение в плагин Gradle для Android и обертки
- Введение в конфигурацию методов build.gradle и DSL

Запуск приложения

Теперь давайте запустим наше приложение HelloWorld. Вы можете либо запустить Android Virtual Device (которое можно настроить с помощью AVD Manager в Android Studio, как описано в примере ниже), либо подключить собственное устройство Android через USB-кабель.

Настройка Android-устройства

Чтобы запустить приложение из Android Studio на устройстве Android, вы должны включить `USB Debugging` в `Developer Options` в настройках вашего устройства.

`Settings > Developer options > USB debugging`

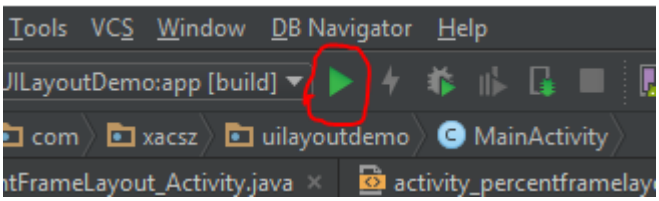
Если `Developer Options` не отображаются в настройках, перейдите в « `About Phone` и коснитесь номера `Build Number` семь раз. Это позволит включить `Developer Options` разработчика в ваших настройках.

`Settings > About phone > Build number`

Вам также может потребоваться изменить конфигурацию `build.gradle` чтобы построить версию, которую имеет ваше устройство.

Запуск из Android Studio

Нажмите зеленую кнопку « `Run` на панели инструментов в верхней части Android Studio. В появившемся окне выберите любое устройство, на которое вы хотите запустить приложение (при необходимости запустите виртуальное устройство Android или см. Настройка AVD (Android Virtual Device), если вам нужно установить его вверх) и нажмите « `OK` .



На устройствах под управлением Android 4.4 (KitKat) и, возможно, выше, появится всплывающее окно для авторизации USB-отладки. Нажмите « `OK` для подтверждения.

Приложение теперь будет установлено и запущено на вашем устройстве Android или эмуляторе.

Расположение файла APK

Когда вы готовите приложение к выпуску, вы настраиваете, создаете и тестируете версию своего приложения. Задачи настройки просты, включая основные операции очистки кода и изменения кода, которые помогают оптимизировать ваше приложение. Процесс сборки похож на процесс сборки отладки и может быть выполнен с использованием инструментов JDK и Android SDK. Задачи тестирования служат окончательной проверкой, гарантируя, что ваше приложение будет работать в соответствии с реальными условиями. Когда вы закончите подготовку своего приложения к выпуску, у вас есть подписанный файл APK, который вы можете распространять непосредственно пользователям или распространять через рынок приложений, например Google Play.

Android Studio

Поскольку в приведенных выше примерах используется Gradle, расположение сгенерированного файла APK: `<Your Project Location>/app/build/outputs/apk/app-debug.apk`

IntelliJ

Если вы являетесь пользователем IntelliJ перед переключением на Studio и импортируете проект IntelliJ напрямую, то ничего не изменилось. Расположение выхода будет таким же:

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

Примечание: это будет устаревать иногда около 1.0

Затмение

Если вы импортируете проект Android Eclipse напрямую, не делайте этого! Как только у вас есть зависимости в вашем проекте (банки или проекты библиотек), это не сработает, и ваш проект будет неправильно настроен. Если у вас нет зависимостей, apk будет находиться в том же месте, что и в Eclipse:

```
bin/...
```

Android-программирование без IDE

Это минималистский пример Hello World, который использует только самые базовые инструменты для Android.

Требования и допущения

- Oracle JDK 1.7 или новее
- Android SDK Tools (только инструменты командной строки)

Этот пример предполагает Linux. Возможно, вам придется настроить синтаксис для своей собственной платформы.

Настройка Android SDK

После распаковки версии SDK:

1. Установите дополнительные пакеты с помощью диспетчера SDK. Не используйте `android update sdk --no-ui` как указано в комплекте Readme.txt; он загружает около 30 ГБ ненужных файлов. Вместо этого используйте интерактивный SDK-менеджер `android sdk` чтобы получить рекомендуемый минимум пакетов.

2. Приложите следующие каталоги JDK и SDK к вашему исполнению PATH. Это необязательно, но приведенные ниже инструкции предполагают это.

    - JDK / бен
    - SDK / платформенные инструменты
    - SDK / инструменты
    - SDK / build-tools / LATEST *(как установлено на шаге 1)*

3. Создайте виртуальное устройство Android. Используйте интерактивный AVD Manager (`android avd`). Возможно, вам придется немного поиграть и найти совет; инструкции на месте не всегда полезны.

    (Вы также можете использовать свое устройство)

4. Запустите устройство:

    ```
    emulator -avd DEVICE
    ```

5. Если экран устройства заблокирован, проведите по экрану, чтобы разблокировать его.

    Оставьте его работать, пока вы программируете приложение.

Кодирование приложения

6. Перейдите в пустой рабочий каталог.

7. Сделайте исходный файл:

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

   Содержание:

```
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. Добавьте манифест:

```
touch AndroidManifest.xml
```

   Содержание:

```
<?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. Создайте подкаталог для объявленных ресурсов:

```
mkdir res
```

   Оставьте его пустым.

Построение кода

10. Создайте источник для объявлений ресурсов. Замените здесь правильный путь к

вашему **SDK** и установленному **API** для создания против (например, «android-23»):

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

Объявления ресурсов (описанные ниже) фактически являются необязательными. Между тем вышеупомянутый вызов ничего не делает, если res / все еще пуст.

11. Скомпилируйте исходный код в байт-код Java (.java → .class):

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

12. Переведите байт-код с Java на Android (.class → .dex):

    Сначала используйте Jill (.class → .jayce):

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

    Затем Jack (.jayce → .dex):

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

    Байт-код Android раньше назывался «исполняемым кодом Dalvik», и поэтому «dex».

    Вы можете заменить шаги 11 и 12 одним звонком на Джек, если хотите; он может компилироваться непосредственно из источника Java (.java → .dex). Но есть преимущества для компиляции с `javac` . Это более известный, хорошо документированный и более широко используемый инструмент.

13. Пакет файлов ресурсов, включая манифест:

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

    Это приводит к частичному APK-файлу (пакет приложений для Android).

14. Сделайте полный APK с `ApkBuilder` инструмента `ApkBuilder` :

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

```
  app.apkUnalign \
  -d -f classes.dex -v -z app.apkPart
```

Он предупреждает: «ЭТО ИНСТРУМЕНТ УДАЛЕН. См.« Справка »для получения дополнительной информации». Если `--help` завершился с ошибкой `ArrayIndexOutOfBoundsException` , вместо этого не передайте аргументы:

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

В нем объясняется, что CLI (`ApkBuilderMain`) устарел в пользу прямого вызова Java API (`ApkBuilder`). (Если вы знаете, как это сделать из командной строки, обновите этот пример.)

15. Оптимизируйте выравнивание данных APK (рекомендуемая практика):

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

Установка и запуск

16. Установите приложение на Android-устройство:

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

17. Запустите приложение:

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

Он должен бежать и приветствовать.

Это все. Это то, что нужно, чтобы поздороваться с базовыми инструментами Android.

Объявление ресурса

Этот раздел не является обязательным. Заявки ресурсов не требуются для простого приложения «привет мир». Если они также не требуются для вашего приложения, вы можете упростить сборку, опуская шаг 10 и удалив ссылку на каталог res / с шага 13.

В противном случае, вот краткий пример того, как объявить ресурс и как ссылаться на него.

18. Добавить файл ресурсов:

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

Содержание:

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

19. Ссылка на ресурс из манифеста XML. Это декларативный стиль ссылки:

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

20. Ссылка на тот же ресурс из источника Java. Это настоятельная рекомендация:

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

21. Проверьте приведенные выше изменения, переустановив, переустановив и перезапустив приложение (шаги 10-17).

    Он должен перезапустить и сказать: «Это приложение называется Saying hello».

Удаление приложения

```
adb uninstall dom.domain
```

Смотрите также

- исходный вопрос - оригинальный вопрос, который вызвал этот пример
- рабочий пример - рабочий скрипт сборки, который использует приведенные выше команды

Основы применения

Приложения для Android написаны на Java. Инструменты Android SDK скомпилируют файлы кода, данных и ресурсов в пакет APK (Android). Как правило, один файл APK содержит весь контент приложения.

Каждое приложение запускается на собственной виртуальной машине (VM), чтобы приложение могло работать отдельно от других приложений. Система Android работает с

принципом наименьших привилегий. Каждое приложение имеет доступ только к тем компонентам, которые ему необходимы для выполнения своей работы, и не более того. Тем не менее, есть способы, по которым приложение может обмениваться данными с другими приложениями, например, путем совместного использования идентификатора пользователя Linux между приложением, или приложения могут запрашивать разрешение на доступ к данным устройства, таким как SD-карта, контакты и т. Д.

Компоненты приложения

Компоненты приложения - это строительные блоки приложения для Android. Каждый компонент играет определенную роль в приложении для Android, которое выполняет определенную задачу и имеет различные жизненные циклы (поток того, как и когда компонент создается и уничтожается). Вот четыре типа компонентов приложения:

1. **Действия:** Действие представляет собой один экран с пользовательским интерфейсом (UI). Приложение Android может иметь более одного действия. (например, приложение электронной почты может иметь одно действие для перечисления всех электронных писем, другое - для отображения содержимого каждого электронного письма, а другое - для создания нового сообщения электронной почты.) Все действия в приложении работают вместе для создания пользовательского eXperience (UX).
2. **Службы:** служба работает в фоновом режиме для выполнения длительных операций или для выполнения работы для удаленных процессов. Служба не предоставляет никакого пользовательского интерфейса, она работает только в фоновом режиме с помощью ввода пользователя. (например, служба может воспроизводить музыку в фоновом режиме, когда пользователь находится в другом приложении, или может загружать данные из Интернета, не блокируя взаимодействие пользователя с устройством Android.)
3. **Поставщики** контента. Поставщик контента управляет данными общего доступа. Существует четыре способа хранения данных в приложении: он может быть записан в файл и сохранен в файловой системе, вставлен или обновлен в базу данных SQLite, отправлен в Интернет или сохранен в любом другом постоянном хранилище, к которому приложение может получить доступ , Через контент-провайдеры другие приложения могут запрашивать или даже изменять данные. (например, система Android предоставляет провайдера контента, который управляет контактной информацией пользователя, чтобы любое приложение, имеющее разрешение, могло запрашивать контакты.) Поставщики контента также могут использоваться для сохранения данных, которые являются приватными для приложения, для лучшей целостности данных.
4. **Широковещательные приемники:** широковещательный приемник реагирует на широковещательные сообщения в масштабе всей системы (например, широковещательная передача, показывающая, что экран выключен, батарея разряжена и т. Д.) Или из приложений (например, чтобы другие приложения знали,

что некоторые данные были загружается на устройство и доступно для них). У широковещательных приемников нет пользовательских интерфейсов, но они могут показывать уведомление в строке состояния, чтобы предупредить пользователя. Обычно широковещательные приемники используются в качестве шлюза для других компонентов приложения, состоящего в основном из видов деятельности и услуг.

Одним из уникальных аспектов системы Android является то, что любое приложение может запускать компонент другого приложения (например, если вы хотите позвонить, отправить SMS, открыть веб-страницу или просмотреть фотографию, есть приложение, которое уже делает это, и ваше приложение может использовать его вместо того, чтобы разрабатывать новую деятельность для одной и той же задачи).

Когда система запускает компонент, он запускает процесс для этого приложения (если он еще не запущен, то есть только один процесс переднего плана для каждого приложения может запускаться в любой момент времени в системе Android) и создает классы, необходимые для этого компонента. Таким образом, компонент работает на процессе этого приложения, к которому он принадлежит. Поэтому, в отличие от приложений на других системах, приложения Android не имеют ни одной точки входа (нет метода `main()`).

Поскольку система запускает каждое приложение в отдельном процессе, одно приложение не может напрямую активировать компоненты другого приложения, однако система Android может. Таким образом, чтобы запустить компонент другого приложения, одно приложение должно отправить сообщение в систему, которое указывает на намерение запуска этого компонента, затем система запустит этот компонент.

контекст

Экземпляры класса `android.content.Context` обеспечивают подключение к системе Android, которая выполняет приложение. Экземпляр контекста необходим для доступа к ресурсам проекта и глобальной информации о среде приложения.

Давайте рассмотрим простой пример: подумайте, что вы в отеле, и вы хотите что-то съесть. Вы звоните в номер-сервис и попросите их принести вам вещи или убрать вещи для вас. Теперь подумайте об этом отеле как о приложении для Android, о себе как о деятельности, а человек, обслуживающий комнату, - это ваш контекст, который предоставляет вам доступ к ресурсам отеля, таким как обслуживание в номерах, продукты питания и т. Д.

Еще один пример: вы находитесь в ресторане, сидящем на столе, в каждом столе есть помощник, когда вы хотите заказать продукты, которые вы попросите у сопровождающего сделать это. Затем сопровождающий размещает ваш заказ, и ваши продукты питания подаются на вашем столе. Опять же, в этом примере ресторан представляет собой приложение для Android, таблицы или клиенты - это компоненты приложения, продукты питания - это ресурсы вашего приложения, а сопровождающий - ваш контекст, что дает

вам доступ к ресурсам, таким как продукты питания.

Для активации любого из вышеуказанных компонентов требуется экземпляр контекста. Не только только выше, но почти каждый системный ресурс: создание пользовательского интерфейса с использованием представлений (обсуждается позже), создание экземпляра системных сервисов, запуск новых действий или служб - все требует контекста.

Более подробное описание написано здесь .

Настройка AVD (Android Virtual Device)

TL; DR В основном это позволяет нам имитировать реальные устройства и тестировать наши приложения без реального устройства.

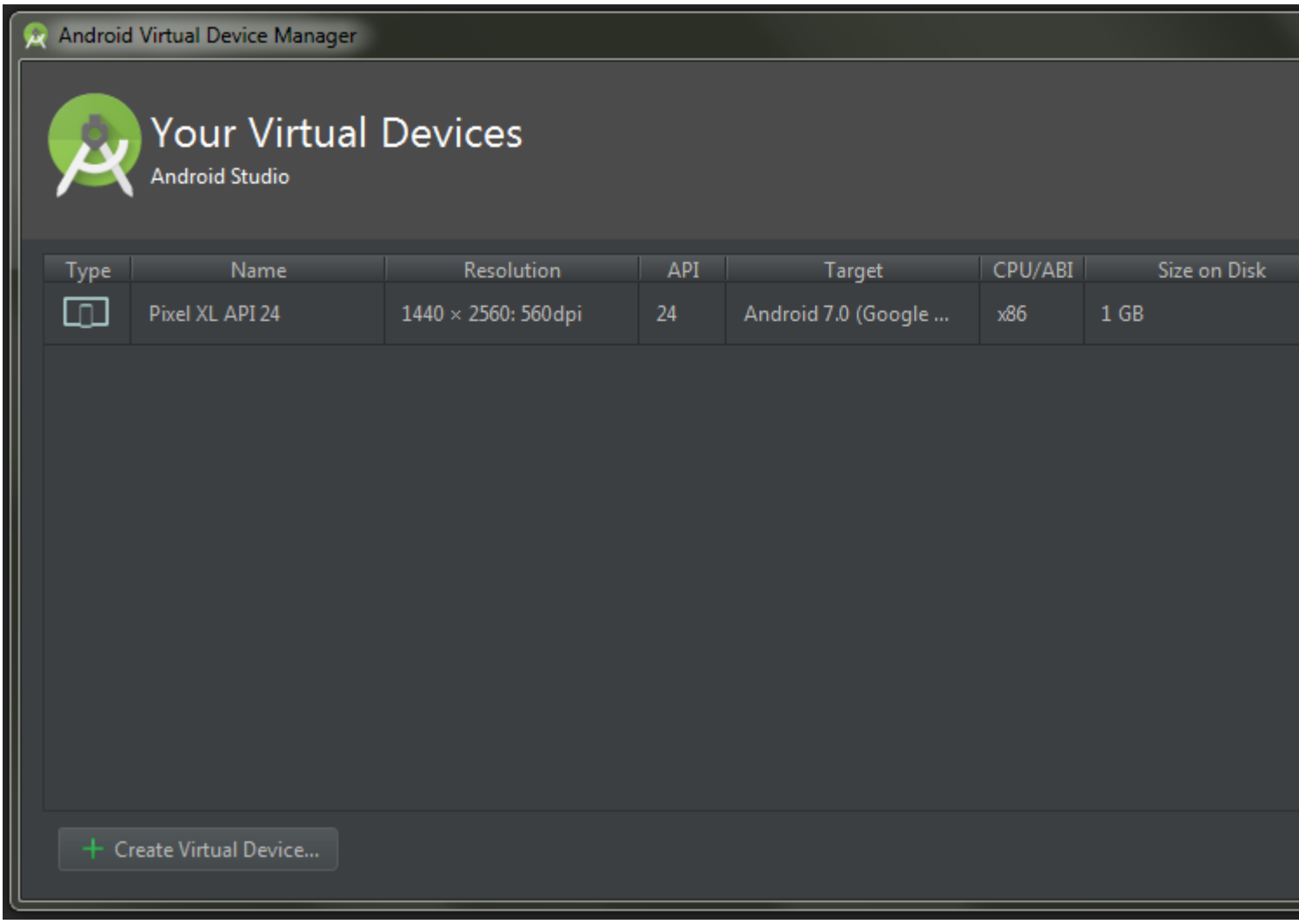
Согласно документации разработчика Android ,

> определение ***виртуального устройства Android (AVD)*** позволяет определить характеристики Android Phone, Tablet, Android Wear или Android TV, которые вы хотите имитировать в Android-эмуляторе. AVD Manager помогает вам легко создавать и управлять AVD.

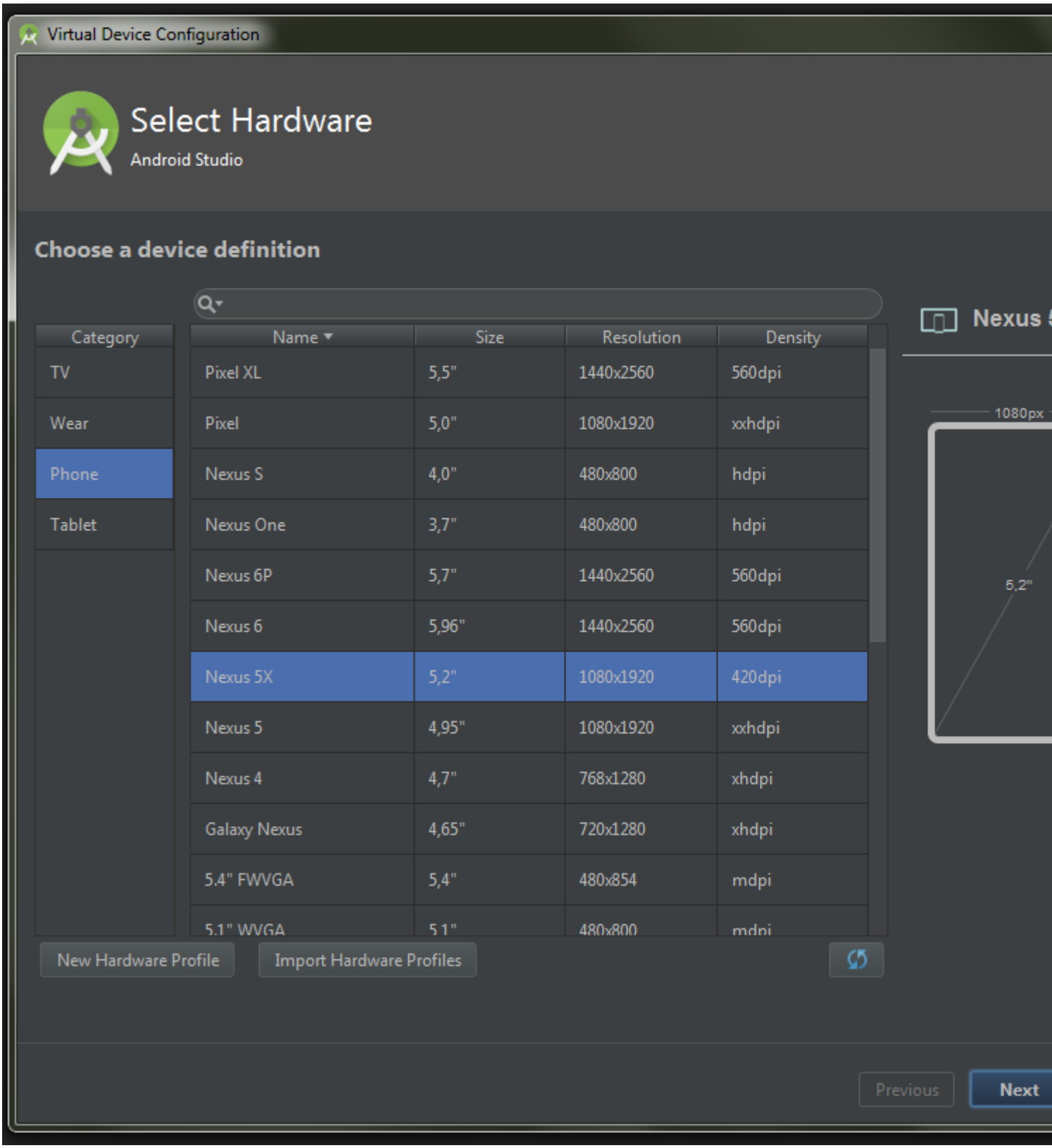
Чтобы настроить AVD, выполните следующие действия:

1. Нажмите эту кнопку, чтобы открыть диспетчер AVD:

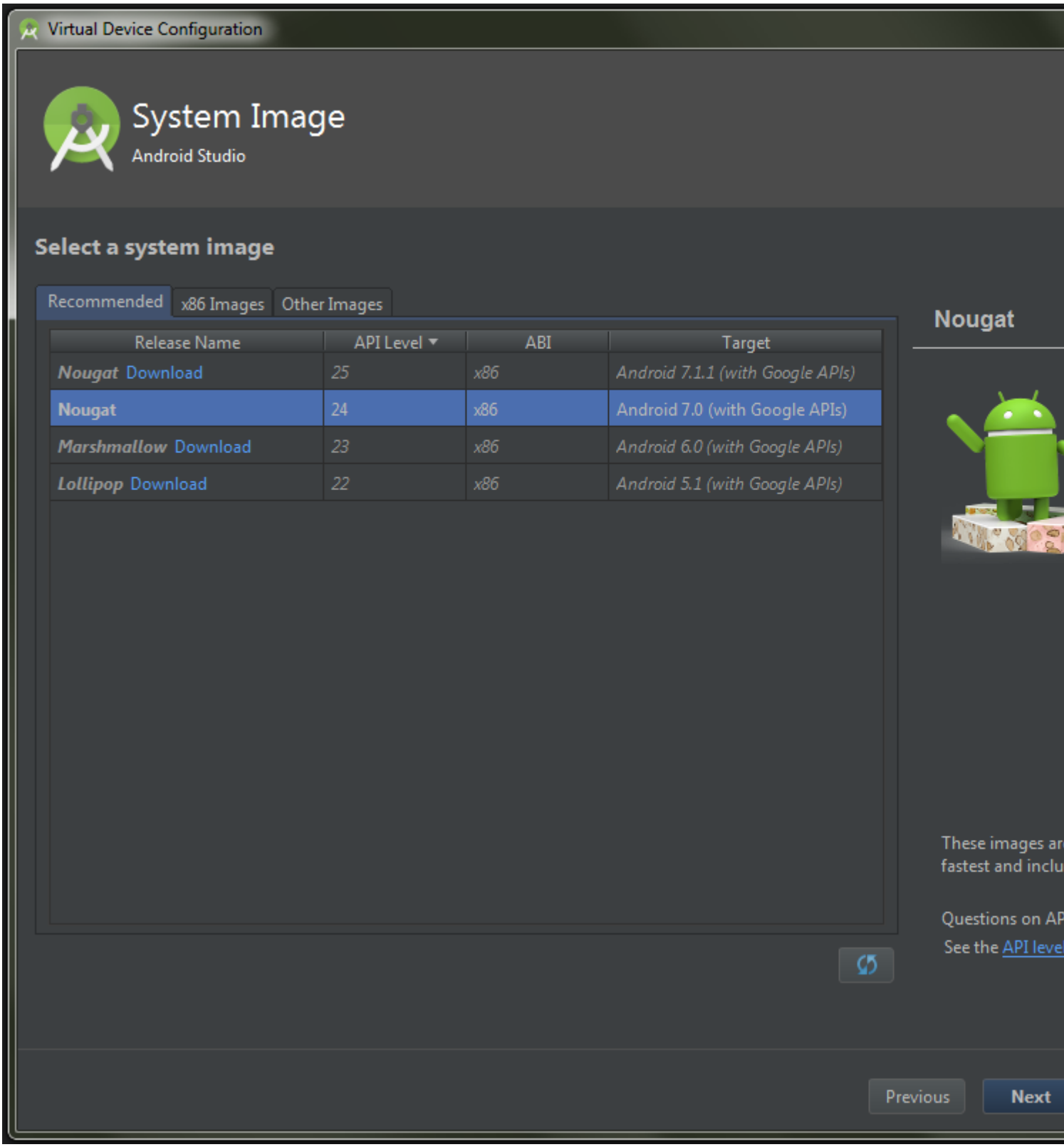
2. Вы должны увидеть такой диалог:



3. Теперь нажмите кнопку `+ Create Virtual Device...` Это вызовет диалог конфигурации виртуальных устройств:



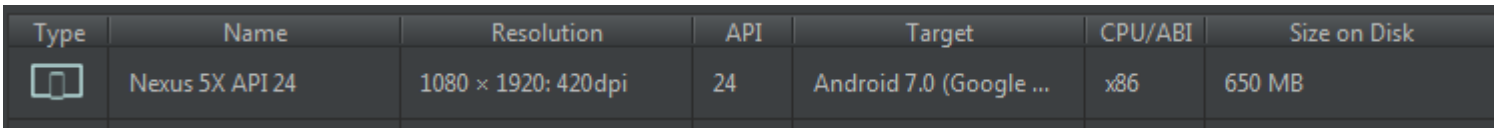
4. Выберите любое устройство, которое вы хотите, затем нажмите « `Next` :



5. Здесь вам нужно выбрать версию Android для своего эмулятора. Вам также может потребоваться загрузить его сначала, нажав `Download` . После того, как вы выбрали версию, нажмите « `Next` .

6. Здесь введите имя для своего эмулятора, начальную ориентацию и хотите ли вы отображать вокруг него рамку. После того, как вы выбрали все это, нажмите « `Finish` .

7. Теперь у вас есть новый AVD, готовый для запуска приложений на нем.

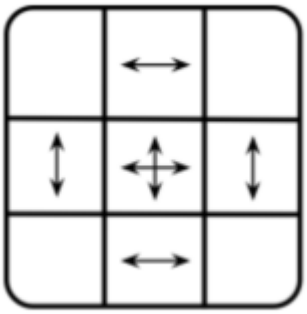


Прочитайте Начало работы с Android онлайн: https://riptutorial.com/ru/android/topic/85/начало-работы-с-android

глава 2: 9-патч-изображения

замечания

Файл с *9 патчами* - это специально отформатированный файл, так что Android знает, какие области / части изображения могут масштабироваться или не могут быть масштабированы. Это разбивает ваше изображение на сетку 3x3. Углы остаются немасштабированными, стороны масштабируются в одном направлении, а центр масштабируется в обоих измерениях.



Изображение Nine Patch (9-Patch) представляет собой растровое изображение, которое имеет ширину одного пикселя вокруг всего изображения. Игнорирование 4 пикселей в углах изображения. Эта граница обеспечивает метаданные для самого растрового изображения. Границы отмечены сплошной черной линией (линиями).

Изображение с `.9.png` хранится с расширением `.9.png` .

Верхняя граница указывает области, которые растягиваются горизонтально. Левая граница указывает области, которые растягиваются вертикально.

Нижняя граница указывает на заполнение по горизонтали. Правая рамка указывает на добавление по вертикали.

Границы заполнения обычно используются для определения того, где текст должен быть нарисован.

Существует отличный инструмент, предоставляемый Google, что *значительно* упрощает создание этих файлов.

Находится в Android SDK: `android-sdk\tools\lib\draw9patch.jar`

Examples

Основные закругленные углы

Ключ к правильному растягиванию находится в верхней и левой границах.

Верхняя граница контролирует горизонтальное растяжение, а левая граница контролирует вертикальное растяжение.

Этот пример создает закругленные углы, подходящие для тоста.

Детали изображения, которые находятся ниже *верхней границы* и справа от *левой границы,* будут расширяться, чтобы заполнить все неиспользуемое пространство.

Этот пример будет растянут на все комбинации размеров, как показано ниже:

Основной счетчик

`Spinner` можно перераспределить в соответствии с вашими требованиями к стилю с помощью Nine Patch.

В качестве примера см. Этот Девять исправлений:

Как вы можете видеть, он имеет 3 чрезвычайно маленьких участка растяжения.

Верхняя граница только слева от значка. Это означает, что я хочу, чтобы левая сторона (полная прозрачность) рисовала, чтобы заполнить вид `Spinner` до тех пор, пока значок не будет достигнут.

На левой границе отмечены прозрачные сегменты вверху и внизу значка. Это означает, что верхний и нижний будут расширяться до размера представления `Spinner` . Это приведет к тому, что сам значок будет центрирован по вертикали.

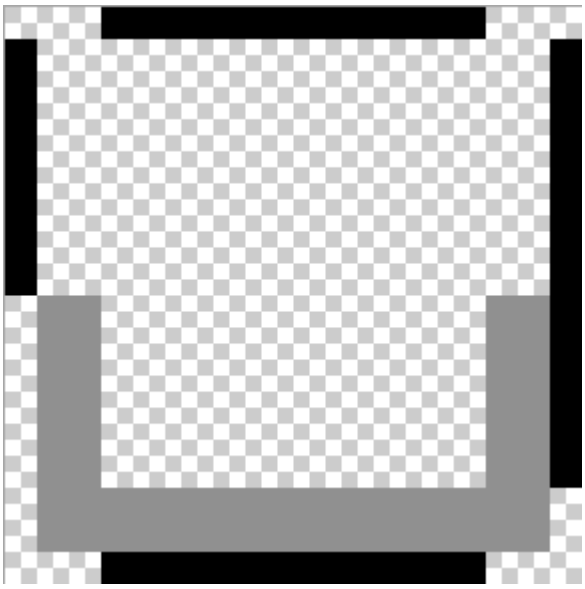
Использование изображения без метаданных Nine Patch:

Использование изображения с помощью метаданных Nine Patch:

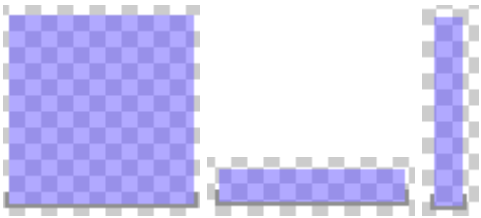
Дополнительные соединительные линии

Изображения с девятью патчами позволяют дополнительно определять линии заполнения в изображении. Проводными линиями являются линии справа и внизу.

Если представление задает изображение с 9 патчами в качестве фона, строки заполнения используются для определения пространства для содержимого View (например, ввода текста в `EditText`). Если строки заполнения не определены, вместо них используются левая и верхняя строки.



Область содержимого растянутого изображения выглядит следующим образом:



Прочитайте 9-патч-изображения онлайн: https://riptutorial.com/ru/android/topic/461/9-патч-изображения

глава 3: ACRA

Синтаксис

- андроид: имя = "ACRAHandler"
- ACRA.init (это, config);
- открытый класс ACRAHandler расширяет Приложение {

параметры

параметр	Описание
@ReportCrashes	Определяет параметры ACRA, например, где они должны быть представлены, пользовательский контент и т. Д.
formUri	путь к файлу, который сообщает об аварии

замечания

- ACRA больше не поддерживает формы Google, поэтому вам нужен бэкэнд: https://github.com/ACRA/acra/wiki/Backends

Examples

ACRAHandler

Пример Класс расширения приложения для обработки отчетов:

```
@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);


    }

}
```

Пример манифеста

```
<?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>
```

Монтаж

специалист

```
<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'
```

Прочитайте ACRA онлайн: https://riptutorial.com/ru/android/topic/1324/acra

глава 4: ADB (Android Debug Bridge)

Вступление

ADB (Android Debug Bridge) - это инструмент командной строки, который используется для связи с экземпляром эмулятора или подключенным устройством Android.

Обзор АБР

Большая часть этой темы была разделена на adb shell

замечания

Список примеров, перенесенных в оболочку adb :

- Предоставление и аннулирование разрешений API 23+
- Отправьте текст, нажав клавишу и коснитесь событий на устройстве Android через ADB
- Список пакетов
- Запись дисплея
- Возможности Open Developer
- Установить дату / время с помощью adb
- Изменение прав доступа к файлам с помощью команды chmod
- Создание трансляции «Загрузка завершена»
- Печать данных приложения
- Просмотр содержимого внешнего / вторичного хранилища
- http://stackoverflow.com/documentation/android/9408/adb-shell/29140/adb-shell
- убить процесс внутри устройства Android

Examples

Печать подробного списка подключенных устройств

Чтобы получить подробный список всех устройств, подключенных к `adb` , введите в свой терминал следующую команду:

```
adb devices -l
```

Пример вывода

```
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
```

- Первый столбец - это серийный номер устройства. Если он начинается с `emulator-` , это устройство является эмулятором.
- `usb:` путь устройства в подсистеме USB.
- `product:` код продукта устройства. Это очень специфично для конкретного производителя, и, как вы можете видеть в случае устройства Archos `A50PL` выше, оно может быть пустым.
- `model:` модель устройства. Как `product` , может быть пустым.
- `device:` код устройства. Это также очень специфично для производителя и может быть пустым.

Чтение информации об устройстве

Напишите в терминале следующую команду:

```
adb shell getprop
```

Это напечатает всю доступную информацию в виде пар ключ / значение.

Вы можете просто прочитать определенную информацию, добавив имя конкретного ключа в команду. Например:

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

Вот несколько интересных фрагментов информации, которые вы получаете:

- `ro.product.model` : Название модели устройства (например, Nexus 6P)
- `ro.build.version.sdk` : Уровень API устройства (например, 23)
- `ro.product.brand` : Брендинг устройства (например, Samsung)

Полный вывод примера

```
[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]
```

Подключить ADB к устройству через WiFi

Стандартная конфигурация ADB включает подключение USB к физическому устройству. Если вы предпочитаете, вы можете переключиться в режим TCP / IP и подключить ADB через WiFi.

Не внедренное устройство

1. **Получить в той же сети:**
    - Убедитесь, что ваше устройство и компьютер находятся в одной сети.
2. **Подключите устройство к главному компьютеру с помощью USB-кабеля.**
3. **Подключить `adb` к устройству через сеть:**

    Пока ваше устройство подключено к `adb` через USB, выполните следующую команду, чтобы прослушать TCP / IP-соединение на порту (по умолчанию 5555):

    - Введите `adb tcpip <port>` (переключитесь в режим TCP / IP).
    - Отсоедините кабель USB от целевого устройства.
    - Тип `adb connect <ip address>:<port>` (порт не является обязательным, по умолчанию 5555).

    Например:

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

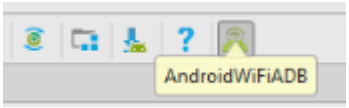
Если вы не знаете IP-адрес своего устройства, вы можете:

- проверьте IP-адрес в настройках WiFi вашего устройства.
- используйте ADB для обнаружения IP (через USB):
    1. Подключите устройство к компьютеру через USB
    2. В командной строке введите команду `adb shell ifconfig` и скопируйте IP-адрес своего устройства

Чтобы **вернуться** к отладке через **USB,** выполните следующую команду:

```
adb usb
```

Вы также можете подключить ADB через WiFi, установив плагин в Android Studio. Для этого перейдите в « *Настройки»> «Плагины»* и «Храните репозитории», найдите « *ADB WiFi»* , установите его и снова запустите Android Studio. Вы увидите новый значок на панели инструментов, как показано на следующем рисунке. Подключите устройство к *главному* компьютеру через USB и щелкните по этому значку *AndroidWiFiADB* . Появится сообщение о том, подключено ли ваше устройство или нет. Как только он подключится, вы можете отключить USB-порт.



Укорененное устройство

Примечание. Некоторые устройства, которые **внедрены,** могут использовать приложение ADB WiFi из Play Маркета, чтобы включить это простым способом. Кроме того, для некоторых устройств (особенно с CyanogenMod ROM) эта опция присутствует в настройках разработчика среди настроек. Включение этого параметра даст вам IP-адрес и номер порта, необходимые для подключения к `adb` , просто выполнив `adb connect <ip address>:<port>` .

Когда у вас есть внедренное устройство, но у вас нет доступа к USB-кабелю

Процесс подробно объясняется в следующем ответе: http://stackoverflow.com/questions/2604727/how-can-i-connect-to-android-with-adb-over-tcp/3623727#3623727 Наиболее важные команды показаны ниже.

Откройте терминал в устройстве и введите следующее:

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

```
start adbd
```

Например:

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

И на вашем компьютере:

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

Например:

```
adb connect 192.168.1.2:5555
```

Чтобы отключить его:

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

Избегайте таймаута

По умолчанию `adb` истечет через 5000 мс. Это может случиться в некоторых случаях, таких как медленный WiFi или большой APK.

Простое изменение конфигурации Gradle может сделать трюк:

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

Потяните (нажмите) файлы с (на) устройства

Вы можете вытащить (загрузить) файлы с устройства, выполнив следующую команду:

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

Например:

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

Вы также можете загружать (загружать) файлы с вашего компьютера на устройство:

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

Например:

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

Пример извлечения базы данных с устройства

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

Перезагрузите устройство

Вы можете перезагрузить устройство, выполнив следующую команду:

```
adb reboot
```

Выполните эту команду для перезагрузки в загрузчик:

```
adb reboot bootloader
```

Перезагрузка в режим восстановления:

```
adb reboot recovery
```

Имейте в виду, что устройство не выключится первым!

Включение / выключение Wi-Fi

Включи:

```
adb shell svc wifi enable
```

Выключи:

```
adb shell svc wifi disable
```

Просмотр доступных устройств

Команда:

```
 adb devices
```

Пример результата:

```
List of devices attached
emulator-5554  device
PhoneRT45Fr54  offline
```

```
123.454.67.45  no device
```

Первый столбец - порядковый номер устройства

Второй столбец - состояние соединения

Документация для Android

Подключение устройства по IP-адресу

Введите эти команды в Android-устройство Terminal

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

После этого вы можете использовать **CMD** и **ADB** для подключения, используя следующую команду

```
adb connect 192.168.0.101.5555
```

И вы можете отключить его и вернуть ADB для прослушивания на USB-устройстве

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

С компьютера, если у вас уже есть USB-доступ (не требуется root)

Еще проще переключиться на использование Wi-Fi, если у вас уже есть USB. Из командной строки на компьютере, на котором подключено устройство через USB, выполните команды

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

Замените 192.168.0.101 на IP-адрес устройства

Start / stop adb

Начать АБР:

```
adb kill-server
```

Остановить АБР:

```
adb start-server
```

Просмотр логарифма

Вы можете запустить `logcat` как команду adb или непосредственно в приглашении оболочки вашего эмулятора или подключенного устройства. Чтобы просмотреть выход журнала с помощью `adb` , перейдите в каталог SDK platform-tools / и выполните:

```
$ adb logcat
```

Кроме того, вы можете создать соединение с устройством и затем выполнить:

```
$ adb shell
$ logcat
```

Одна полезная команда:

```
adb logcat -v threadtime
```

Это отображает дату, время вызова, приоритет, тег и PID и TID потока, выдающего сообщение в формате длинного сообщения.

фильтрация

Журналы Logcat получили так называемые уровни журналов:

> **V** - Verbose, **D** - Debug, **I** - Info, **W** - Предупреждение, **E** - Ошибка, **F** - Фатальный, **S** - Беззвучный

Вы также можете фильтровать logcat по уровню журнала. Например, если вы хотите только выводить уровень отладки:

```
adb logcat *:D
```

Logcat можно отфильтровать по имени пакета, конечно, вы можете объединить его с фильтром уровня журнала:

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

Кроме того, можно фильтровать с помощью Grep журнала (больше на фильтрации вывода LogCat здесь):

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

В Windows фильтр можно использовать с помощью findstr, например:

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

Чтобы просмотреть альтернативный буфер журнала [main | events | radio], запустите `logcat` с параметром `-b` :

```
adb logcat -b radio
```

Сохранить вывод в файле:

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

Сохраните вывод в файле, а также просмотрите его:

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

Очистка журналов:

```
adb logcat -c
```

Прямая команда ADB для определенного устройства в настройке нескольких устройств

1. Настроить устройство по серийному номеру

Используйте параметр `-s` а затем имя устройства, чтобы выбрать, на каком устройстве должна запускаться команда `adb` . Параметры `-s` должны быть сначала в строке, **перед** командой.

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

Пример:

```
adb devices

List of devices attached
emulator-5554       device
02157df2d1faeb33    device

adb -s emulator-5554 shell
```

Пример # 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. Нацеливайте устройство, если подключен только один тип устройства

Вы можете настроить целевой эмулятор только на -e

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

Или вы можете настроить таргетинг только на подключенное USB-устройство на -d

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

Выполнение снимков экрана и видео (только для kitkat) с дисплея устройства

Снимок экрана: вариант 1 (чистый adb)

Команда adb `shell` позволяет нам выполнять команды с использованием встроенной оболочки устройства. Команда `screencap` shell захватывает содержимое, видимое в данный момент на устройстве, и сохраняет его в заданный файл изображения, например `/sdcard/screen.png` :

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

Затем вы можете использовать команду pull для загрузки файла с устройства в текущий каталог на вашем компьютере:

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

Снимок экрана: Вариант 2 (быстрее)

Выполните следующий однострочный:

(Зефир и ранее):

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

(Нугат и позже):

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

Флаг `-p` перенаправляет вывод команды `screencap` в stdout. Выражение Perl, которое транслируется, очищает некоторые проблемы конца строки в Marshmallow и ранее. Затем поток записывается в файл с именем `screen.png` в текущем каталоге. См. Эту статью и эту статью для получения дополнительной информации.

видео

это работает только в KitKat и только через ADB. Это не работает ниже Kitkat Чтобы начать запись экрана вашего устройства, выполните следующую команду:

`adb shell screenrecord /sdcard/example.mp4` Эта команда начнет запись экрана вашего устройства с использованием настроек по умолчанию и сохранит полученное видео в файл `/sdcard/example.mp4` на вашем устройстве.

Когда вы закончите запись, нажмите Ctrl + C (z в Linux) в окне командной строки, чтобы остановить запись на экране. Затем вы можете найти файл записи экрана в указанном вами местоположении. Обратите внимание, что запись на экране сохраняется во внутреннем хранилище вашего устройства, а не на вашем компьютере.

Настройки по умолчанию - использовать стандартное разрешение экрана вашего устройства, кодировать видео с битрейтом 4 Мбит / с и устанавливать максимальное время записи экрана на 180 секунд. Для получения дополнительной информации о параметрах командной строки, которые вы можете использовать, выполните следующую команду:

`adb shell screenrecord -help` , Это работает без укоренения устройства. Надеюсь это поможет.

Очистить данные приложения

Можно удалить пользовательские данные определенного приложения с помощью `adb` :

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

Это то же самое, что и для просмотра настроек на телефоне, выберите приложение и нажмите кнопку очистки данных.

- `pm` вызывает диспетчер пакетов на устройстве
- `clear` удаляет все данные, связанные с пакетом

Отправка трансляции

Можно отправлять трансляцию `BroadcastReceiver` с помощью `adb` .

В этом примере мы отправляем трансляцию с действием `com.test.app.ACTION` и string extra в

bundle `'foo'='bar'` :

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

Вы можете поместить любой поддерживаемый тип в пакет, а не только строки:

--ez - boolean
--ei - целое число
--el - long
-ef - float
--eu - uri
--eia - массив int (разделенный символом ',')
--ela - длинный массив (разделенный символом ',')
--efa - массив с плавающей точкой (разделенный символом ',')
--esa - строковый массив (разделенный символом ',')

Для отправки намерения использовать определенный параметр package / class `-n` или `-p` . Отправка в пакет:

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

Отправка на конкретный компонент (`SomeReceiver` класса в `com.test.app package`):

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

Полезные примеры:

- Отправка «полной загрузки»
- Отправка трансляции «по времени» после установки времени с помощью команды adb

Установка и запуск приложения

Чтобы установить файл APK , используйте следующую команду:

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

или если приложение существует, и мы хотим переустановить

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

Чтобы удалить приложение , мы должны указать его пакет

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

Используйте следующую команду для запуска приложения с предоставленным именем пакета (или определенной деятельностью в приложении):

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

Например, чтобы запустить Waze:

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

Резервное копирование

Вы можете использовать команду `adb backup` для резервного копирования вашего устройства.

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

`-f <filename>` указать имя файла по **умолчанию:** *создает backup.ab в текущем каталоге*

`-apk|noapk` включить / отключить резервное копирование по **умолчанию** *.apks* **:** *-noapk*

`-obb|noobb` включить / отключить резервное копирование дополнительных файлов по **умолчанию:** *-noobb*

`-shared|noshared` к содержимому общего хранилища / содержимого SD-хранилища *noshared по* **умолчанию:** *-noshared*

`-all` резервную копию всех установленных Apllications

`-system|nosystem` : система включает системные приложения по **умолчанию:** *-система*

`<packages>` список пакетов для резервного копирования (например , com.example.android.myapp) (не требуется , если `-all` указано)

Для полной резервной копии устройства, включая все, используйте

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

> **Примечание.** Выполнение полной резервной копии может занять много времени.

Чтобы восстановить резервную копию, используйте

```
adb restore backup.ab
```

Установка ADB в системе Linux

Как установить Android Debugging Bridge (ADB) в систему Linux с терминалом, используя

репозитории вашего дистрибутива.

Установите в систему Ubuntu / Debian через apt:

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

Установите в систему Fedora / CentOS через yum:

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

Установите в систему Gentoo с портом:

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

Установите в систему openSUSE с помощью zypper:

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

Установите в систему Arch с помощью pacman:

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

Список всех разрешений, требующих предоставления времени выполнения от пользователей на Android 6.0

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

Просмотр внутренних данных приложения (данные / данные /) на устройстве

Во-первых, убедитесь, что ваше приложение может быть скопировано в `AndroidManifest.xml` , т `android:allowBackup` не является `false` .

Команда резервного копирования:

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

Создайте tar с командой dd:

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

Извлеките смолу:

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

Затем вы можете просмотреть извлеченный контент.

Просмотреть стек активности

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

Очень полезно при использовании вместе с командой `watch` unix:

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

Просмотр и извлечение файлов кеша приложения

Вы можете использовать эту команду для перечисления файлов для собственного отлаживаемого apk:

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

И этот скрипт для вытаскивания из кеша, сначала скопируйте содержимое на SDCard, потяните, а затем удалите его в конце:

```
#!/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'"
```

Затем вы можете вытащить файл из кеша следующим образом:

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

Получить файл базы данных через ADB

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

Прочитайте ADB (Android Debug Bridge) онлайн:
https://riptutorial.com/ru/android/topic/1051/adb--android-debug-bridge-

глава 5: adb shell

Вступление

`adb shell` открывает `adb shell` Linux в целевом устройстве или эмуляторе. Это самый мощный и универсальный способ управления Android-устройством через `adb` .

Этот раздел был разделен на ADB (Android Debug Bridge) из-за достижения предела примеров, многие из которых `adb shell` команду `adb shell` .

Синтаксис

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

параметры

параметр	подробности
-e	выбрать escape-символ или «none»; default '~'
-n	не читайте из stdin
-T	отключить выделение PTY
-t	принудительное распределение PTY
-Икс	отключить коды удаленного выхода и разделение stdout / stderr

Examples

Отправьте текст, нажав клавишу и коснитесь событий на устройстве Android через ADB

выполните следующую команду, чтобы вставить текст в представление с фокусом (если он поддерживает ввод текста)

6,0

Отправьте текст на SDK 23+

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

Если вы уже подключены к вашему устройству через `adb` :

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

6,0

Отправить текст до SDK 23

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

Пробелы не принимаются в качестве входных данных, заменяя их% s.

Отправить события

Чтобы имитировать нажатие клавиши включения питания

```
adb shell input keyevent 26
```

или альтернативно

```
adb shell input keyevent POWER
```

Даже если у вас нет аппаратного ключа, вы все равно можете использовать `keyevent` для выполнения эквивалентного действия

```
adb shell input keyevent CAMERA
```

Отправить событие касания как вход

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

Отправить событие салфетки как вход

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

* DURATION является необязательным, по умолчанию = 300 мс. источник

Получите точки X и Y, включив расположение указателя в опции разработчика.

Сценарий оболочки образца ADB

> Чтобы запустить скрипт в Ubuntu, Create script.sh щелкните правой кнопкой мыши файл и добавьте права на чтение / запись и отметьте флажок, **чтобы выполнить файл как программу** .
>
> Откройте эмулятор терминала и запустите команду ./script.sh

Script.sh

```
for (( c=1; c<=5; c++ ))
   do
      adb shell input tap X Y
      echo "Clicked $c times"
      sleep 5s
   done
```

Полный список номеров событий

- краткое изложение нескольких интересных событий ADB Shell Input Events
- справочная документация https://developer.android.com/reference/android/view/KeyEvent.html#KEYCODE_POWER .

Список пакетов

Распечатывает все пакеты, необязательно только те, чье имя пакета содержит текст в <FILTER>.

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

All <FILTER>

adb shell pm list packages
```

Атрибуты:

`-f` чтобы увидеть их связанный файл.

`-i` См. установщик для пакетов.

`-u` также включить удаленные пакеты.

`-u` Также включают удаленные пакеты.

Атрибуты, которые фильтруют:

`-d` для отключенных пакетов.

`-e` для включенных пакетов.

`-s` для системных пакетов.

`-3` для сторонних пакетов.

`--user <USER_ID>` для конкретного пользовательского пространства для запроса.

Предоставление и аннулирование разрешений API 23+

Однострочный, который помогает предоставлять или отзывать уязвимые разрешения.

- **предоставление**

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

- **отменяющий**

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

- **Предоставление всех разрешений во время выполнения при установке (-g)**

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

Печать данных приложения

Эта команда печатает все соответствующие данные приложения:

- код версии
- название версии
- предоставленные разрешения (Android API 23+)
- так далее..

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

Запись дисплея

4,4

Запись дисплея устройств под управлением Android 4.4 (API уровня 19) и выше:

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

(нажмите Ctrl-C, чтобы остановить запись)

Загрузите файл с устройства:

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

> Примечание. Остановите запись по экрану, нажав Ctrl-C, в противном случае запись автоматически прекратится через три минуты или ограничение по времени `--time-limit` .

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

Устанавливает размер видео: 1280x720. Значением по умолчанию является собственное разрешение дисплея устройства (если поддерживается), 1280x720, если нет. Для достижения наилучших результатов используйте размер, поддерживаемый кодером Advanced Video Coding (AVC) вашего устройства.

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

Устанавливает скорость передачи видео для видео в мегабит в секунду. Значение по умолчанию - 4 Мбит / с. Вы можете увеличить скорость передачи данных, чтобы улучшить качество видео, но при этом получается увеличение файлов фильмов. В следующем примере скорость записи составляет 5 Мбит / с:

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

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

Устанавливает максимальное время записи в секундах. Значение по умолчанию и максимальное значение равно 180 (3 минуты).

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

Поворачивает выход на 90 градусов. Эта особенность экспериментальна.

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

Отображает информацию журнала на экране командной строки. Если вы не установите этот параметр, утилита не отображает информацию во время работы.

> Примечание. Это может не работать на некоторых устройствах.

4,4

Команда записи экрана несовместима с версиями Android версии 4.4

> Команда screenrecord - это утилита оболочки для записи дисплеев устройств под управлением Android 4.4 (API уровня 19) и выше. Утилита записывает активность экрана в файл MPEG-4.

Изменение прав доступа к файлам с помощью команды chmod

Обратите внимание, что для изменения доступа к файлам ваше устройство должно быть укоренено, *`su` binary не поставляется с заводскими устройствами!*

Конвенция:

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

Числовое разрешение, построенное из пользовательских, групповых и глобальных разделов.

Например, если вы хотите изменить файл для чтения, записи и выполнения всеми, это будет ваша команда:

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

Или же

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

если вы намерены отказаться от каких-либо разрешений на него.

1- я цифра - указывает разрешение пользователя, **2-я цифра** - указывает разрешение группы, **3- я цифра** - указывает разрешение на мир (другие).

Разрешения доступа:

```
--- :   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)
```

Установить дату / время с помощью adb

6,0

Формат по умолчанию: `MMDDhhmm[[CC]YY][.ss]` , это (2 цифры)

Например, чтобы установить 17 июля 10:10 утра, не меняя текущий год, введите:

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

Совет 1: изменение даты не будет немедленно отображено, и заметные изменения произойдут только после того, как системные часы перейдут на следующую минуту.
Вы можете принудительно обновить, `TIME_SET` к вашему вызову трансляцию намерений `TIME_SET` , например:

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

Совет 2: синхронизация часов Android с локальной машиной:

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"
```

Оба совета вместе:

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

6,0

Формат по умолчанию: «YYYYMMDD.HHmmss»

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

Подсказка: синхронизировать часы Android с локальной (на основе Linux):

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

Возможности Open Developer

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

Будет перемещаться по вашему устройству / эмулятору в разделе « `Developer Options` ».

Создание трансляции «Загрузка завершена»

Это актуально для приложений, реализующих `BootListener` . Проверьте свое приложение, убив ваше приложение, а затем проверьте:

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

(замените `your.package/your.app.BootListener` с правильными значениями).

Просмотр содержимого внешнего / вторичного хранилища

Просмотр содержимого:

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

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

Просмотр пути:

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

убить процесс внутри устройства Android

Иногда логарифм Android работает бесконечно с ошибками, возникающими из-за какого-то процесса, который не принадлежит вам, разряжая батарею или просто затрудняя отладку вашего кода.

Удобный способ устранения проблемы без перезапуска устройства - найти и убить процесс, вызывающий проблему.

От Logcat

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

обратите внимание на номер процесса: 1550

Теперь мы можем открыть оболочку и убить процесс. Обратите внимание, что мы не можем убить процесс `root` .

```
adb shell
```

внутри оболочки мы можем больше узнать о процессе, используя

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

и убить его, если мы хотим:

```
kill -9 1550
```

Прочитайте adb shell онлайн: https://riptutorial.com/ru/android/topic/9408/adb-shell

глава 6: AdMob

Синтаксис

- compile 'com.google.firebase: firebase-ads: 10.2.1' // ПРИМЕЧАНИЕ: УСТАНОВИТЕ НА НОВУЮ ВЕРСИЮ, ЕСЛИ ИМЕЕТСЯ ДОСТУПНО
- `<uses-permission android:name="android.permission.INTERNET" />` Требуется для извлечения объявления
- AdRequest adRequest = новый AdRequest.Builder (). Build (); // Баннерное объявление
- AdView mAdView = (AdView) findViewById (R.id.adView); // Баннерное объявление
- mAdView.loadAd (adRequest); // Объявление баннера

параметры

Param	подробности
объявления: AdUnitId = "@ строка / main_screen_ad"	Идентификатор вашего объявления. Получите свой ID с сайта admob. *«Хотя это не является обязательным требованием, сохранение значений идентификатора вашего рекламного блока в файле ресурсов является хорошей практикой. Поскольку ваше приложение растет, и публикация объявлений требует созревания, может потребоваться изменить значения идентификатора. Если вы храните их в ресурсе файла, вам никогда не придется искать код, который ищет их ».* . [1]

замечания

- Требуется действительная учетная запись Admob
- Прочитайте политику admob . Удостоверьтесь, что вы не делаете ничего, что может заблокировать ваш аккаунт admob

Examples

Внедрение

Примечание. В этом примере требуется действительная учетная запись Admob и действительный рекламный код Admob.

Build.gradle на уровне приложения

Перейдите к последней версии, если она существует:

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

манифест

Для доступа к данным объявлений требуется разрешение на доступ к Интернету. Обратите внимание, что это разрешение не нужно запрашивать (с использованием API 23+), поскольку это нормальное разрешение и не опасно:

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

XML

В следующем примере XML показано объявление баннера:

```
<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" />
```

Код другого типа см. В справке Google AdMob .

Джава

Следующий код предназначен для интеграции баннерной рекламы. Обратите внимание, что для других типов объявлений может потребоваться другая интеграция:

```
// 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);
```

Добавьте `AdView` жизненного цикла `AdView` методы `onResume()` , `onPause()` и `onDestroy()` вашей деятельности:

```
@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();
}
```

Прочитайте AdMob онлайн: https://riptutorial.com/ru/android/topic/5334/admob

глава 7: AIDL

Вступление

AIDL - это язык определения интерфейса Android.

Какие? Зачем? Как ?

Какие? Это ограниченные услуги. Эта служба AIDL будет активна до тех пор, пока не будет существовать один из клиентов. Он работает на основе маршалинга и концепции unmarshaling.

Зачем? Удаленные приложения могут получить доступ к вашему сервису + Multi Threading (удаленный запрос приложения).

Как? Создайте файл .aidl. Внедрите интерфейс. Откройте интерфейс для клиентов.

Examples

Служба 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;
        }
    };

}
```

Подключение к службе

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

Прочитайте AIDL онлайн: https://riptutorial.com/ru/android/topic/9504/aidl

глава 8: AlarmManager

Examples

Запустите намерение позднее

1. Создайте приемник. Этот класс получит намерение и обработает его, как вы пожелаете.

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

2. Дайте намерение AlarmManager. Этот пример вызовет намерение отправляться в AlarmReceiver через 1 минуту.

```
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 );
```

Как отменить будильник

Если вы хотите отменить будильник, и у вас нет ссылки на исходный PendingIntent, используемый для установки будильника, вам нужно воссоздать PendingIntent точно так, как это было, когда оно было изначально создано.

Настройщик считается равным с помощью AlarmManager :

> если их действие, данные, тип, класс и категории одинаковы. Это не сравнивает никаких дополнительных данных, включенных в намерения.

Обычно код запроса для каждого аварийного сигнала определяется как константа:

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

Итак, для простой тревоги, настроенной так:

```
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);
```

Вот как вы создадите новую ссылку PendingIntent, которую вы можете использовать для отмены тревоги с помощью новой ссылки 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);
}
```

Создание точных сигналов тревоги на всех версиях Android

С течением времени в систему Android все больше и больше оптимизируются батареи, методы `AlarmManager` также значительно изменились (чтобы обеспечить более мягкое время). Однако для некоторых приложений по-прежнему необходимо быть как можно более точным на всех версиях Android. Следующий помощник использует самый точный метод, доступный на всех платформах для планирования `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);
    }
}
```

Режим API23 + Doze вмешивается в AlarmManager

Android 6 (API23) представил режим Doze, который мешает AlarmManager. Он использует определенные окна обслуживания для обработки аварийных сигналов, поэтому, даже если вы использовали `setExactAndAllowWhileIdle()` вы не можете убедиться, что ваш будильник срабатывает в нужный момент времени.

Вы можете отключить это поведение для своего приложения, используя настройки вашего телефона (`Settings/General/Battery & power saving/Battery usage/Ignore optimizations` или подобное)

Внутри приложения вы можете проверить этот параметр ...

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

... и в итоге отобразите соответствующий диалог настроек:

```
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);
```

Прочитайте AlarmManager онлайн: https://riptutorial.com/ru/android/topic/1361/alarmmanager

глава 9: Android Authenticator

Examples

Основная служба проверки подлинности учетной записи

Система аутентификации учетной записи Android может использоваться для аутентификации клиента с удаленного сервера. Требуется три части информации:

- Служба, инициированная `android.accounts.AccountAuthenticator` . Его метод `onBind` должен возвращать подкласс `AbstractAccountAuthenticator` .
- Активность для запроса пользователю учетных данных (активность входа)
- Файл ресурсов xml для описания учетной записи

1. Услуга:

В AndroidManifest.xml укажите следующие разрешения:

```
<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" />
```

Объявите службу в файле манифеста:

```
<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>
```

Обратите внимание, что `android.accounts.AccountAuthenticator` включен в тег `intent-filter` объекта. Ресурс xml (именованный `authenticator` здесь) указан в теге `meta-data` .

Класс обслуживания:

```
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. Ресурс 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" />
```

Не назначьте строку для `android:label` или назначьте недостающие чертежи. Он сработает без предупреждения.

3. Расширьте класс 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;
    }
}
```

Метод `addAccount()` в классе `AbstractAccountAuthenticator` важен, поскольку этот метод вызывается при добавлении учетной записи с экрана «Добавить учетную запись» в настройках. `AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE` важен, так как он будет включать объект AccountAuthenticatorResponse, который необходим для возврата ключей учетной записи при успешной проверке пользователя.

Прочитайте Android Authenticator онлайн: https://riptutorial.com/ru/android/topic/6759/android-authenticator

глава 10: Android NDK

Examples

Создание собственных исполняемых файлов для Android

Проект / JNI / main.c

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

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

Проект / JNI / Android.mk

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

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

Проект / JNI / Application.mk

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

Если вы хотите поддерживать устройства под управлением версий Android версии ниже 5.0 (API 21), вам необходимо скомпилировать ваш двоичный файл с помощью `APP_PLATFORM` установленного на более старый API, например, `android-8` . Это является следствием того, что Android 5.0 обеспечивает *независимые бинарные позиции* (PIE), тогда как более старые устройства не обязательно поддерживают PIE. Поэтому в зависимости от версии устройства вам необходимо использовать PIE или не-PIE. Если вы хотите использовать двоичный код в своем приложении для Android, вам нужно проверить уровень API и извлечь правильный двоичный файл.

`APP_ABI` можно изменить на определенные платформы, такие как `armeabi` для создания двоичного `armeabi` только для этих архитектур.

В худшем случае вы будете иметь как PIE, так и не-PIE двоичные файлы для каждой архитектуры (около 14 различных двоичных файлов, использующих ndk-r10e).

Чтобы создать исполняемый файл:

```
cd project
ndk-build
```

Вы найдете двоичные файлы в `project/libs/<architecture>/hello_world` . Вы можете использовать их через `ADB` (`push` и `chmod` с исполняемым разрешением) или из вашего приложения (извлечь и `chmod` с разрешением на выполнение).

Чтобы определить архитектуру ЦП, `ro.product.cpu.abi` свойство build `ro.product.cpu.abi` для первичной архитектуры или `ro.product.cpu.abilist` (на новых устройствах) для получения полного списка поддерживаемых архитектур. Вы можете сделать это, используя класс `android.os.Build` из вашего приложения или используя `getprop <name>` через ADB.

Как очистить сборку

Если вам нужно очистить сборку:

```
ndk-build clean
```

Как использовать makefile, отличный от Android.mk

ndk-build NDK_PROJECT_PATH = PROJECT_PATH APP_BUILD_SCRIPT = MyAndroid.mk

Как войти в систему ndk

Сначала убедитесь, что вы связываетесь с библиотекой протоколирования в файле `Android.mk` :

```
LOCAL_LDLIBS := -llog
```

Затем используйте один из следующих `__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)
```

Или используйте их более удобным способом, определяя соответствующие макросы:

```
#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__)
```

Пример :

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

Прочитайте Android NDK онлайн: https://riptutorial.com/ru/android/topic/492/android-ndk

глава 11: Android Studio

Examples

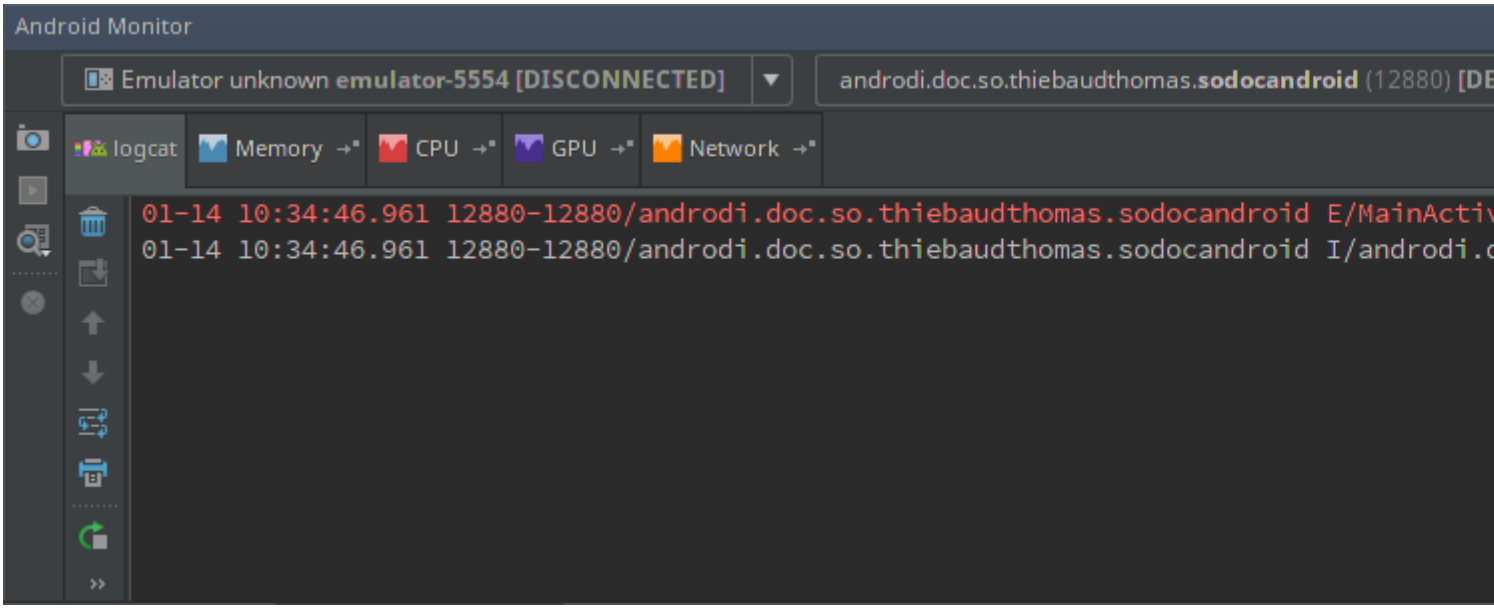
Фильтровать журналы из пользовательского интерфейса

Журналы Android можно фильтровать непосредственно из пользовательского интерфейса. Использование этого кода

```
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");
    }
}
```

Если я использую регулярное выражение `TAG1|TAG2` и уровень `verbose` я получаю

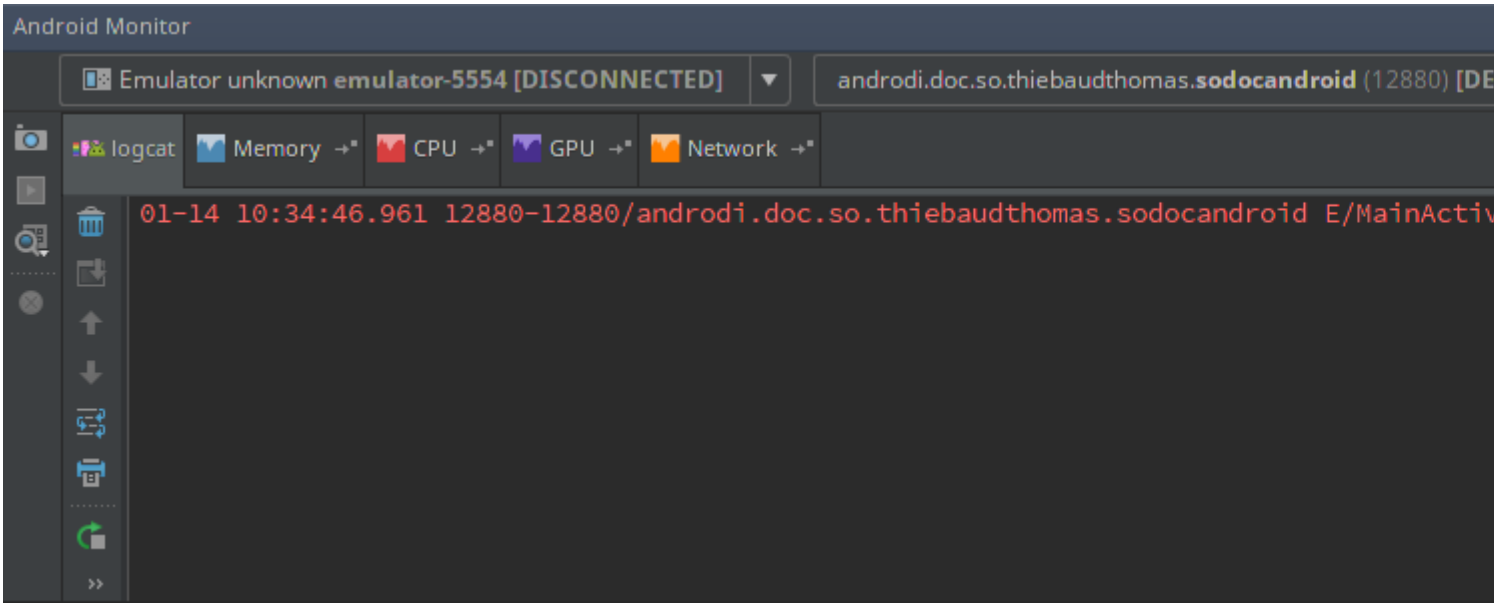
```
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log
from onCreate method with TAG1
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid
I/androdi.doc.so.thiebaudthomas.sodocandroid.MainActivity: Log from onCreate method with TAG2
```



Уровень может быть установлен для получения журналов с заданным уровнем и выше. Например, `verbose` уровень будет захватывать `verbose, debug, info, warn, error and assert` журналы.

Используя тот же пример, если я установил уровень на `error` , я получаю только

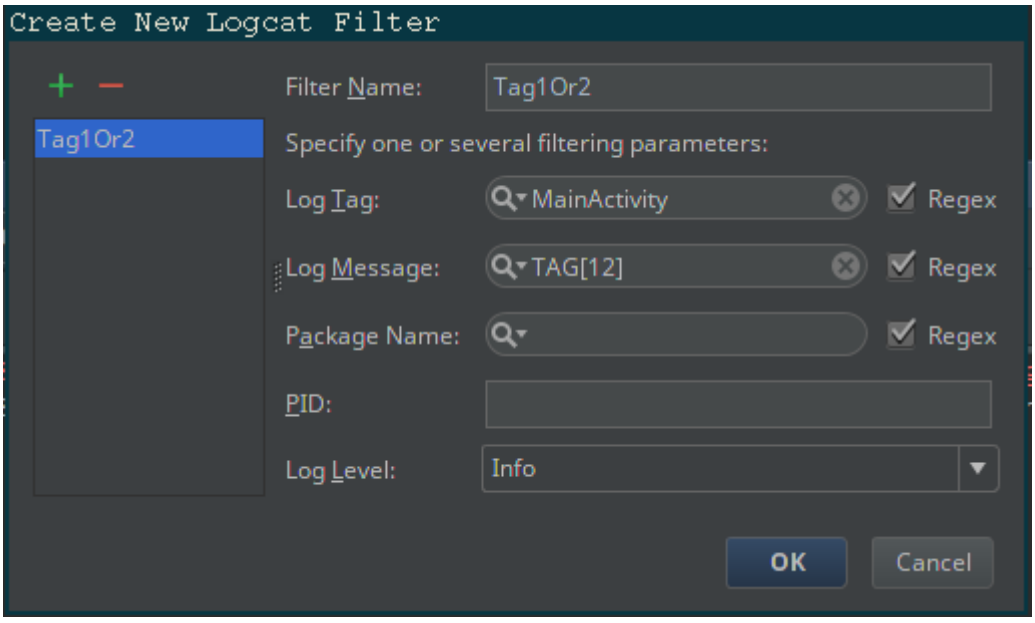
```
01-14 10:34:46.961 12880-12880/androdi.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log
from onCreate method with TAG1
```



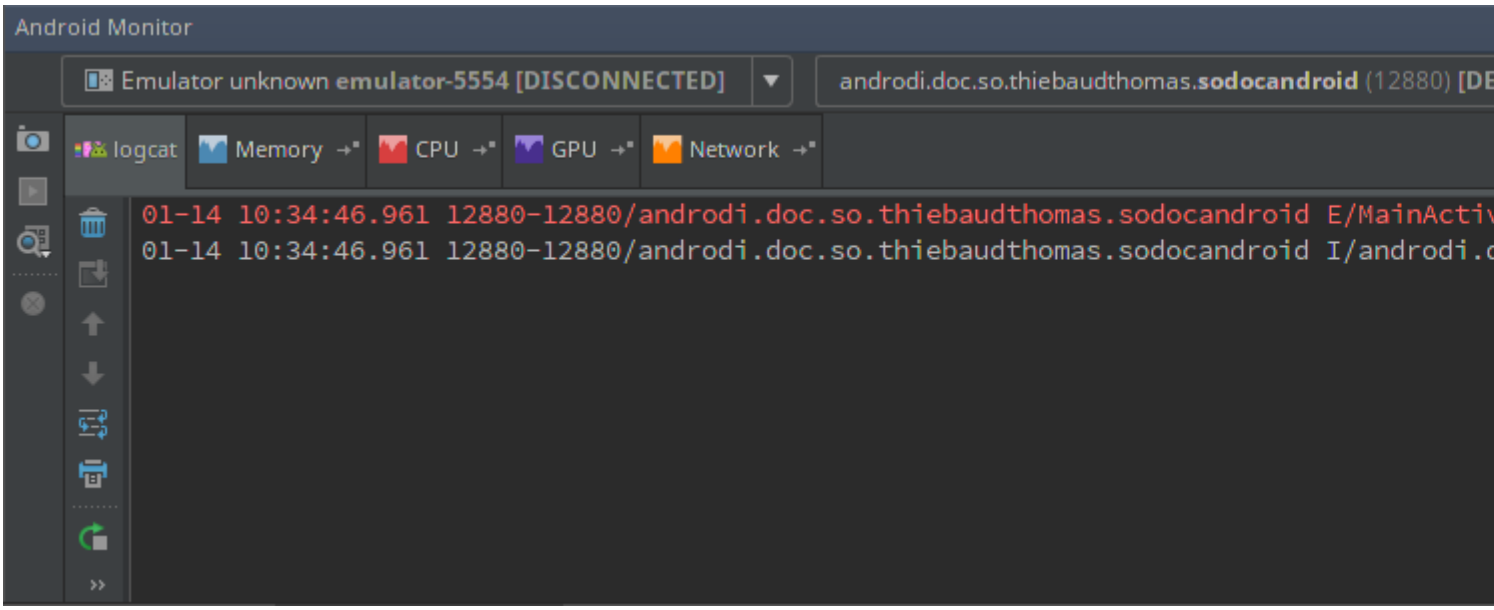
Создание конфигурации фильтров

Пользовательские фильтры можно установить и сохранить из пользовательского интерфейса. На вкладке `AndroidMonitor` нажмите правую раскрывающуюся панель (необходимо `Show only selected application` или « `No filters`) и выберите « `Edit filter configuration` .

Введите необходимый фильтр

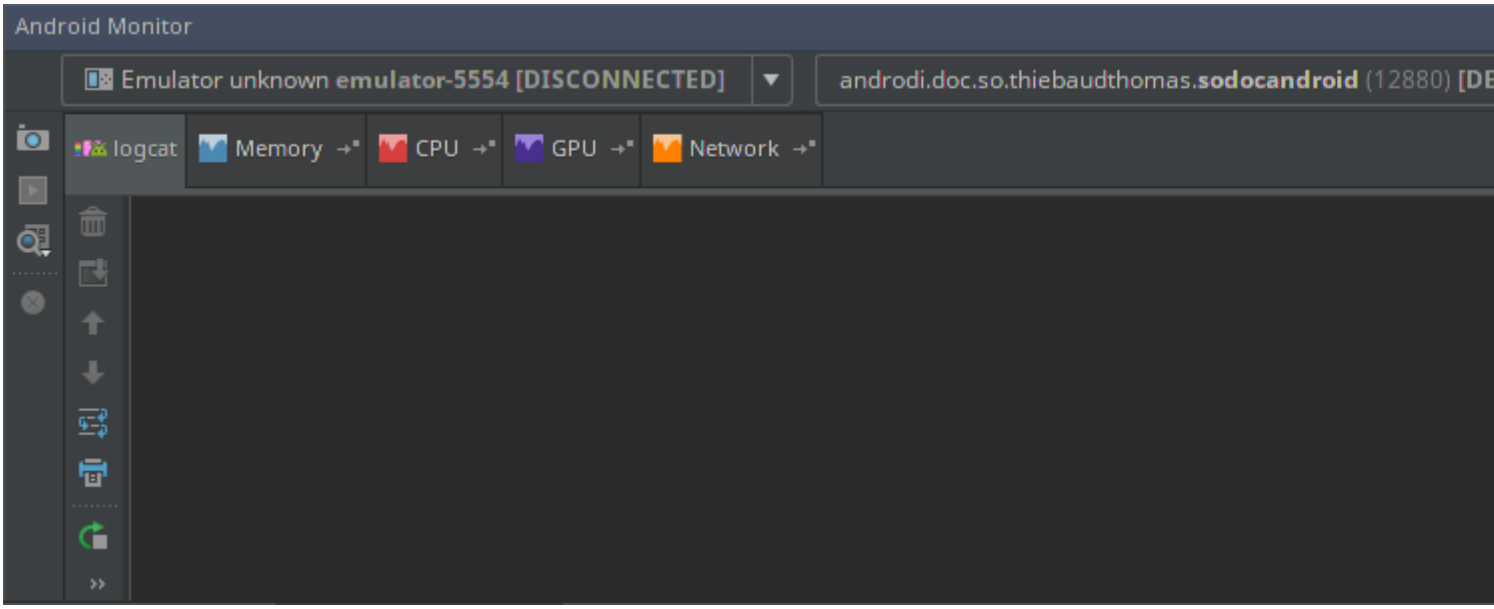


И используйте его (вы можете выбрать его из того же раскрывающегося списка)

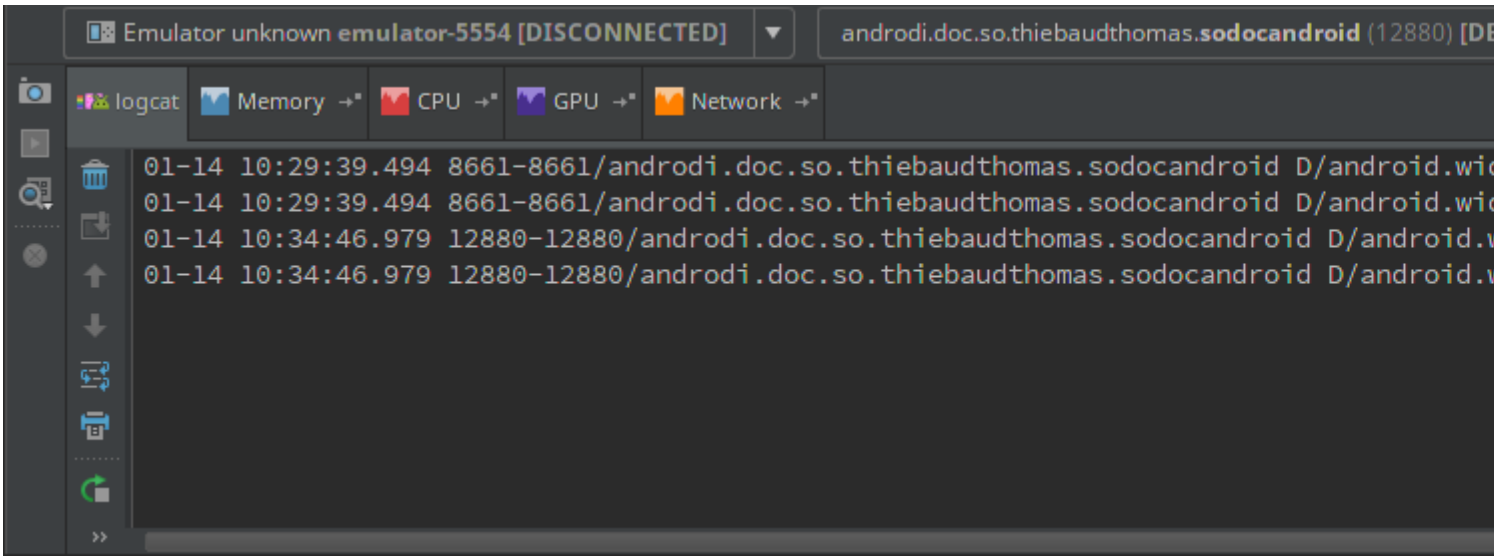


Важно. Если вы добавите ввод в панель фильтров, студия Android будет рассматривать как ваш фильтр, так и ваш вход.

Поскольку вход и фильтр не имеют выхода



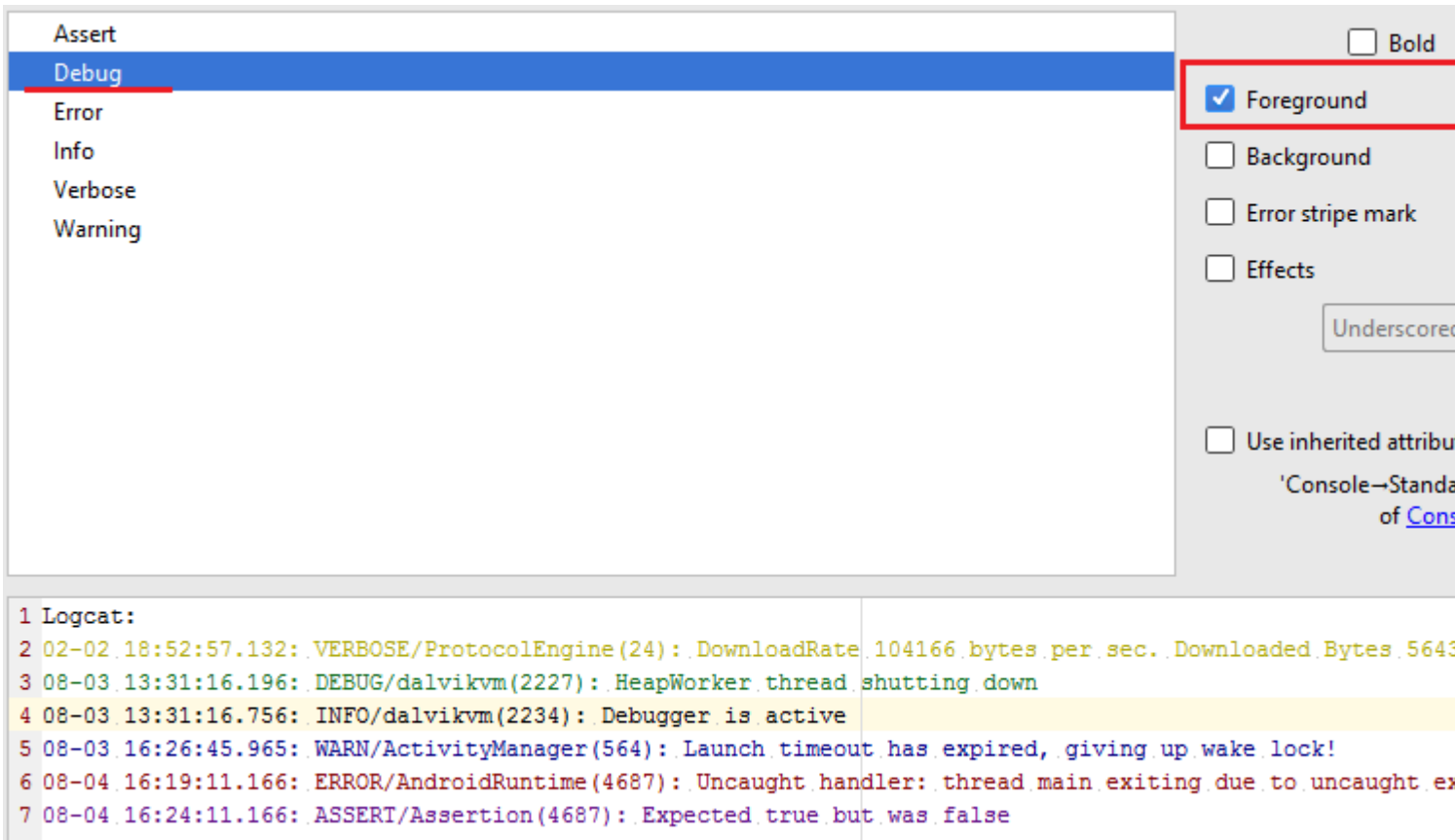
Без фильтра есть некоторые выходы



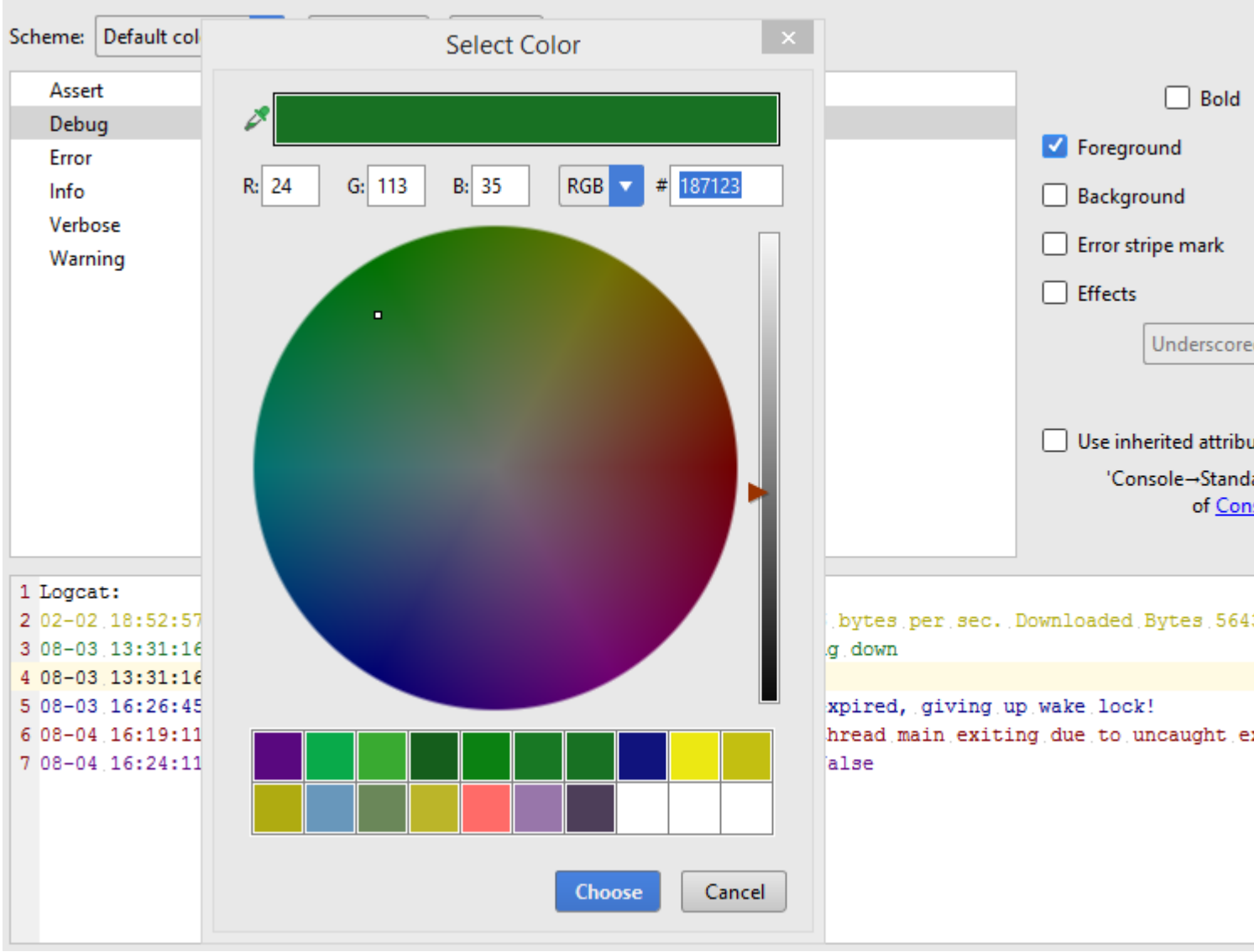
Пользовательские цвета сообщения logcat, основанные на важности сообщения

Перейдите в меню Файл -> Настройки -> Редактор -> Цвета и шрифты -> Android Logcat

Измените цвета по мере необходимости:

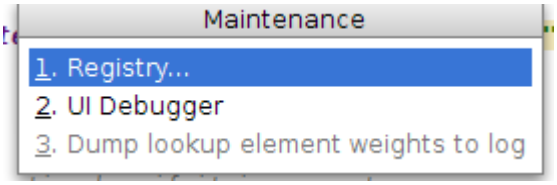


Выберите подходящий цвет:

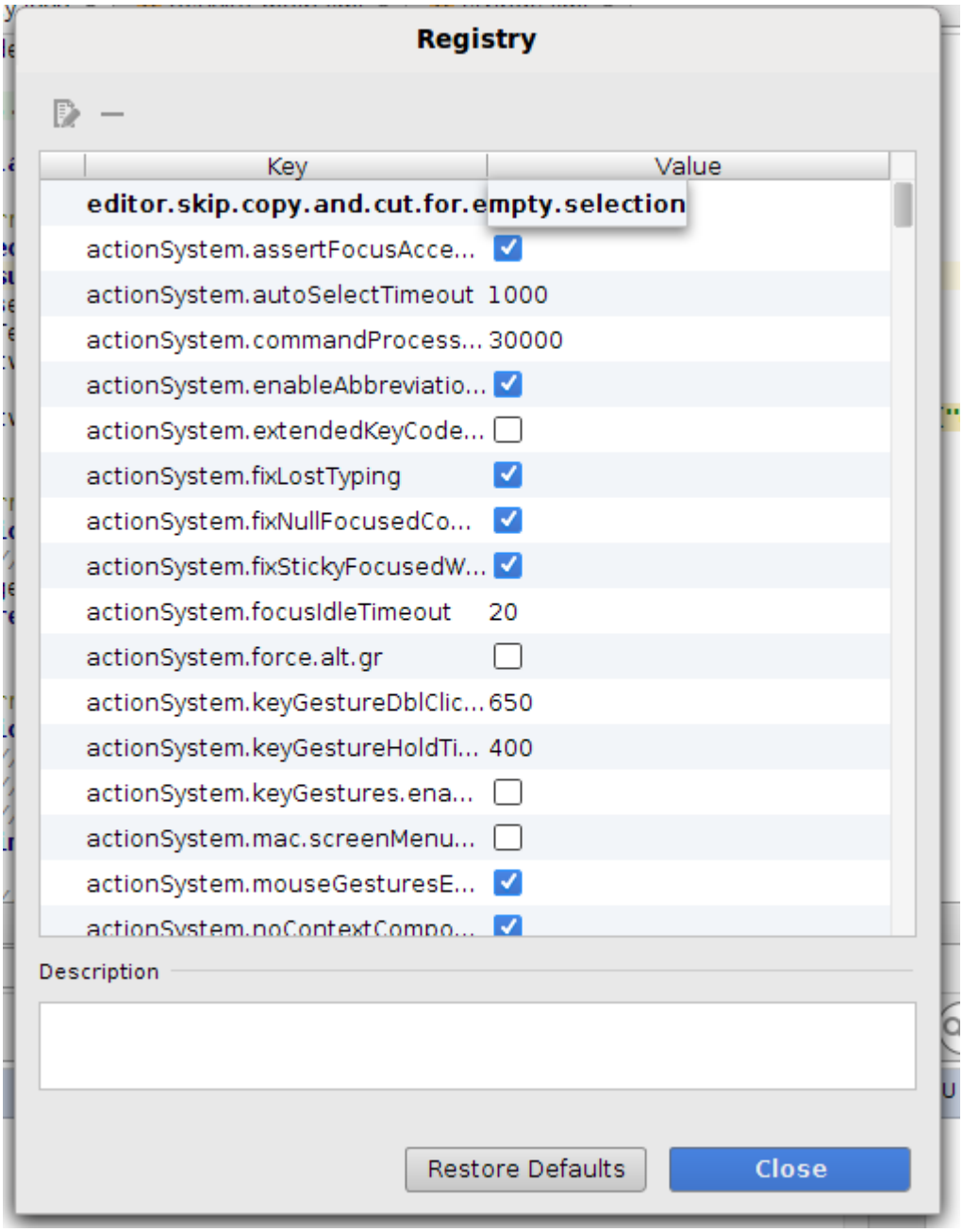


Включить / отключить пустую строку

`ctrl + alt + shift + /` (`cmd + alt + shift + /` on `MacOS`) должен показать вам следующее диалоговое окно:



Нажав на `Registry` вы получите



Ключ, который вы хотите включить / отключить,

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

Протестировано на `Linux Ubuntu` и `MacOS` .

Полезные ярлыки для Android Studio

Ниже приведены некоторые из наиболее распространенных / полезных ярлыков.

Они основаны на стандартной карте быстрого доступа IntelliJ. Вы можете переключиться на другие общие ярлыки IDE через `File -> Settings -> Keymap -> <Choose Eclipse/Visual Studio/etc from Keymaps dropdown>`

действие	кратчайший путь
Формат кода	`CTRL` + `ALT` + `L`
Добавить нереализованные методы	`CTRL` + `I`
Показать logcat	`ALT` + `6`
строить	`CTRL` + `F9`
Построить и запустить	`CTRL` + `F10`
найти	`CTRL` + `F`
Найти в проекте	`CTRL` + `SHIFT` + `F`
Найти и заменить	`CTRL` + `R`
Найти и заменить в проекте	`CTRL` + `SHIFT` + `R`
Методы переопределения	`CTRL` + `O`
Показать проект	`ALT` + `1`
Скрыть проект - logcat	`SHIFT` + `ESC`
Свернуть все	`CTRL` + `SHIFT` + `NumPad +`
Просмотр точек отладки	`CTRL` + `SHIFT` + `F8`
Расширить все	`CTRL` + `SHIFT` + `NumPad -`
Открыть настройки	`ALT` + `s`
Выберите Target (открыть текущий файл в представлении Project)	`ALT` + `F1` → `ENTER`
Поиск по всему миру	`SHIFT` → `SHIFT` (двойная смена)
Код | Surround With	`CTRL` → `ALT` + `T`
Создать метод формы выбранного кода	`ALT` + `CTRL`

Рефакторинг:

действие	кратчайший путь
Refactor Это (меню / выборщик для всех применимых	Mac `CTRL` + `T` - Win / Linux

действие	кратчайший путь
действий рефакторирования текущего элемента)	`CTRL` + `ALT` + `T`
переименовывать	`SHIFT` + `F6`
Метод извлечения	Mac `CMD` + `ALT` + `M` - Win / Linux `CTRL` + `ALT` + `M`
Извлечь параметр	Mac `CMD` + `ALT` + `P` - Win / Linux `CTRL` + `ALT` + `P`
Извлечь переменную	Mac `CMD` + `ALT` + `V` - Win / Linux `CTRL` + `ALT` + `V`

Android Studio улучшает производительность

Включить автономную работу:

1. Нажмите «Файл» -> «Настройки». Найдите «gradle» и щелкните в `Offline work` .
2. Перейдите в `--offline` Компилятор» (в том же диалоговом окне настроек ниже `Gradle`) и добавьте в текстовое поле « `Command-line Options --offline` .

Улучшить производительность колышек

Добавьте в свой файл gradle.properties следующие две строки кода.

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

Увеличение значения `-Xmx` и `-Xms` в `studio.vmoptions` файле

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

Окно

> % USERPROFILE%. {FOLDER_NAME} \ studio.exe.vmoptions и / или% USERPROFILE%. {FOLDER_NAME} \ studio64.exe.vmoptions

макинтош

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

Linux

~ /. {FOLDER_NAME} /studio.vmoptions и / или ~ /. {FOLDER_NAME} /studio64.vmoptions

Настройка Android Studio

Системные Требования

- Microsoft® Windows® 8/7 / Vista / 2003 (32 или 64 бит).
- Mac® OS X® 10.8.5 или выше, до 10.9 (Mavericks)
- Рабочий стол GNOME или KDE

Монтаж

Окно

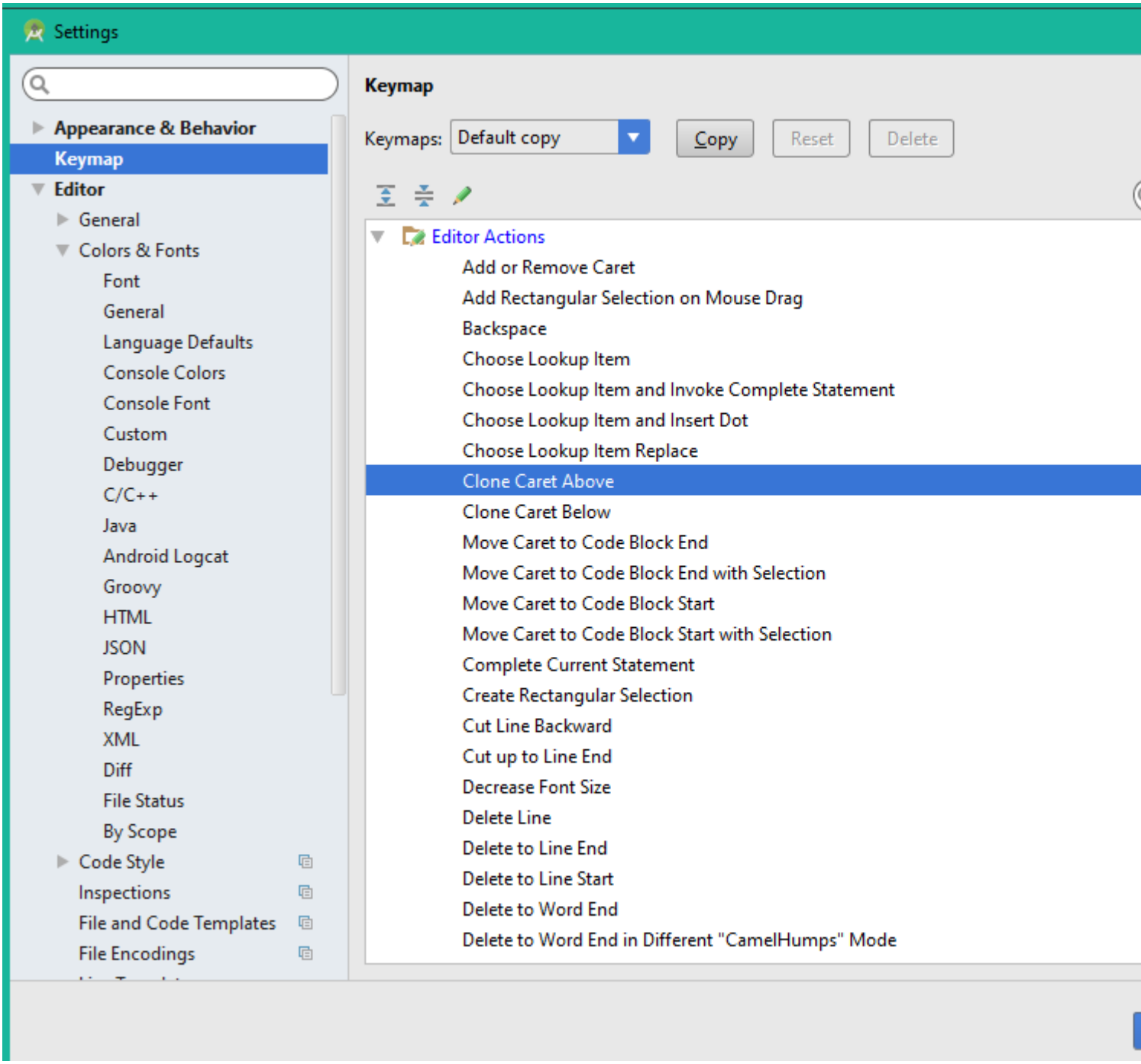
1. Загрузите и установите JDK (Java Development Kit) версии 8
2. Загрузить Android Studio
3. Запустите `Android Studio.exe` затем укажите путь JDK и загрузите последнюю версию SDK

Linux

1. Загрузите и установите JDK (Java Development Kit) версии 8
2. Загрузить Android Studio
3. Извлечь zip-файл
4. Откройте терминал, cd в извлеченную папку, cd в bin (пример `cd android-studio/bin`)
5. Запустить ./studio.sh

Просмотр и добавление ярлыков в Android Studio

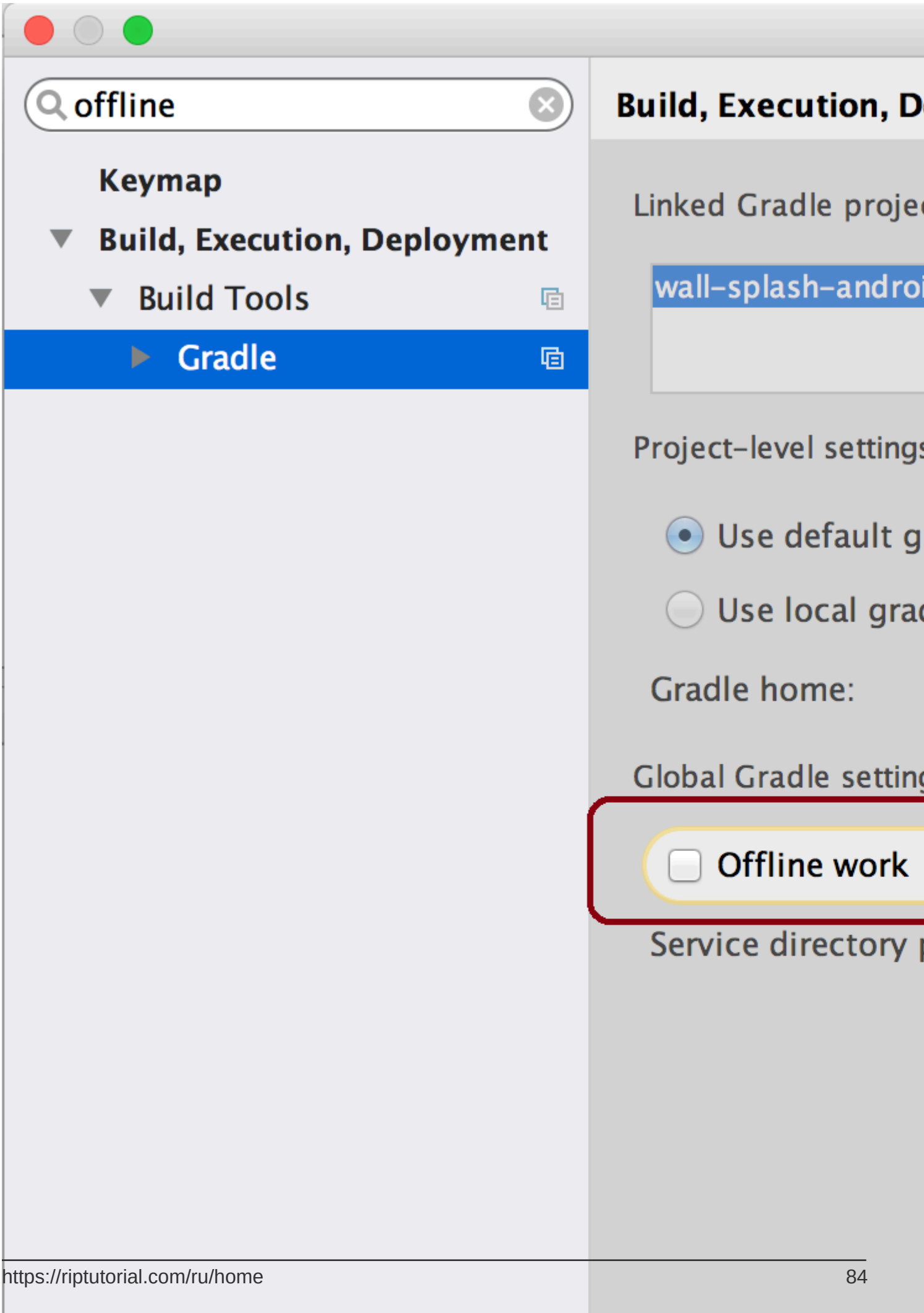
Перейдя в меню «Настройки»> «Ключ», появится всплывающее окно, отображающее все `Editor Actions` с их именем и ярлыками. Некоторые из `Editor Actions` не имеют ярлыков. Поэтому щелкните правой кнопкой мыши на этом и добавьте новый ярлык к этому. Проверьте изображение ниже



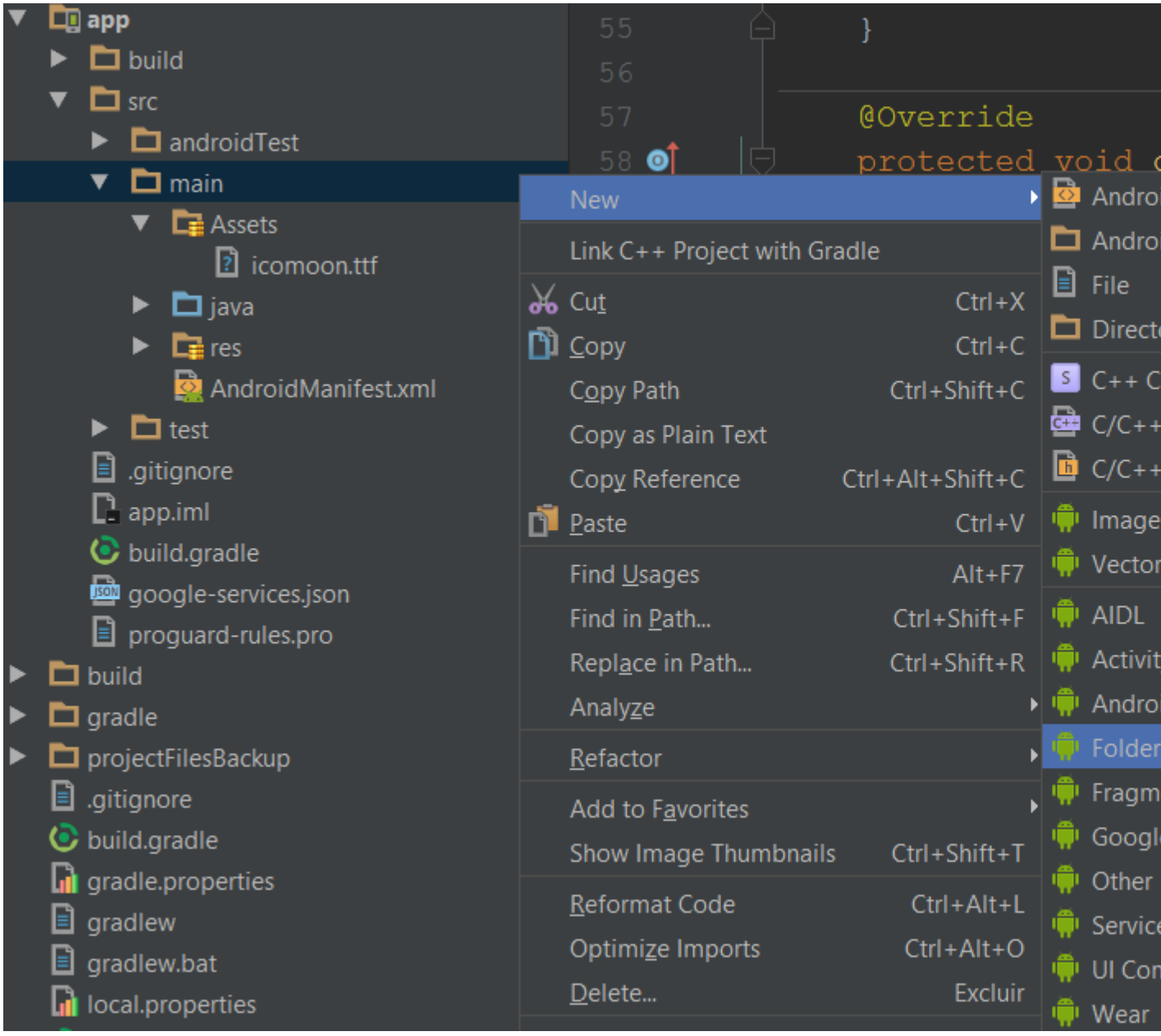
Проект строительства «Градл» берет навсегда

Android Studio -> **Настройки** -> **Грейдл** -> Отметьте **автономную работу,** а затем перезапустите свою студию Android.

Справочный снимок экрана:


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

- Папка «Ресурсы» будет находиться в папке MAIN с тем же символом, что и папка RES.
  RES.
- В этом примере я помещаю файл шрифта.



Прочитайте Android Studio онлайн: https://riptutorial.com/ru/android/topic/107/android-studio

глава 12: Android Vk Sdk

Examples

Инициализация и логин

1. Создайте новое приложение здесь: создайте приложение
2. Выберите автономное приложение и подтвердите создание приложения с помощью SMS.
3. Введите имя **пакета для Android в** качестве текущего имени пакета. *Вы можете получить имя своего пакета внутри файла манифеста android, в самом начале.*
4. Получите свой **отпечаток сертификата** , выполнив эту команду в своей оболочке / cmd:

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

Вы также можете получить этот отпечаток с помощью самого SDK:

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

5. Добавьте полученный отпечаток пальца в свой **отпечаток пальца сертификата подписки для Android:** поле в настройках приложения Vk (где вы указали название своего пакета)

6. Затем добавьте это в свой файл gradle:

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

8. Инициализируйте SDK при запуске, используя следующий метод. Лучший способ - назвать его в методе 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);
}
```

Это лучший способ инициализировать VKSdk. Не используйте metid, где VK_ID должен быть помещен внутри strings.xml, потому что api не будет работать правильно после него.

9. Заключительный шаг - войти в систему с помощью 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();
            }
        });
    }
```

Прочитайте Android Vk Sdk онлайн: https://riptutorial.com/ru/android/topic/6046/android-vk-sdk

глава 13: Android Вещи

Examples

Управление сервомотором

В этом примере предполагается, что у вас есть сервопривод со следующими характеристиками, которые типичны:

- движение между 0 и 180 градусами
- период импульса 20 мс
- минимальная длительность импульса 0,5 мс
- максимальная длительность импульса 2,5 мс

Вам нужно проверить, соответствуют ли эти значения вашим аппаратным средствам, поскольку принуждение к выходу за пределы указанного рабочего диапазона может привести к повреждению сервопривода. Поврежденный сервопривод, в свою очередь, может повредить устройство Android Things. Пример класса `ServoController` состоит из двух методов: `setup()` и `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();
        }
    }
}
```

Вы можете узнать имена контактов, которые поддерживают PWM на вашем устройстве, следующим образом:

```
PeripheralManagerService service = new PeripheralManagerService();

for (String pinName : service.getPwmList() ) {
    Log.i("ServoControlled","Pwm pin found: " + pinName);
}
```

Чтобы ваш сервопривод колебался навсегда между 80 и 100 градусами, вы можете просто использовать следующий код:

```
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();
```

Вы можете скомпилировать и развернуть весь вышеприведенный код без фактического подключения сервомоторов к вычислительному устройству. Для подключения, обратитесь к вычислительному устройству распиновки диаграмме (например, Raspberry Pi 3 разводки график доступен здесь).

Затем вам нужно подключить сервопривод к Vcc, Gnd и сигналу.

Прочитайте Android Вещи онлайн: https://riptutorial.com/ru/android/topic/8938/android-вещи

глава 14: Android-x86 в VirtualBox

Вступление

Идея этого раздела - освещать, как установить и использовать VirtualBox с Android-x86 для целей отладки. Это сложная задача, поскольку существуют различия между версиями. На данный момент я собираюсь охватить 6.0, из которых я должен был работать, и тогда нам придется найти сходство.

Он не охватывает VirtualBox или Linux подробно, но показывает команды, которые я использовал, чтобы заставить его работать.

Examples

Настройка виртуальной машины

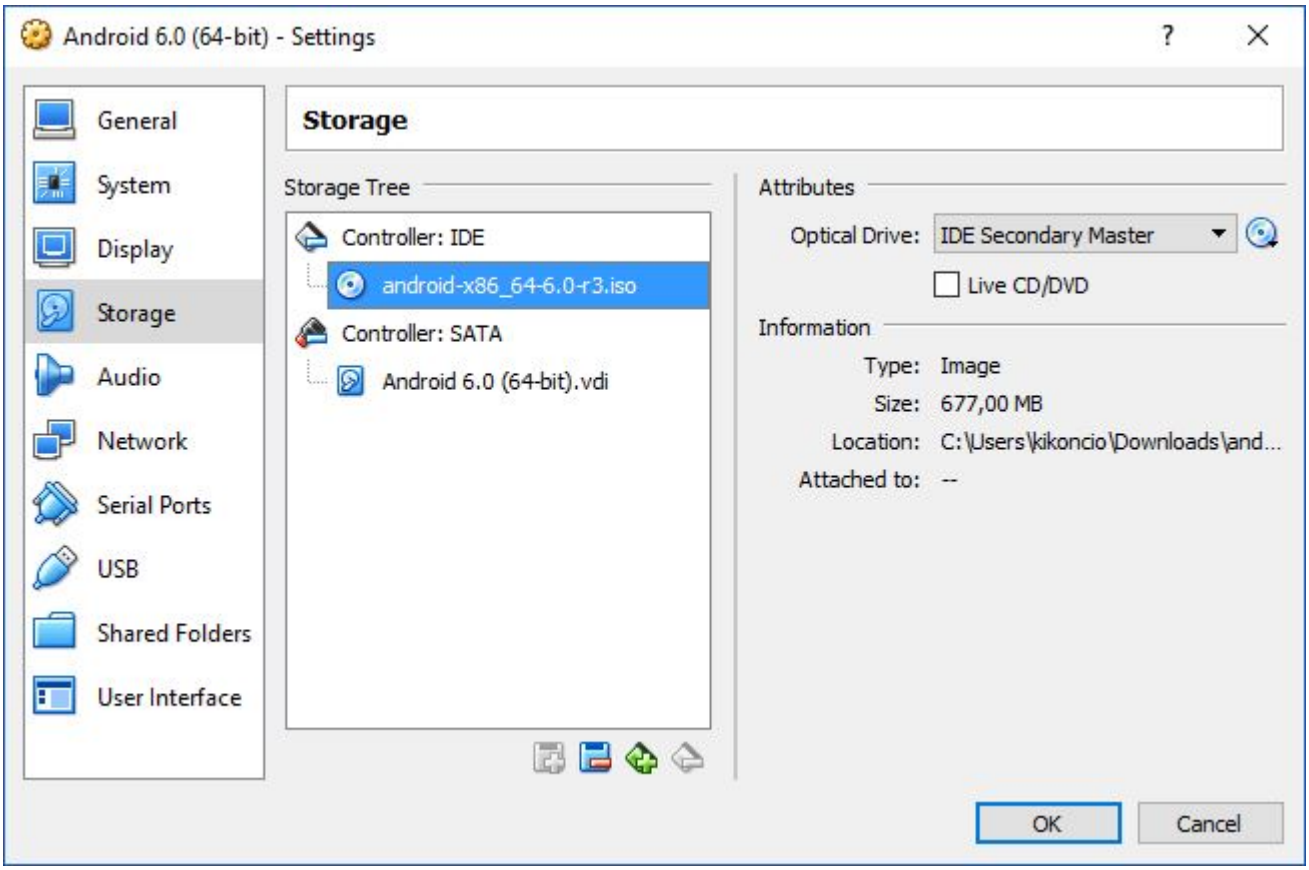
Это мои настройки VirtualBox:

- OS Тип: Linux 2.6 (у меня 64-разрядный пользователь, потому что мой компьютер может его поддерживать)
- Размер виртуального жесткого диска: 4 ГБ
- Память о раме: 2048
- Видеопамять: 8M
- Звуковое устройство: Sound Blaster 16.
- Сетевое устройство: PCnet-Fast III, подключенное к NAT. Вы также можете использовать модемный адаптер, но вам нужен сервер DHCP в вашей среде.

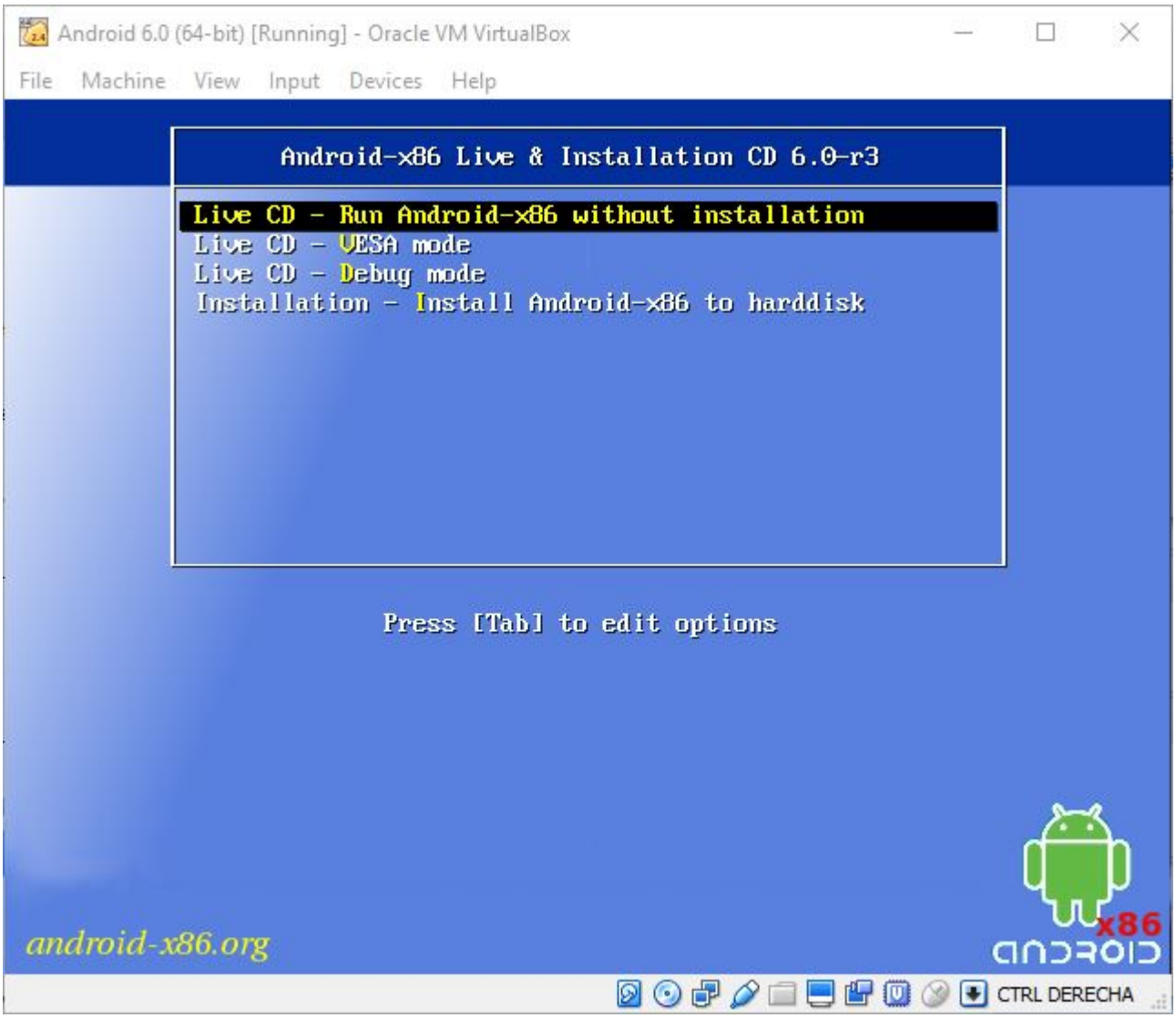
Изображение, используемое с этой конфигурацией, было андроид-x86_64-6.0-r3.iso (оно 64 бит) загружено с http://www.android-x86.org/download . Я полагаю, что он также работает с 32-битной версией.

Настройка виртуального жесткого диска для поддержки SDCARD

С созданным виртуальным жестким диском загрузите виртуальную машину с изображением android-x86 в оптическом диске.



После загрузки вы можете увидеть меню grub Live CD



Выберите параметр «Режим отладки», затем вы увидите приглашение оболочки. Это оболочка busybox. Вы можете получить больше оболочки, переключившись между виртуальной консолью Alt-F1 / F2 / F3.

Создайте два раздела с помощью fdisk (некоторые другие версии будут использовать cfdisk). Отформатируйте их до ext3. Затем перезагрузите компьютер:

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

Затем введите:

«n» (новый раздел)

«p» (первичный раздел)

«1» (1-й раздел)

«1» (первый цилиндр)

«261» (выберите цилиндр, мы оставим 50% диска для второго раздела)

«2» (2-й раздел)

«262» (262-й цилиндр)

«522» (выберите последний цилиндр)

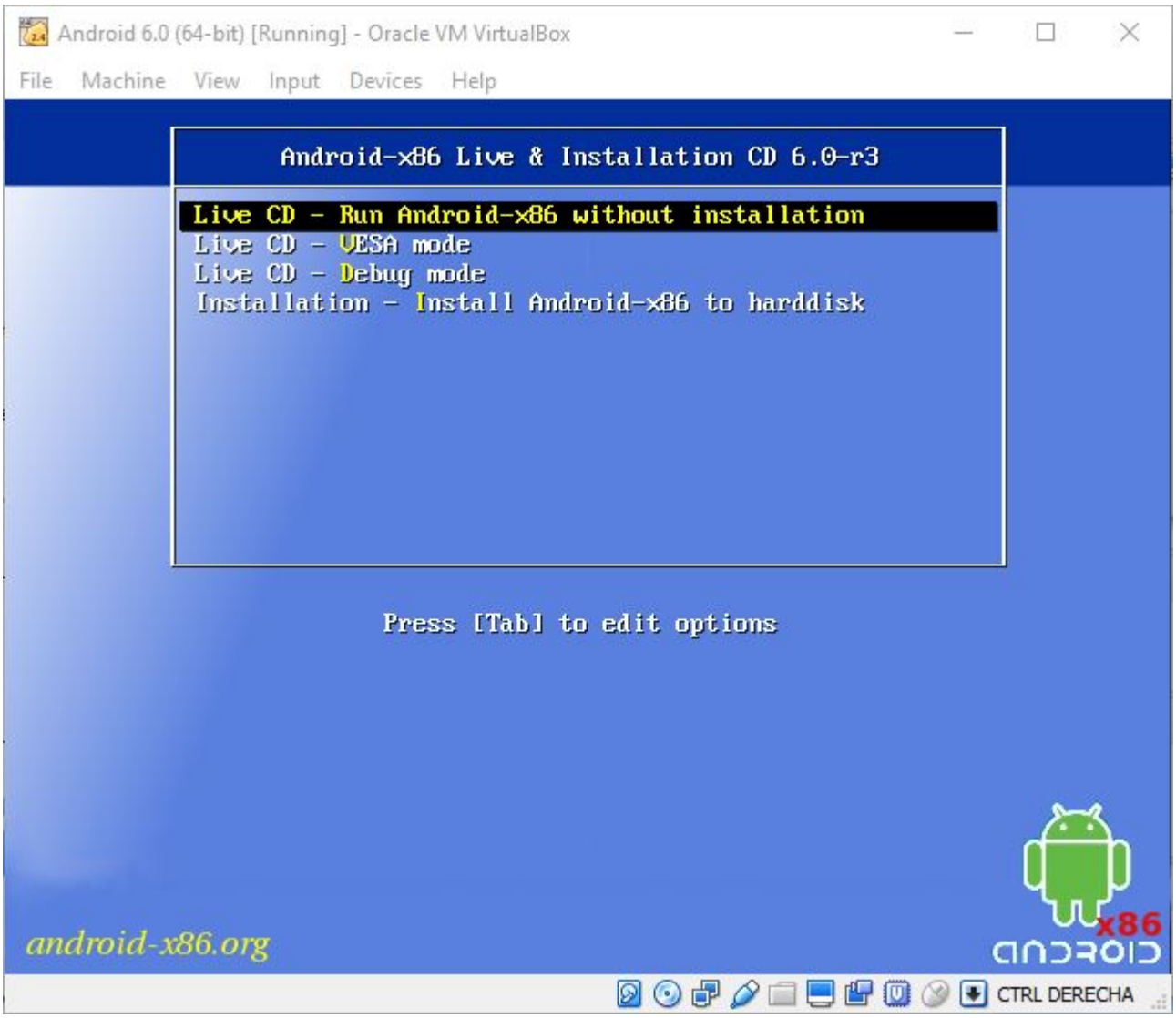
«w» (записать раздел)

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

Когда вы перезапустите виртуальную машину и появится меню grub, и вы сможете отредактировать строку загрузки ядра, чтобы добавить `DATA=sda1 SDCARD=sda2` , чтобы указать на SD-карту или раздел данных.

Установка в разделе

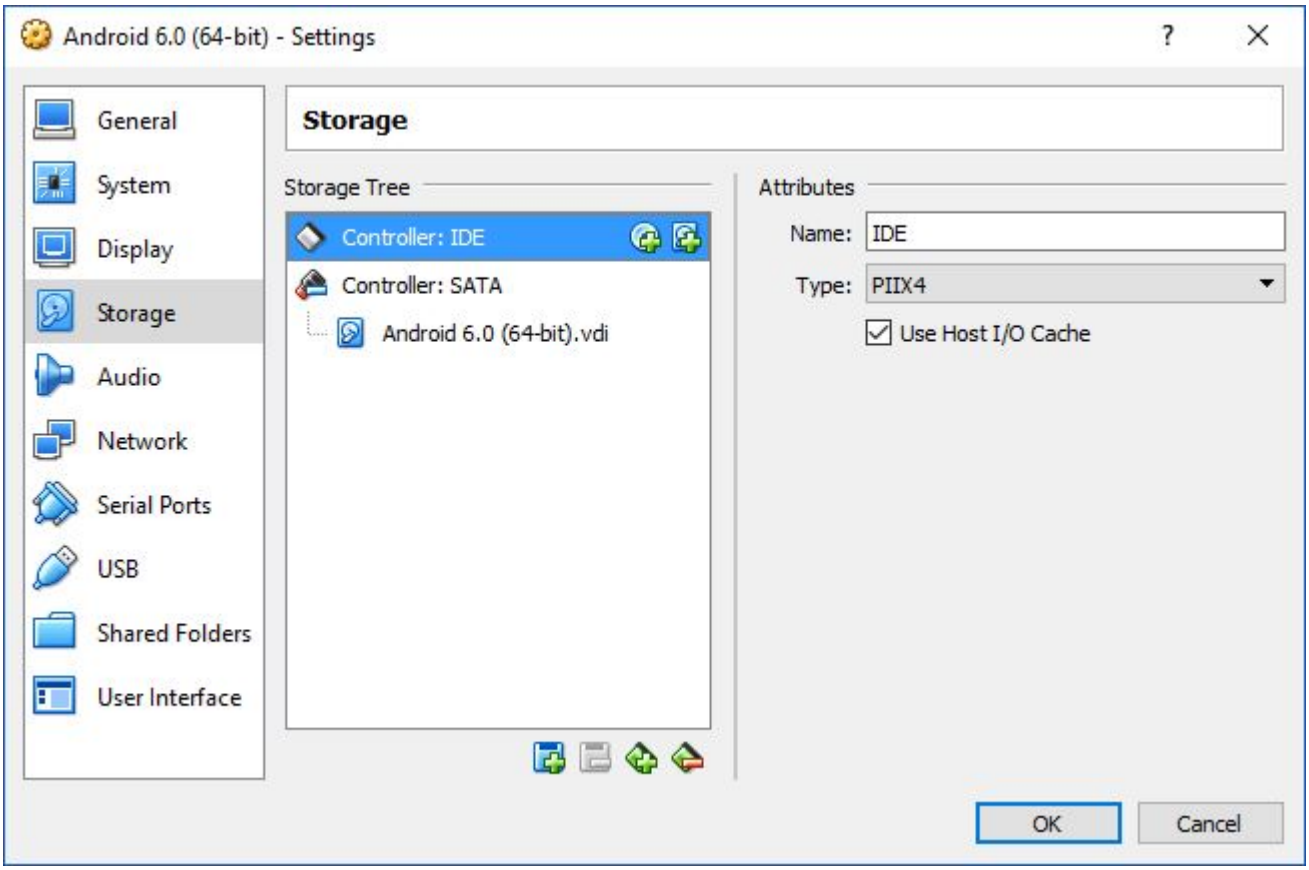
С созданным виртуальным жестким диском загрузите виртуальную машину с изображением android-x86 в качестве оптического диска.



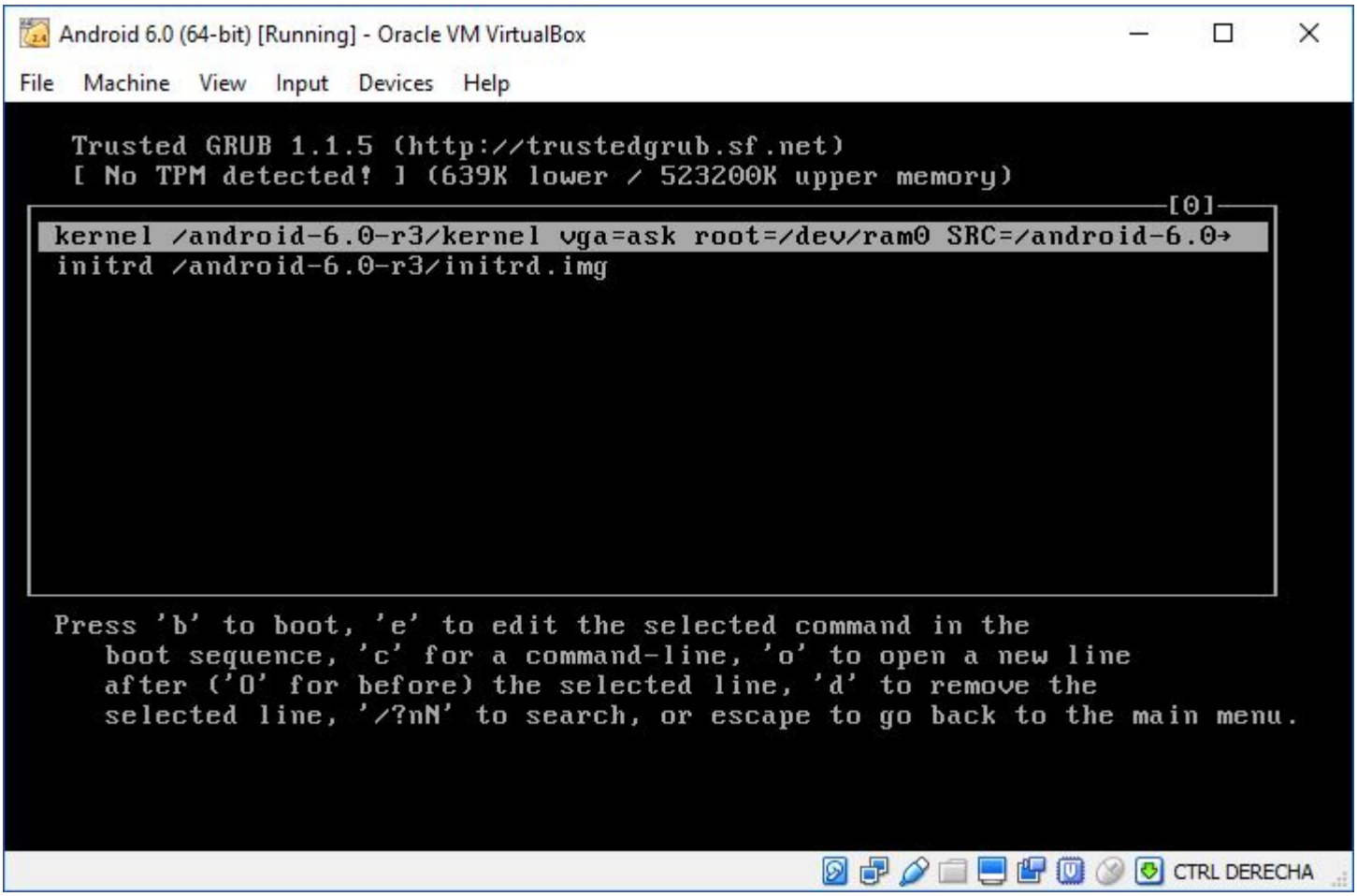
В настройках загрузки Live CD выберите «Установка - Установка Android на жесткий диск»

Выберите раздел sda1 и установите Android, и мы установим grub.

Перезагрузите виртуальную машину, но убедитесь, что изображение не находится на оптическом диске, поэтому оно может перезапускаться с виртуального жесткого диска.



В меню grub нам нужно отредактировать ядро, как в опции «Android-x86 6.0-r3», и нажмите e.

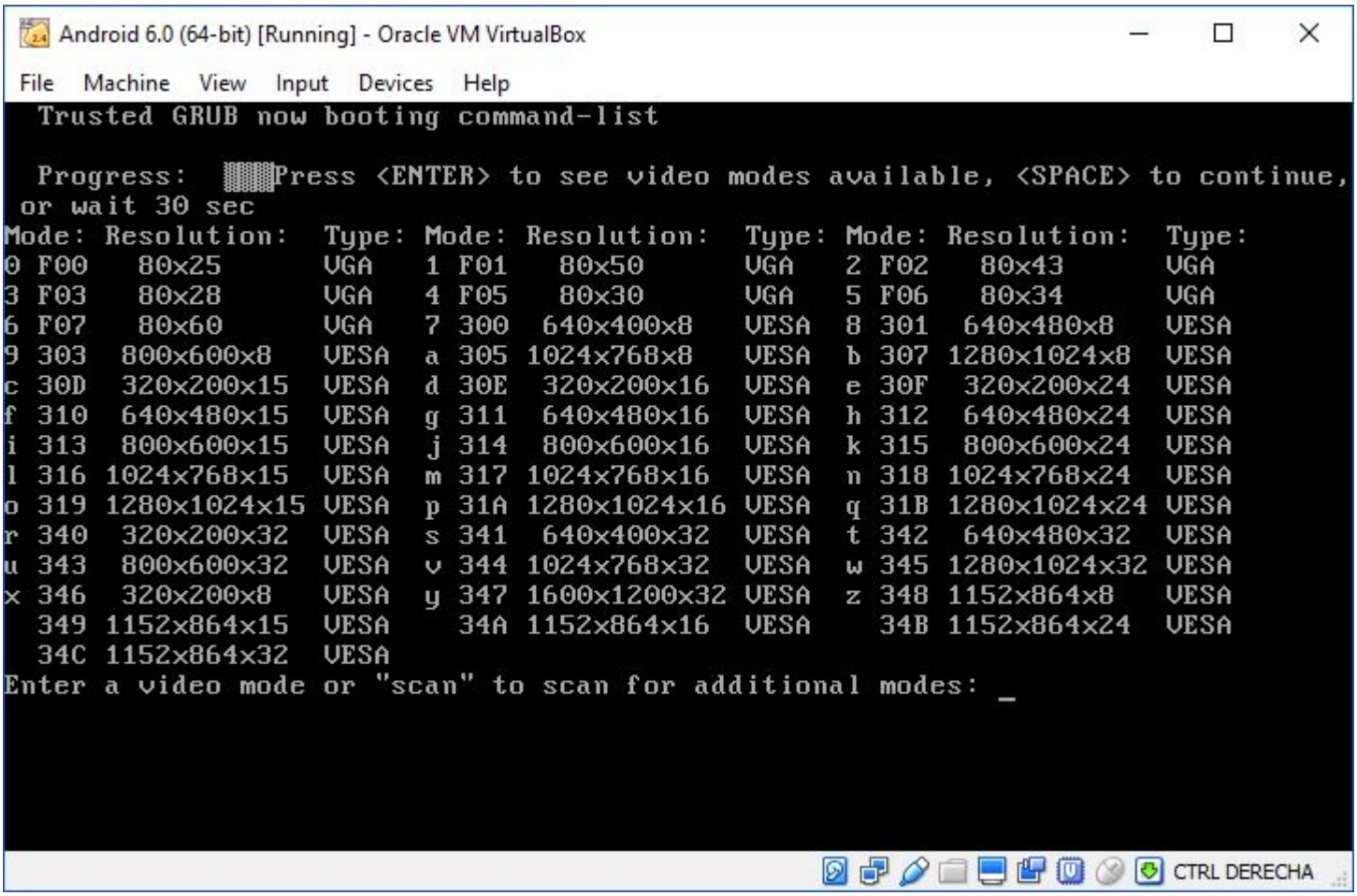


Затем мы заменяем «quiet» на «vga = ask» и добавляем опцию «SDCARD = sda2»

В моем случае строка ядра выглядит так:

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

Нажмите b для загрузки, затем вы сможете выбрать размер экрана, нажав ENTER (опция `vga=ask`)



После запуска мастера установки выберите язык. Я мог выбрать английский (США) и испанский (США), и мне не удалось выбрать какой-либо другой.

Прочитайте Android-x86 в VirtualBox онлайн:
https://riptutorial.com/ru/android/topic/9903/android-x86-в-virtualbox

глава 15: API Android Places

Examples

Пример использования места выбора

Place Picker - это действительно простой виджет пользовательского интерфейса, предоставляемый Places API. Он предоставляет встроенную карту, текущее местоположение, близлежащие места, возможности поиска и автозаполнение.

Это пример использования виджета Place Picker UI.

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

```
    }
}
```

Получение текущих мест с помощью API мест

Вы можете получить текущее местоположение и локальные места пользователя, используя API Google Адресов .

`PlaceDetectionApi.getCurrentPlace()` первых, вы должны вызвать метод `PlaceDetectionApi.getCurrentPlace()` для извлечения локального бизнеса или других мест. Этот метод возвращает объект `PlaceLikelihoodBuffer` который содержит список объектов `PlaceLikelihood` . Затем вы можете получить объект `Place` , вызвав метод `PlaceLikelihood.getPlace()` .

Важно: вы должны запросить и получить разрешение `ACCESS_FINE_LOCATION` , чтобы ваше приложение `ACCESS_FINE_LOCATION` получать точную информацию о местоположении.

```
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());
}
```

Размещение интеграции автозаполнения

Функция автозаполнения в API Google Адресов для Android предоставляет предсказания мест пользователям. В то время как пользовательские типы в окне поиска, автозаполнение показывает места в соответствии с запросами пользователя.

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"/>
```

Добавление нескольких завершенных действий 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.
           }
        }
    }
```

Фильтры типа места для PlaceAutocomplete

В некоторых сценариях нам может понадобиться сузить результаты, показанные **PlaceAutocomplete,** в определенной стране или, возможно, показывать только регионы. Этого можно достичь, установив **AutocompleteFilter** на намерение. Например, если я хочу посмотреть только на места типа REGION и принадлежать только Индии, я бы сделал следующее:

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 запустится автоматически, и вы сможете выбрать место из результатов, которые будут только типа **REGION** и будут принадлежать только указанной стране. Цель также может быть запущена одним нажатием кнопки.

Прочитайте API Android Places онлайн: https://riptutorial.com/ru/android/topic/4111/api-android-places

глава 16: API Google Диска

Вступление

Google Drive - это служба хостинга файлов, созданная **Google** . Он предоставляет службу хранения файлов и позволяет пользователю загружать файлы в облаке, а также делиться с другими людьми. Используя Google Drive API, мы можем синхронизировать файлы между компьютером или мобильным устройством и областью Google Диска.

замечания

легальный

Если вы используете Android API Google Диска в своем приложении, вы должны включить текст атрибуции Google Play Services в раздел «Юридические уведомления» в своем приложении.

Рекомендуется включать юридические уведомления в качестве отдельного элемента меню или как часть элемента меню «О программе».

Вы можете позвонить в `GooglePlayServicesUtil.getOpenSourceSoftwareLicenseInfo()` чтобы получить текст атрибуции во время выполнения.

Examples

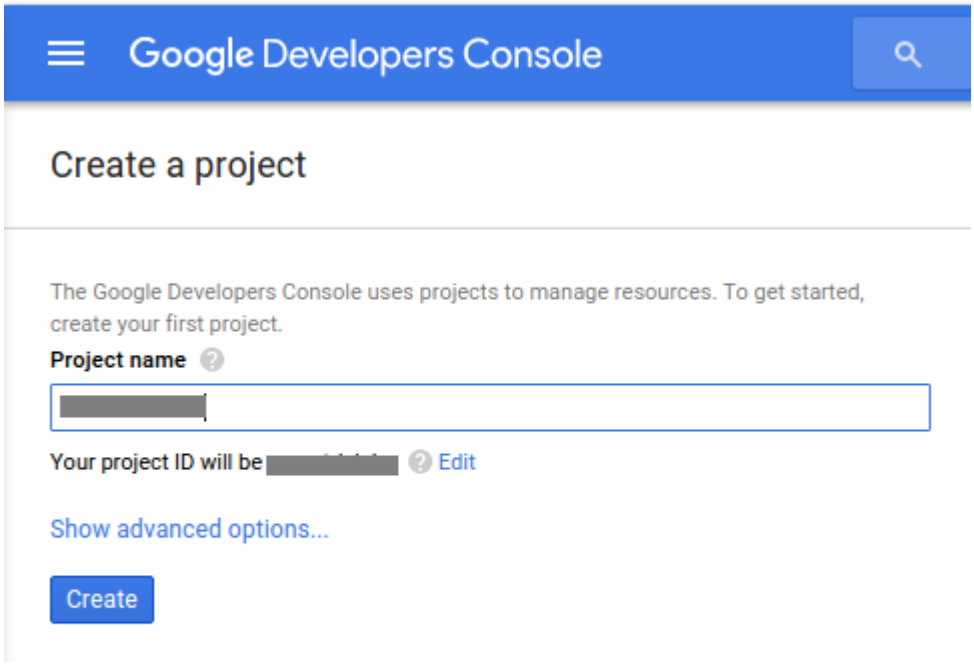
Интеграция Google Диска в Android

Создание нового проекта в консоли разработчика Google

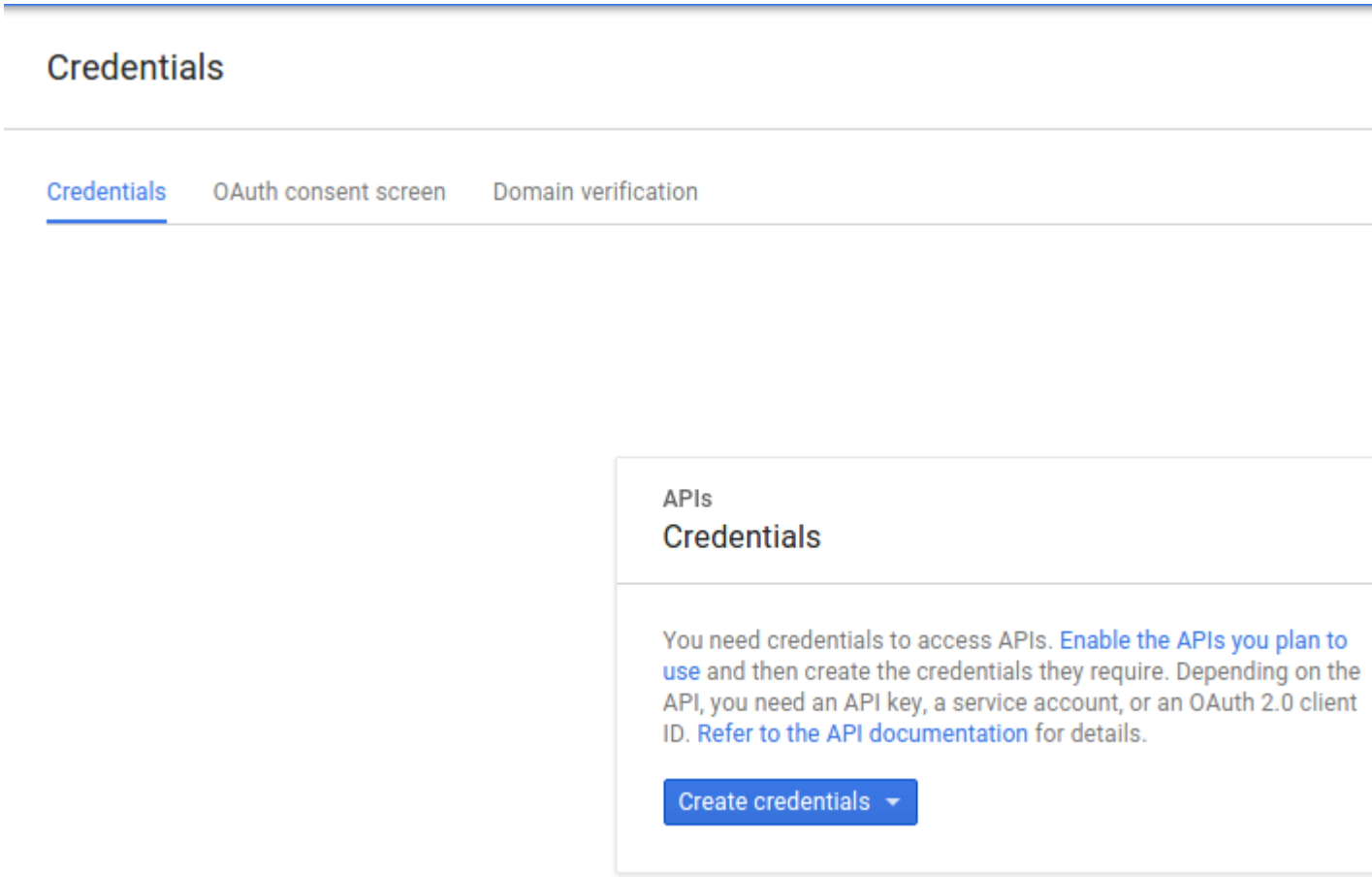
Чтобы интегрировать приложение Android с Google Диском, создайте учетные данные проекта в Google Developers Console. Итак, нам нужно создать проект на консоли Google Developer.

Чтобы создать проект в Google Developer Console, выполните следующие действия:

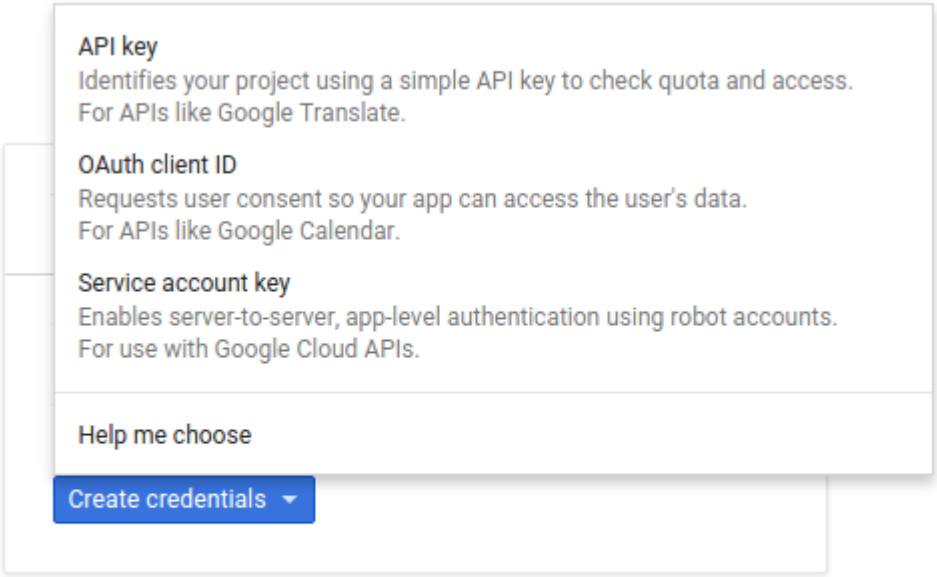
- Перейдите в Google Developer Console для Android. Введите **название проекта** в поле ввода и нажмите кнопку « **Создать»** , чтобы создать новый проект на консоли разработчика Google.



- Нам нужно создать учетные данные для доступа к API. Итак, нажмите кнопку « **Создать учетные данные**» .

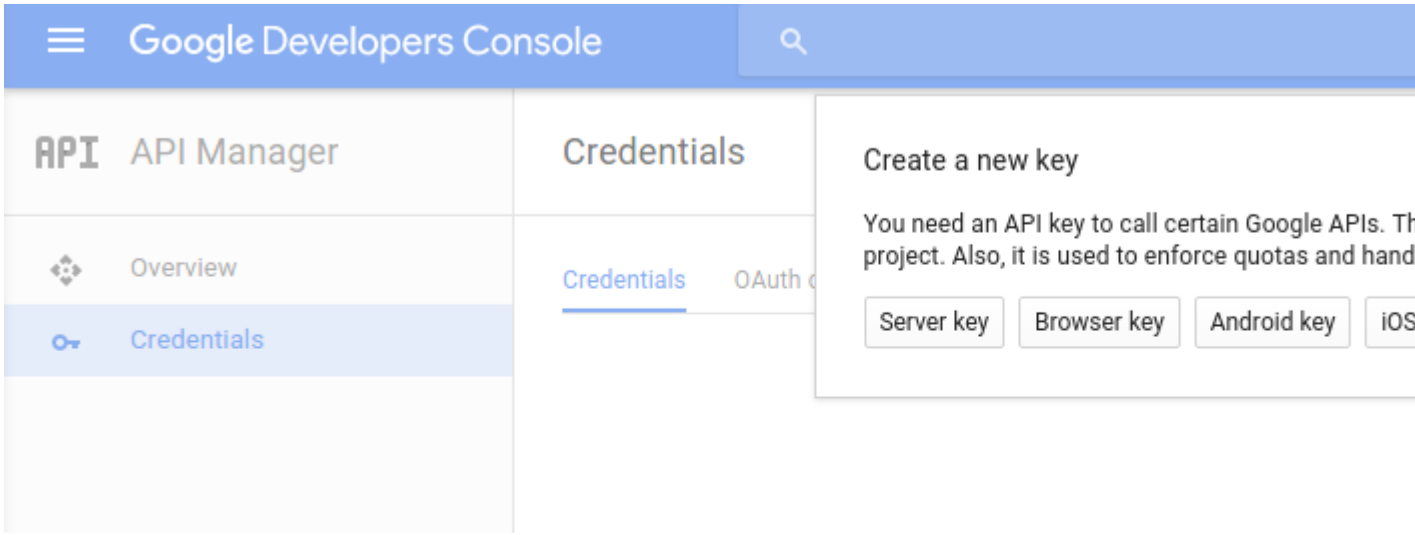


- Теперь откроется всплывающее окно. Нажмите кнопку **API Key** в списке, чтобы

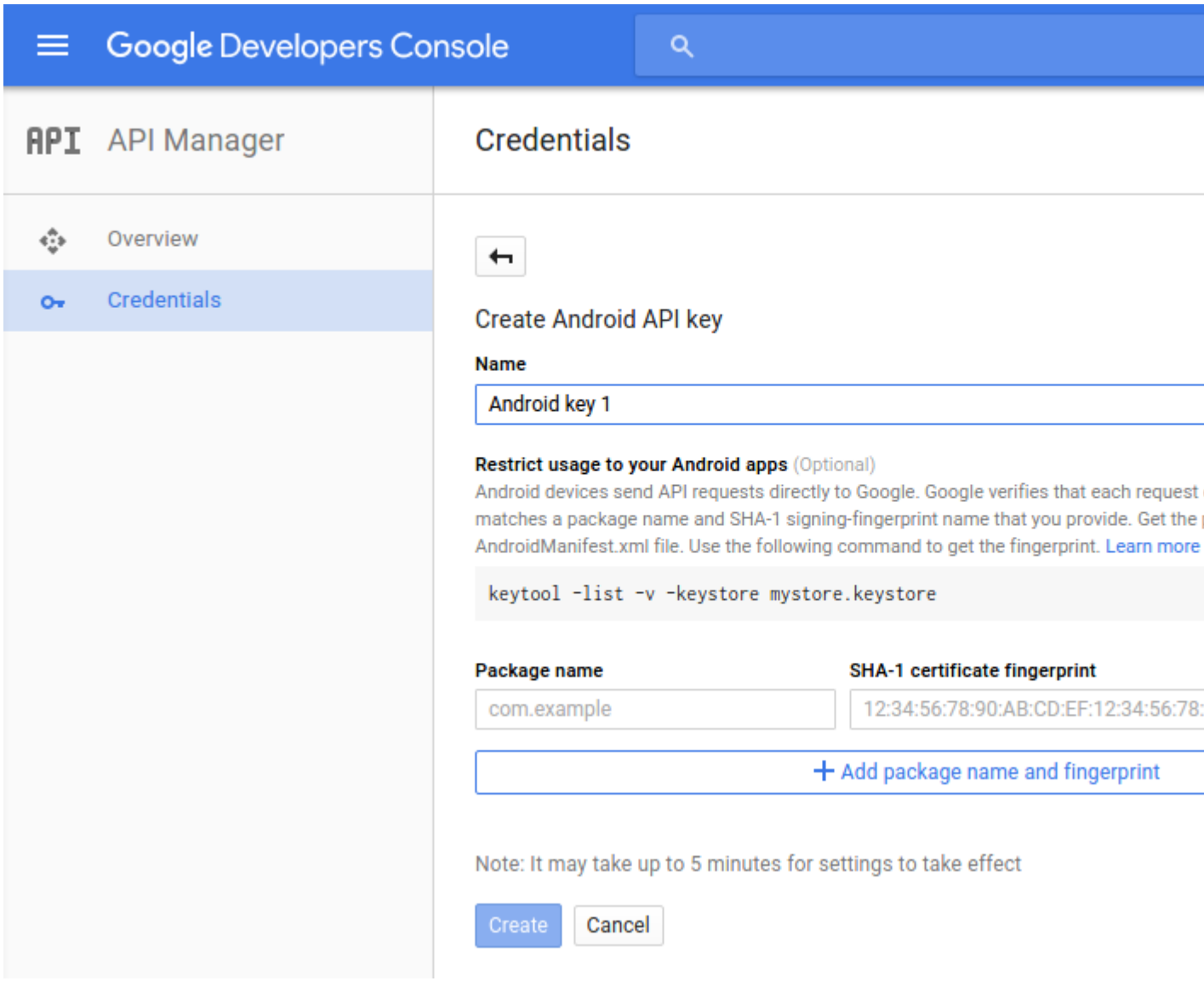


создать ключ API.

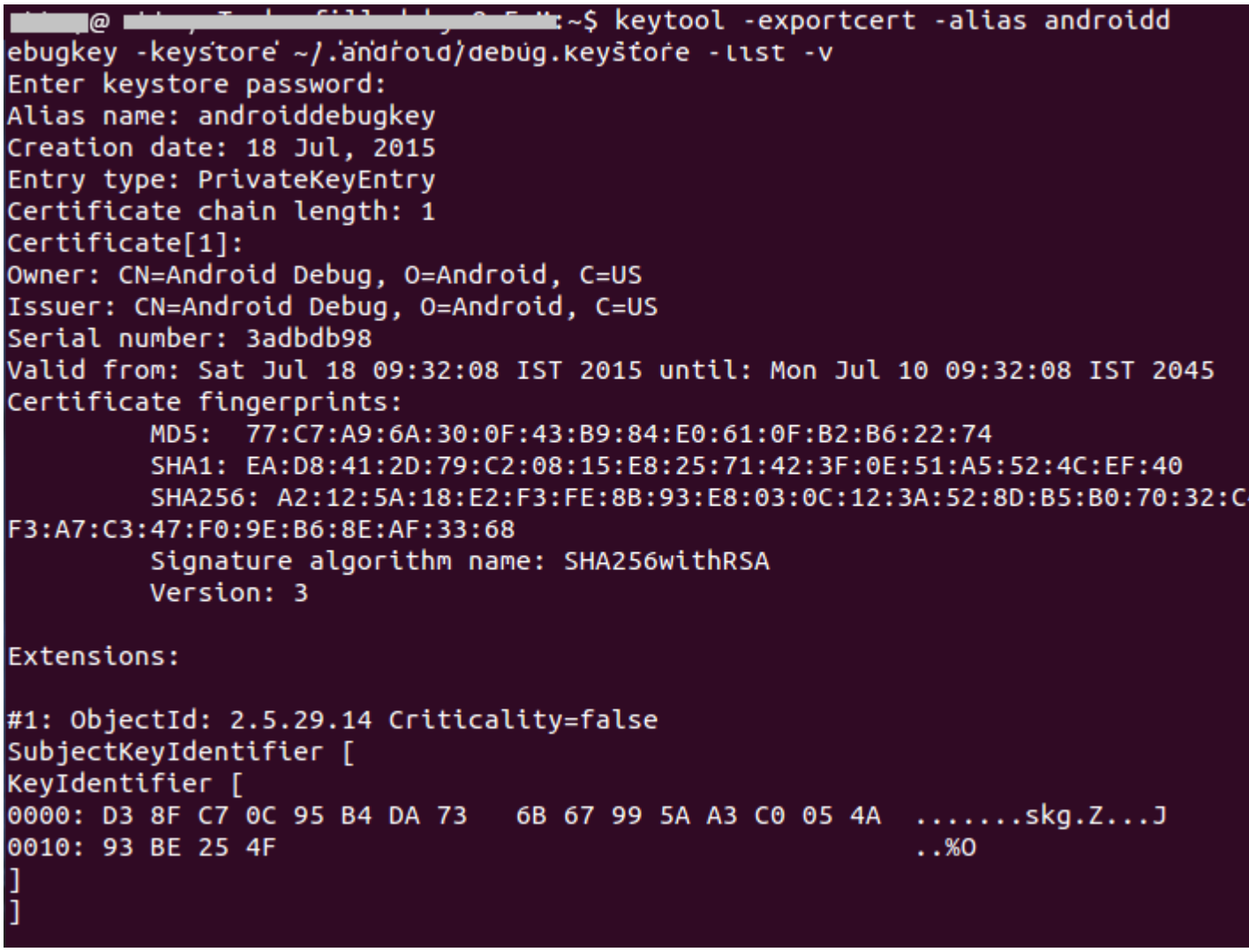
- Нам нужен ключ API для вызова API Google для Android. Итак, нажмите на **Android-ключ,** чтобы определить свой Android-проект.



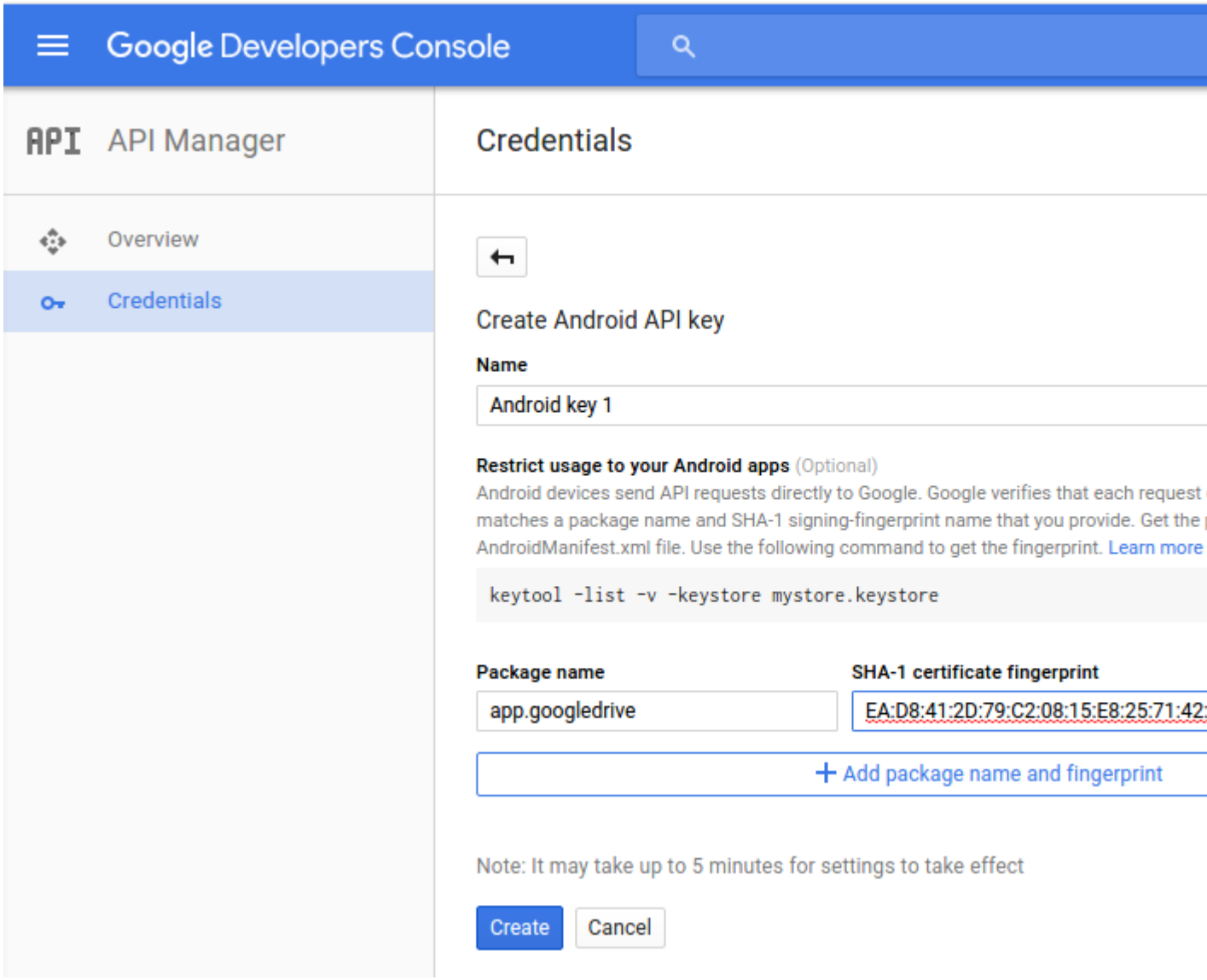
- Затем нам нужно добавить имя пакета Android Project и **SHA-1 отпечатка пальца** в поля ввода для создания ключа API.



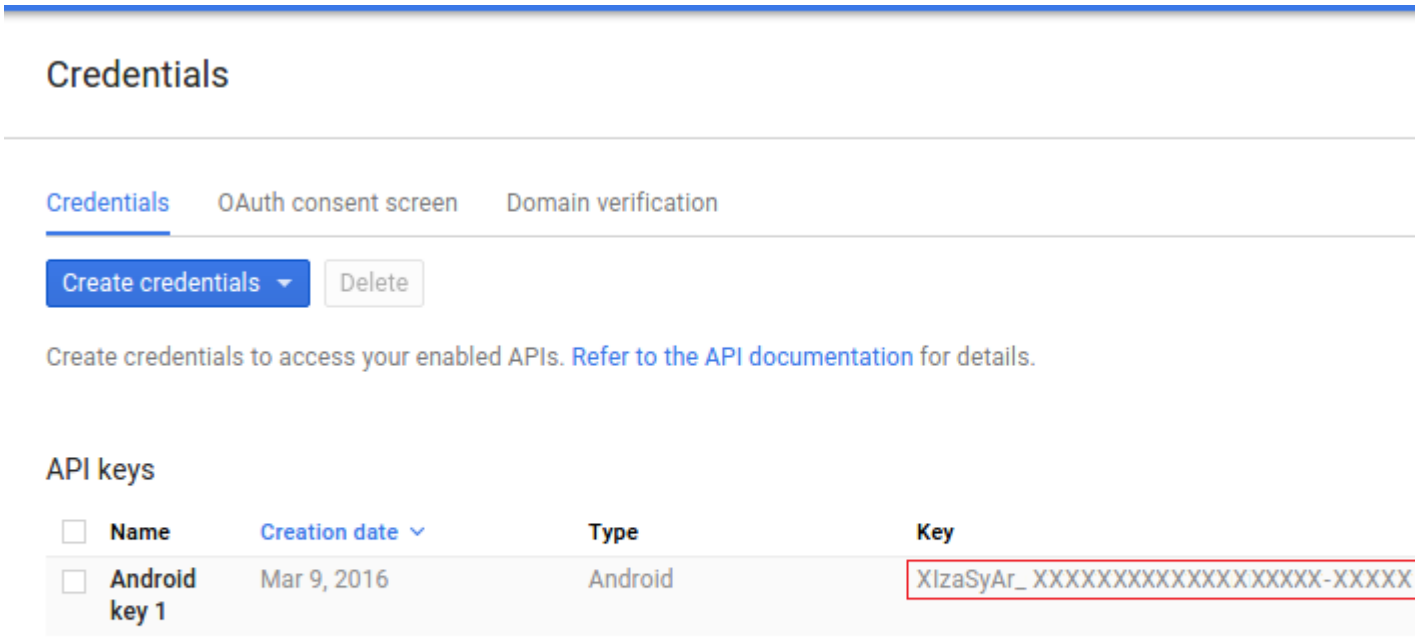
- Нам нужно создать **отпечаток SHA-1** . Итак, откройте свой терминал и запустите **утилиту Keytool,** чтобы получить отпечаток SHA1. При запуске утилиты Keytool вам необходимо предоставить **пароль хранилища ключей** . Пароль по умолчанию для keytool - **«android»** . `keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -list -v`



- Теперь добавьте **имя пакета** и **отпечаток SHA-1** в поля ввода на странице учетных данных. Наконец, нажмите кнопку «Создать», чтобы создать ключ API.



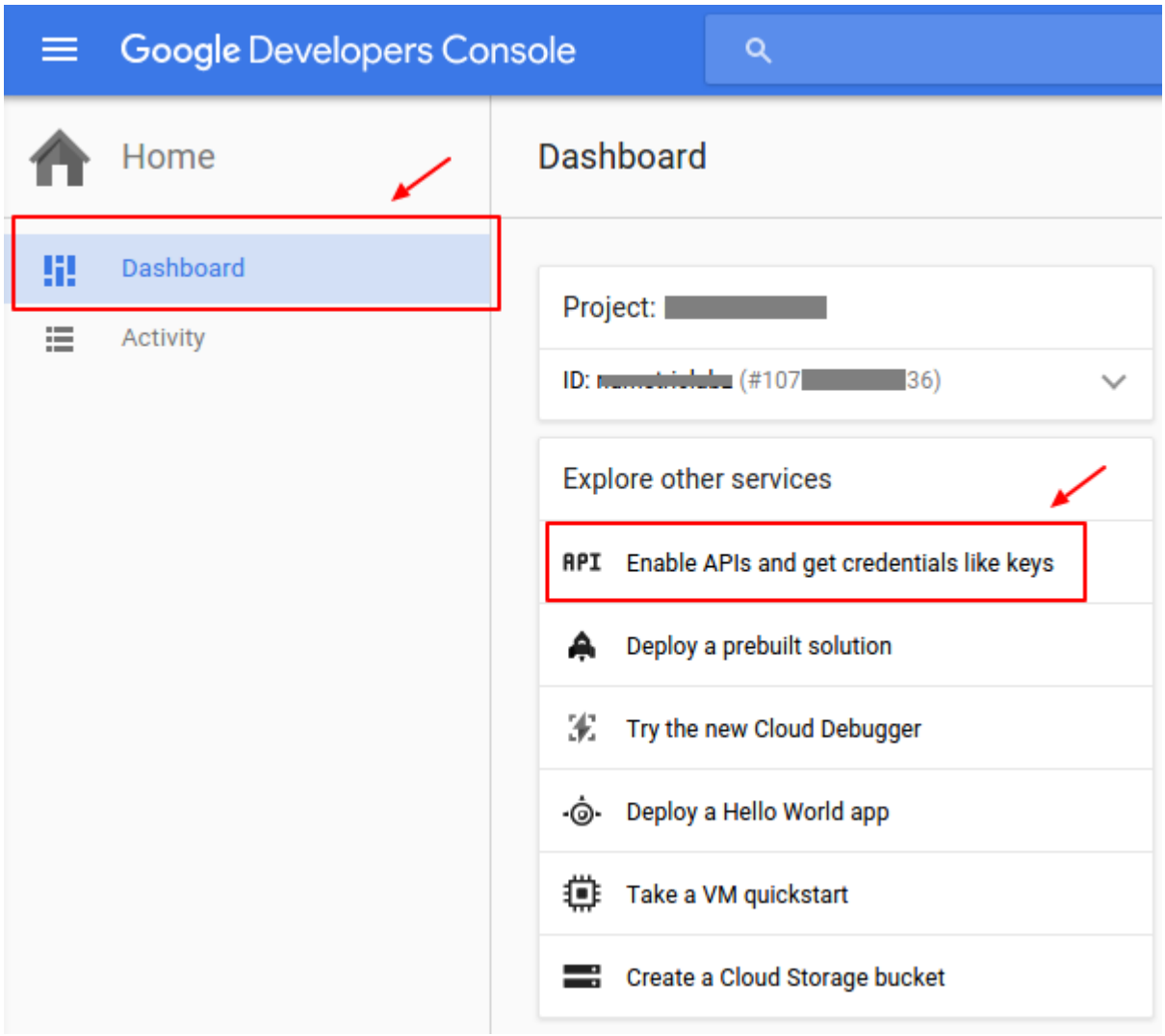
- Это создаст ключ API для Android. Мы будем использовать этот ключ API для интеграции приложения Android с Google Диском.



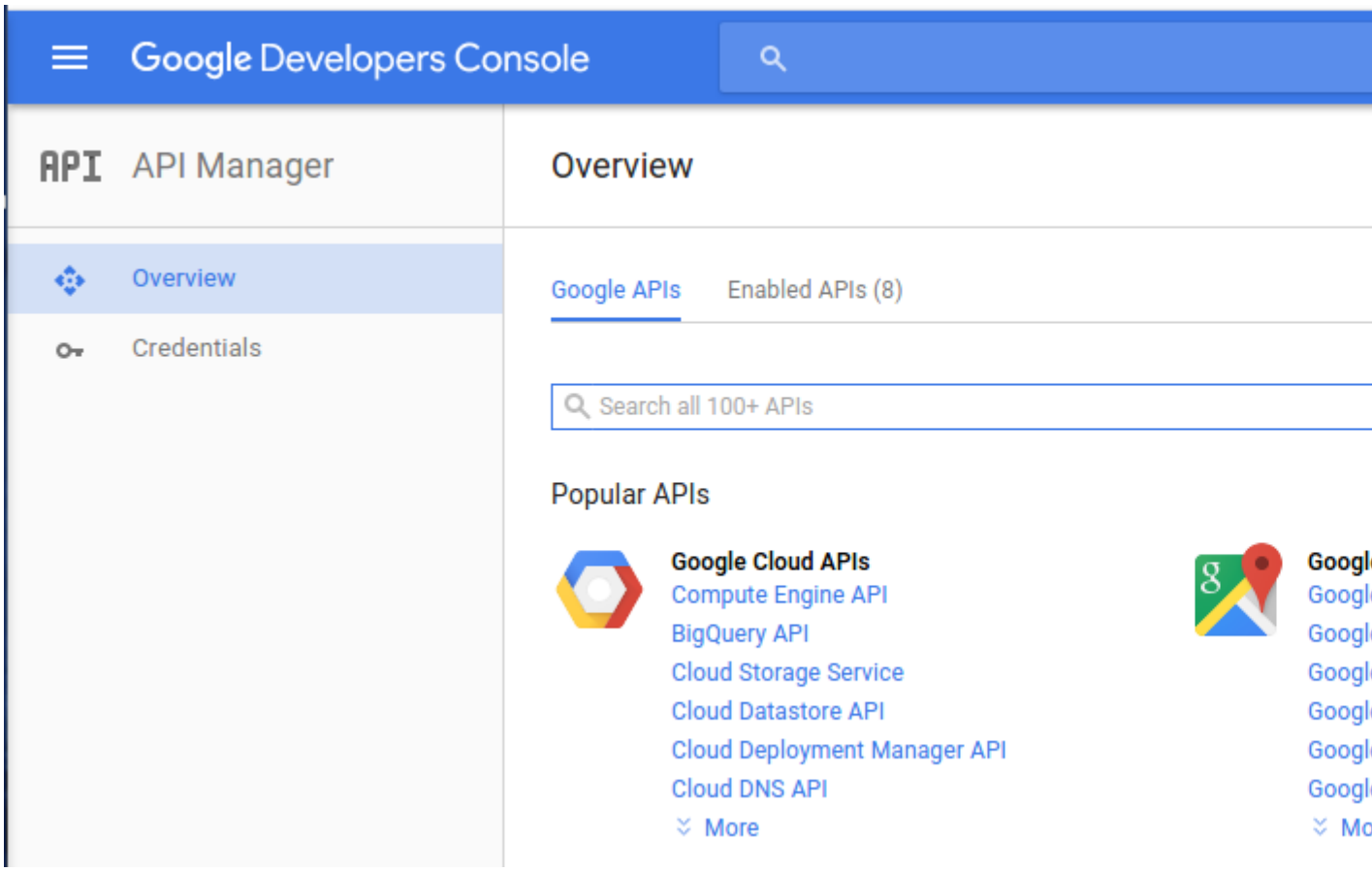
Включить API Диска Google

Нам нужно включить Google Drive Api для доступа к файлам, хранящимся на Google Диске, из приложения Android. Чтобы включить API Google Диска, выполните следующие действия:

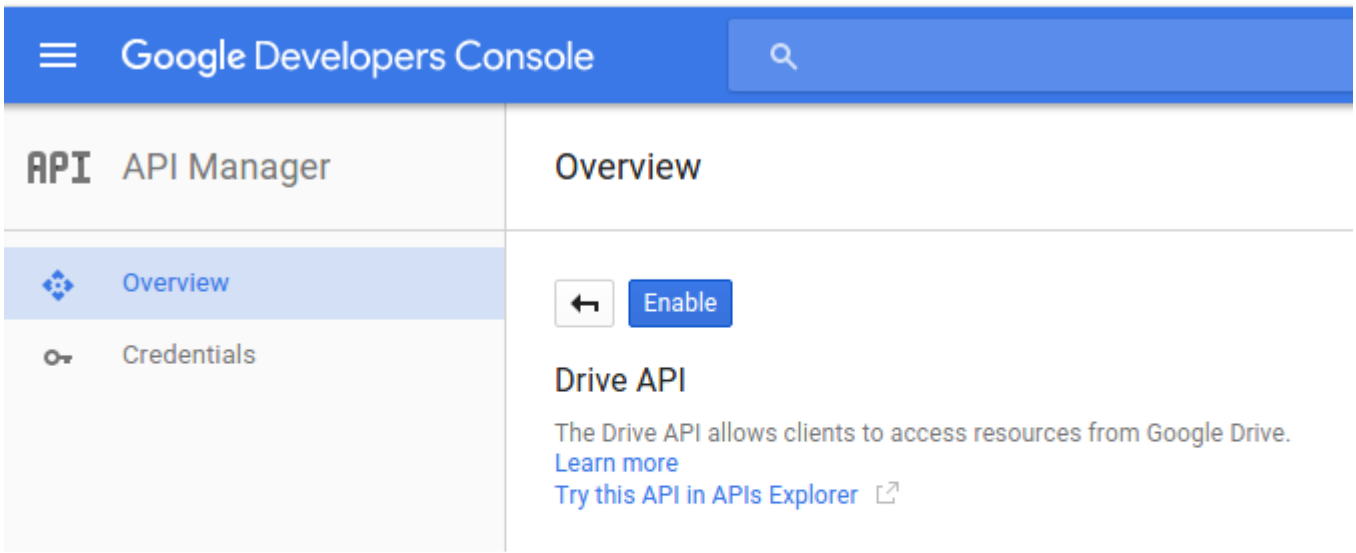
- Перейдите на панель инструментов консоли разработчика Google и нажмите « **Включить API», чтобы получить учетные данные, такие как ключи,** после чего вы увидите популярные API Google.



- Нажмите ссылку **API Диска,** чтобы открыть страницу обзора API Google Диска.



- Нажмите кнопку «Включить», чтобы включить API дисков Google. Он позволяет клиенту получить доступ к Google Диску.



Добавить разрешение в Интернете

App необходим доступ в **Интернет** Google Drive файлы. Используйте следующий код для настройки разрешений Интернета в файле AndroidManifest.xml:

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

Добавление сервисов Google Play

Мы будем использовать **API сервисов Google Play,** который включает в себя **API Android для Google Диска** . Итак, нам нужно настроить SDK сервисов Google Play в приложении Android. Откройте файл `build.gradle` (app module) и добавьте SDK сервисов Google Play в качестве зависимостей.

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

Добавить ключ API в файл манифеста

Чтобы использовать Google API в приложении для Android, нам нужно добавить ключ API и версию Службы Google Play в файле AndroidManifest.xml. Добавьте теги метаданных в тег файла AndroidManifest.xml.

Подключить и авторизировать API Google Диска Android

Нам необходимо аутентифицировать и подключить Android-приложение **Google Диска** с Android-приложением. Авторизация **Google Диска Android API** обрабатывается **GoogleApiClient** . Мы будем использовать **метод GoogleApiClient** внутри **метода 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();
    }
```

Отключить API Google Deive для Android

Когда активность прекратится, мы **отключим** соединение Google Android Android API с Android-приложением, вызвав метод **disconnect ()** внутри метода **onStop ()** .

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

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

Реализовать обратные вызовы соединения и подключенный прослушиватель

Мы реализуем Connection Callbacks и Connection Failed Listener клиента Google API в файле MainActivity.java, чтобы узнать статус подключения клиента Google API. Эти слушатели предоставляют **onConnected (), onConnectionFailed (), onConnectionSuspended ()** метод обработки проблем подключения между приложением и Диском.

Если пользователь разрешил приложение, **вызывается** метод **onConnected ()** . Если пользователь не имеет авторизованного приложения, **вызывается** метод **onConnectionFailed (),** и пользователю отображается сообщение о том, что ваше приложение не имеет права доступа к Google Диску. В случае приостановки соединения вызывается метод **onConnectionSuspended ()** .

Вам необходимо реализовать **ConnectionCallbacks** и **OnConnectionFailedListener** в своей деятельности. Используйте следующий код в файле 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");
    }
```

Создание файла на Google Диске

Мы добавим файл на Google Диск. Мы будем использовать метод `createFile()` объекта `Drive` для создания файла программно на Google Диске. В этом примере мы добавляем новый текстовый файл в корневую папку пользователя. Когда файл добавляется, нам нужно указать начальный набор метаданных, содержимого файла и родительской папки.

Нам нужно создать метод обратного вызова `CreateMyFile()` и в рамках этого метода используйте объект « `Drive` для извлечения ресурса `DriveContents` . Затем мы передаем клиент API на объект « `Drive` и вызываем `driveContentsCallback` обратного вызова `driveContentsCallback` для обработки результата `DriveContents` .

`DriveContents` содержит временную копию двоичного потока файла, доступного только для приложения.

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

Обработчик результатов DriveContents

Для обработки ответа требуется проверить, был ли вызов успешным или нет. Если вызов был успешным, мы можем получить ресурс `DriveContents` .

Мы создадим обработчик результатов `DriveContents` . В рамках этого метода мы вызываем метод `CreateFileOnGoogleDrive()` и передаем результат `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);
                    }
                }
            }
        };
```

Создать файл программно

Для создания файлов нам нужно использовать объект `MetadataChangeSet` . Используя этот объект, мы устанавливаем заголовок (имя файла) и тип файла. Кроме того, мы должны использовать метод `createFile()` класса `DriveFolder` и передать API-интерфейс клиента Google, объект `MetaDataChangeSet` и `driveContents` для создания файла. Мы вызываем обратный вызов обработчика результата для обработки результата созданного файла.

Мы используем следующий код для создания нового текстового файла в корневой папке пользователя:

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

Результат обработки созданного файла

Следующий код создаст метод обратного вызова для обработки результата созданного файла:

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

Прочитайте API Google Диска онлайн: https://riptutorial.com/ru/android/topic/10646/api-google-диска

глава 17: API Twitter

Examples

Создание логина с помощью кнопки Twitter и присоединение к ней обратного вызова

1. Внутри макета добавьте кнопку «Вход» со следующим кодом:

```
<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. В Управлении или фрагменте, который отображает кнопку, вам необходимо создать и прикрепить Обратный звонок к кнопке Logina следующим образом:

```
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. Передайте результат проверки подлинности обратно на кнопку:

```
@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);
}
```

Примечание. Если вы используете TwitterLoginButton во фрагменте, используйте следующие шаги:

```
 @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. Добавьте следующие строки в ваши зависимости **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;
 }
```

Прочитайте API Twitter онлайн: https://riptutorial.com/ru/android/topic/4801/api-twitter

глава 18: API камеры 2

параметры

параметр	подробности
`CameraCaptureSession`	Конфигурированный сеанс захвата для `CameraDevice` , используемый для захвата изображений с камеры или обработки изображений, снятых с камеры в том же сеансе ранее
`CameraDevice`	Представление одной камеры, подключенной к Android-устройству
`CameraCharacteristics`	Свойства, описывающие CameraDevice. Эти свойства фиксируются для данного CameraDevice и могут быть запрошены через интерфейс `getCameraCharacteristics(String)` с помощью `getCameraCharacteristics(String)`
`CameraManager`	Диспетчер системных служб для обнаружения, характеризации и подключения к `CameraDevices` . Вы можете получить экземпляр этого класса, вызвав `Context.getSystemService()`
`CaptureRequest`	Неизменяемый пакет настроек и выходов, необходимых для захвата одного изображения с устройства камеры. Содержит конфигурацию оборудования для захвата (датчик, объектив, вспышка), конвейер обработки, алгоритмы управления и выходные буферы. Также содержит список целевых поверхностей для отправки данных изображения для этого захвата. Может быть создан с использованием экземпляра `CaptureRequest.Builder` , полученного путем вызова `createCaptureRequest(int)`
`CaptureResult`	Подмножество результатов захвата одного изображения с датчика изображения. Содержит подмножество окончательной конфигурации для аппаратуры захвата (датчик, объектив, вспышка), конвейер обработки, алгоритмы управления и выходные буферы. Он производится с помощью `CameraDevice` после обработки `CaptureRequest`

замечания

- API-интерфейсы Camera2 доступны в API 21+ (Lollipop и выше)

- Даже если на Android-устройстве официально установлено 21 + ROM, нет гарантии, что он реализует API Camera2, это полностью зависит от производителя для его реализации или нет (например, LG G2 имеет официальную поддержку Lollipop, но не API Camera2)
- В Camera2 камера («Камера1») устарела
- С большой мощностью приходит большая ответственность: легче использовать его при использовании этих API.
- Помните, что если вы хотите только сделать снимок в своем приложении и просто получить его, вам **не** нужно использовать Camera2, вы можете открыть приложение камеры с помощью Intent и получить его обратно

Examples

Предварительный просмотр основной камеры в TextureView

В этом случае, строя против API 23, так что разрешения также обрабатываются.

Вы должны добавить в манифесту следующие разрешения (везде, где используется уровень API):

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

Мы собираемся создать активность (Camera2Activity.java), которая заполняет `TextureView` предварительным просмотром камеры устройства.

Активность, которую мы собираемся использовать, - это типичная AppCompatActivity:

```
public class Camera2Activity extends AppCompatActivity {
```

Атрибуты (вам может потребоваться прочитать весь пример, чтобы понять его часть)

`MAX_PREVIEW_SIZE` гарантированный Camera2 API, - 1920x1080

```
private static final int MAX_PREVIEW_WIDTH = 1920;
private static final int MAX_PREVIEW_HEIGHT = 1080;
```

`TextureView.SurfaceTextureListener` обрабатывает несколько событий жизненного цикла в `TextureView` . В этом случае мы слушаем эти события. Когда SurfaceTexture готова, мы инициализируем камеру. Когда размер изменяется, мы настраиваем предварительный просмотр, поступающий с камеры соответственно

```
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) {
    }

};
```

`CameraDevice` представляет собой камеру одного физического устройства. В этом атрибуте мы сохраняем идентификатор текущей `CameraDevice`

```
private String mCameraId;
```

Это представление (`TextureView`), которое мы будем использовать для «рисования» предварительного просмотра камеры

```
private TextureView mTextureView;
```

`CameraCaptureSession` для предварительного просмотра камеры

```
private CameraCaptureSession mCaptureSession;
```

Ссылка на открытую `CameraDevice`

```
private CameraDevice mCameraDevice;
```

`Size` предварительного просмотра камеры.

```
private Size mPreviewSize;
```

`CameraDevice.StateCallback` вызывается, когда `CameraDevice` меняет свое состояние

```
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();
    }

};
```

Дополнительный поток для запуска задач, которые не должны блокировать пользовательский интерфейс

```
private HandlerThread mBackgroundThread;
```

`Handler` для выполнения задач в фоновом режиме

```
private Handler mBackgroundHandler;
```

`ImageReader` который обрабатывает захват неподвижных изображений

```
private ImageReader mImageReader;
```

`CaptureRequest.Builder` для предварительного просмотра камеры

```
private CaptureRequest.Builder mPreviewRequestBuilder;
```

`CaptureRequest` сгенерированный `mPreviewRequestBuilder`

```
private CaptureRequest mPreviewRequest;
```

`Semaphore` для предотвращения выхода приложения перед закрытием камеры.

```
private Semaphore mCameraOpenCloseLock = new Semaphore(1);
```

Постоянный идентификатор запроса на разрешение

```
private static final int REQUEST_CAMERA_PERMISSION = 1;
```

Методы жизненного цикла Android

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_camera2);

    mTextureView = (TextureView) findViewById(R.id.texture);
}

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

    // When the screen is turned off and turned back on, the SurfaceTexture is already
    // available, and "onSurfaceTextureAvailable" will not be called. In that case, we can
open
    // a camera and start preview from here (otherwise, we wait until the surface is ready in
    // the SurfaceTextureListener).
    if (mTextureView.isAvailable()) {
        openCamera(mTextureView.getWidth(), mTextureView.getHeight());
    } else {
        mTextureView.setSurfaceTextureListener(mSurfaceTextureListener);
    }
}

@Override
public void onPause() {
    closeCamera();
    stopBackgroundThread();
    super.onPause();
}
```

Связанные с Camera2 методы

Это методы, которые используют 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);
    }
}
```

Закрывает текущую камеру

```
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();
    }
}
```

Устанавливает переменные-члены, связанные с камерой

```
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();
    }
}
```

Создает новый `CameraCaptureSession` для предварительного просмотра камеры.

```
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();
    }
}
```

Методы, связанные с разрешениями для Android API 23+

```
private void requestCameraPermission() {
    if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.CAMERA))
{
        new AlertDialog.Builder(Camera2Activity.this)
                .setMessage("R string request permission")
                .setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener()
{
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        ActivityCompat.requestPermissions(Camera2Activity.this,
                                new String[]{Manifest.permission.CAMERA},
                                REQUEST_CAMERA_PERMISSION);
                    }
                })
                .setNegativeButton(android.R.string.cancel,
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                finish();

                            }
                        })
                .create();

    } else {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA},
                REQUEST_CAMERA_PERMISSION);
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
```

```
    if (requestCode == REQUEST_CAMERA_PERMISSION) {
        if (grantResults.length != 1 || grantResults[0] != PackageManager.PERMISSION_GRANTED)
{
            Toast.makeText(Camera2Activity.this, "ERROR: Camera permissions not granted",
Toast.LENGTH_LONG).show();
        }
    } else {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

Методы фонового потока / обработчика

```
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();
    }
}
```

Полезные методы

При выборе `Size` s, поддерживаемых камерой, выберите наименьший размер, который по крайней мере на большом уровне соответствует размеру текстурного вида, и размер такого же размера, как соответствующий максимальный размер, и соотношение сторон которого соответствует указанному значению. Если этого не существует, выберите наибольший размер, который не превосходит соответствующий максимальный размер, и соотношение сторон которого соответствует указанному значению

```
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];
    }
}
```

Этот метод сопоставляет необходимое преобразование `Matrix` с `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);
}
```

Этот метод сравнивает два `Size` s на основе их областей.

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

Не так много, чтобы увидеть здесь

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

Прочитайте API камеры 2 онлайн: https://riptutorial.com/ru/android/topic/619/api-камеры-2

глава 19: API отпечатков пальцев в android

замечания

смотрите также

Проект Github

Android-разработчик blogspot

Сайт разработчика Android

Examples

Добавление сканера отпечатков пальцев в приложение для Android

Android поддерживает отпечаток api с Android 6.0 (Marshmallow) SDK 23

> Чтобы использовать эту функцию в своем приложении, сначала добавьте разрешение USE_FINGERPRINT в манифест.

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

Здесь следует следовать процедуре

> Сначала вам нужно создать симметричный ключ в Android Key Store с помощью KeyGenerator, который можно использовать только после аутентификации пользователя с помощью отпечатка пальца и передачи 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();
```

> Установив KeyGenParameterSpec.Builder.setUserAuthenticationRequired в true, вы можете разрешить использование ключа только после аутентификации пользователя, в том числе при аутентификации с помощью отпечатка пальца пользователя.

```
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);
```

> Затем начните прослушивать отпечаток пальца на датчике отпечатка пальца, вызвав FingerprintManager.authenticate с помощью Cipher, инициализированного созданным симметричным ключом. Или, альтернативно, вы можете вернуться на серверный проверенный пароль в качестве аутентификатора.

Создайте и инициализируйте `FingerprintManger` с помощью `fingerprintManger.class`

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

Для аутентификации используйте `FingerprintManger` api и создайте подкласс, используя

`FingerprintManager.AuthenticationCallback` и переопределить методы

```
onAuthenticationError
onAuthenticationHelp
onAuthenticationSucceeded
onAuthenticationFailed
```

Начать

Чтобы запустить проверку подлинности метода проверки подлинности вызова fingerPrint с криптографическим

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

отменить

прекратить прослушивание вызова сканера

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

> После проверки отпечатка пальца (или пароля) вызывается обратный вызов FingerprintManager.AuthenticationCallback # onAuthenticationSucceeded ().

```
@Override

public void onAuthenticationSucceeded(AuthenticationResult result) {

            }
```

Как использовать API Fingerprint API для сохранения паролей

пользователей

Этот класс вспомогательного класса взаимодействует с менеджером печати пальцев и выполняет шифрование и дешифрование пароля. Обратите внимание, что метод, используемый для шифрования в этом примере, - AES. Это не единственный способ шифрования и другие примеры . В этом примере данные шифруются и дешифруются следующим образом:

Шифрование:

1. Пользователь предоставляет помощнику желаемый незашифрованный пароль.
2. Пользователь должен предоставить отпечаток пальца.
3. После аутентификации помощник получает ключ от `KeyStore` и шифрует пароль с помощью `Cipher` .
4. Пароль и соль IV (IV воссозданы для каждого шифрования и не используются повторно) сохраняются в общих настройках, которые будут использоваться позже в процессе дешифрования.

Дешифрирование:

1. Пользователь просит расшифровать пароль.
2. Пользователь должен предоставить отпечаток пальца.
3. Помощник создает `Cipher` с использованием IV, и как только пользователь аутентифицируется, KeyStore получает ключ от `KeyStore` и расшифровывает пароль.

```
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());
```

```
            }
        }
    }
}
```

Эта деятельность ниже является очень простым примером того, как получить пароль, сохраненный пользователем, и взаимодействовать с помощником.

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

Прочитайте API отпечатков пальцев в android онлайн:
https://riptutorial.com/ru/android/topic/7523/api-отпечатков-пальцев-в-android

глава 20: API поддержки Google

замечания

Помните, что Snapshot API используется для запроса текущего состояния, в то время как Fence API постоянно проверяет наличие указанного состояния и отправляет обратные вызовы, когда приложение не работает.

В целом, есть несколько основных шагов, чтобы использовать API-интерфейс Snapshot или Fence API:

- Получить ключ API из консоли Google Developers
- Добавьте в манифест необходимые разрешения и ключ API:

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

- Инициализируйте `GoogleApiClient` где-нибудь, желательно в методе onCreate () вашей активности.

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

- Вызовите API по вашему выбору
- Результат анализа

Простым способом проверки необходимого разрешения пользователя является такой метод, как:

```
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

Получить текущую активность пользователя с помощью Snapshot API

Для одноразовых непостоянных запросов для физической активности пользователя используйте API-интерфейс 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());
        }
    });
```

Получить состояние наушников с помощью Snapshot API

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

Получить текущее местоположение с помощью Snapshot API

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

Получайте близлежащие места с помощью Snapshot API

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

Что касается получения данных в этих местах, вот несколько вариантов:

```
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()));
```

Получить текущую погоду с помощью API-интерфейса моментального снимка

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

Получите изменения в активности пользователей с помощью Fence API

Если вы хотите определить, когда ваш пользователь запускает или завершает такие действия, как ходьба, работа или любое другое действие класса `DetectedActivityFence` , вы можете создать забор для активности, которую вы хотите обнаружить, и получать уведомление, когда ваш пользователь запускает / завершает эту деятельность. Используя `BroadcastReceiver` , вы получите `Intent` с данными, которые содержат активность:

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

Теперь вы можете получить намерение с `BroadcastReceiver` для получения обратных вызовов, когда пользователь изменяет действие:

```
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;
        }
    }
}
```

Получить изменения для местоположения в определенном диапазоне с помощью Fence API

Если вы хотите обнаружить, когда пользователь входит в определенное место, вы можете создать забор для определенного местоположения с радиусом, который вы хотите, и получать уведомление, когда ваш пользователь входит или покидает место.

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

Теперь создайте BroadcastReciver для обновления обновлений в пользовательском состоянии:

```
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;
        }
    }
}
```

Прочитайте API поддержки Google онлайн: https://riptutorial.com/ru/android/topic/3361/api-поддержки-google

глава 21: AsyncTask

параметры

параметр	подробности
Params	тип параметров, отправленных в задачу при выполнении.
Прогресс	тип единиц прогресса, опубликованных во время фоновых вычислений
Результат	тип результата фонового вычисления.

Examples

Основное использование

В действиях и службах Android большинство обратных вызовов запускаются в основном потоке . Это упрощает обновление пользовательского интерфейса, но выполнение основных задач с процессором или I / O в основном потоке может привести к тому, что ваш пользовательский интерфейс приостановится и перестанет отвечать (официальная документация о том, что происходит).

Вы можете исправить это, поставив эти более тяжелые задачи на фоновый поток.

Один из способов сделать это - использовать AsyncTask , который обеспечивает основу для облегчения использования фонового потока, а также выполнения задач потока пользовательского интерфейса до, во время и после того, как фоновый поток завершил свою работу.

Методы, которые можно переопределить при расширении `AsyncTask` :

- `onPreExecute()` : вызывается в **потоке пользовательского интерфейса** перед выполнением задачи
- `doInBackground()` : вызывается **в фоновом потоке** сразу после того, как `onPreExecute()` завершает выполнение.
- `onProgressUpdate()` : вызывается в **потоке пользовательского интерфейса** после вызова `publishProgress(Progress...)` .
- `onPostExecute()` : вызывается в **потоке пользовательского интерфейса** после завершения фоновых вычислений

пример

```
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.");
        }
    }

}
```

Использование:

```
MyCustomAsyncTask asyncTask = new MyCustomAsyncTask<File, Void, String>();
// Run the task with a user supplied filename.
asyncTask.execute(userSuppliedFilename);
```

или просто:

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

Заметка

При определении `AsyncTask` мы можем передавать три типа между скобками `< >` . Определяется как `<Params, Progress, Result>` (см. **Раздел «Параметры»**)

В предыдущем примере мы использовали типы `<File, Void, String>` :

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

`Void` используется, если вы хотите пометить тип как неиспользуемый.

Обратите внимание, что вы не можете передавать примитивные типы (т. `int` , `float` и еще 6 других) в качестве параметров. В таких случаях вы должны передать свои классы-оболочки , например `Integer` вместо `int` , или `Float` вместо `float` .

Жизненный цикл AsyncTask и Activity

AsyncTasks не соответствуют жизненному циклу экземпляров действий. Если вы запустите AsyncTask внутри Activity и вы повернете устройство, действие будет уничтожено, и будет создан новый экземпляр. Но AsyncTask не умрет. Он будет продолжаться до тех пор, пока он не завершится.

Решение: AsyncTaskLoader

Одним из подклассов Loaders является AsyncTaskLoader. Этот класс выполняет ту же функцию, что и AsyncTask, но намного лучше. Он легче справляется с изменениями конфигурации деятельности и ведет себя в жизненных циклах фрагментов и действий. Приятно, что AsyncTaskLoader можно использовать в любой ситуации, в которой используется AsyncTask. В любое время данные должны загружаться в память для Activity / Fragment для обработки, AsyncTaskLoader может выполнять работу лучше.

Отмена AsyncTask

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

Это не остановит вашу задачу, если она была в процессе, она просто устанавливает отмененный флаг, который можно проверить, проверив возвращаемое значение `isCancelled()` (если ваш код запущен в данный момент), выполнив следующие действия:

```
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;
            }
        }
    }
}
```

Заметка

Если AsyncTask отменяется, а `doInBackground(Params... params)` все еще выполняется, тогда метод `onPostExecute(Result result)` **НЕ** будет вызываться после `doInBackground(Params... params)` . AsyncTask вместо этого вызовет `onCancelled(Result result)` чтобы указать, что задача была отменена во время выполнения.

Успех публикации

Иногда нам необходимо обновить ход вычислений, выполненный с помощью `AsyncTask` . Этот прогресс может быть представлен строкой, целым числом и т. Д. Для этого мы должны использовать две функции. Во-первых, нам нужно установить функцию `onProgressUpdate` , тип параметра которой совпадает с параметром второго типа нашей `AsyncTask` .

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

Во-вторых, мы должны использовать функцию `publishProgress` обязательно `doInBackground` функции `doInBackground` , и это все, предыдущий метод выполнит всю работу.

```
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;
 }
```

Загрузить изображение с помощью AsyncTask в Android

В этом руководстве объясняется, как загрузить изображение с помощью AsyncTask в Android. Пример ниже загружает изображение, пока отображается индикатор выполнения во время загрузки.

Понимание Android AsyncTask

Задача Async позволяет реализовать MultiThreading, не втирая руки в потоки. AsyncTask обеспечивает правильное и простое использование потока пользовательского интерфейса. Он позволяет выполнять фоновые операции и передавать результаты в потоке пользовательского интерфейса. Если вы делаете что-то изолированное, связанное с пользовательским интерфейсом, например, загружая данные в список в списке, используйте AsyncTask.

- AsyncTasks идеально подходит для коротких операций (максимум на несколько секунд).
- Асинхронная задача определяется тремя универсальными типами, называемыми Params, Progress и Result, и 4 шагами, называемыми `onPreExecute()` , `doInBackground()` , `onProgressUpdate()` и `onPostExecute()` .
- В `onPreExecute()` вы можете определить код, который должен быть выполнен до начала фоновой обработки.
- doInBackground имеет код, который необходимо выполнить в фоновом режиме, здесь, в `doInBackground()` мы можем отправить результаты несколько раз в поток событий методом publishProgress (), чтобы уведомить об обработке фона, мы можем просто вернуть результаты.
- `onProgressUpdate()` получает обновления прогресса от `doInBackground()` , который публикуется методом `publishProgress()` , и этот метод может использовать это обновление для обновления потока событий
- `onPostExecute()` обрабатывает результаты, возвращаемые методом `doInBackground()` .
- Используемые общие типы
    - Парамс, тип параметров, отправленных в задачу при выполнении
    - Прогресс, тип единиц прогресса, опубликованных во время фоновых вычислений.
    - Результат, тип результата фонового вычисления.
- Если задача async не использует какие-либо типы, тогда ее можно пометить как тип Void.
- Запущенную задачу async можно отменить, вызвав метод `cancel(boolean)` .

Загрузка изображения с помощью Android AsyncTask

ваш макет .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>
```

класс .java

```
package com.javatechig.droid;

import java.io.InputStream;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import android.app.Activity;
import android.app.ProgressDialog;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;

public class ImageDownladerActivity extends Activity {

    private ImageView downloadedImg;
    private ProgressDialog simpleWaitDialog;
    private String downloadUrl = "http://www.9ori.com/store/media/images/8ab579a656.jpg";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.asynch);
        Button imageDownloaderBtn = (Button) findViewById(R.id.downloadButton);

        downloadedImg = (ImageView) findViewById(R.id.imageView);

        imageDownloaderBtn.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
```

```
                // TODO Auto-generated method stub
                new ImageDownloader().execute(downloadUrl);
            }

        });
    }

    private class ImageDownloader extends AsyncTask {

        @Override
        protected Bitmap doInBackground(String... param) {
            // TODO Auto-generated method stub
            return downloadBitmap(param[0]);
        }

        @Override
        protected void onPreExecute() {
            Log.i("Async-Example", "onPreExecute Called");
            simpleWaitDialog = ProgressDialog.show(ImageDownladerActivity.this,
                    "Wait", "Downloading Image");

        }

        @Override
        protected void onPostExecute(Bitmap result) {
            Log.i("Async-Example", "onPostExecute Called");
            downloadedImg.setImageBitmap(result);
            simpleWaitDialog.dismiss();

        }

        private Bitmap downloadBitmap(String url) {
            // initilize the default HTTP client object
            final DefaultHttpClient client = new DefaultHttpClient();

            //forming a HttpGet request
            final HttpGet getRequest = new HttpGet(url);
            try {

                HttpResponse response = client.execute(getRequest);

                //check 200 OK for success
                final int statusCode = response.getStatusLine().getStatusCode();

                if (statusCode != HttpStatus.SC_OK) {
                    Log.w("ImageDownloader", "Error " + statusCode +
                            " while retrieving bitmap from " + url);
                    return null;

                }

                final HttpEntity entity = response.getEntity();
                if (entity != null) {
                    InputStream inputStream = null;
                    try {
                        // getting contents from the stream
                        inputStream = entity.getContent();

                        // decoding stream data back into image Bitmap that android
understands
                        final Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
```

```
                    return bitmap;
                } finally {
                    if (inputStream != null) {
                        inputStream.close();
                    }
                    entity.consumeContent();
                }
            }
        } catch (Exception e) {
            // You Could provide a more explicit error message for IOException
            getRequest.abort();
            Log.e("ImageDownloader", "Something went wrong while" +
                    " retrieving bitmap from " + url + e.toString());
        }

        return null;
    }
}
}
```

Поскольку в настоящее время для примеров нет поля комментариев (или я его не нашел или у меня нет разрешения на это), вот несколько комментариев об этом:

Это хороший пример того, что можно сделать с помощью AsyncTask.

Однако в настоящее время в этом примере возникают проблемы с

- возможные утечки памяти
- приложение, если произошел поворот экрана незадолго до завершения асинхронной задачи.

Подробнее см .:

- Передача активности как WeakReference во избежание утечек памяти
- http://stackoverflow.com/documentation/android/117/asynctask/5377/possible-problems-with-inner-async-tasks
- Избегайте утечки активности с помощью AsyncTask

Передача активности как WeakReference во избежание утечек памяти

Для AsyncTask обычно требуется ссылка на вызвавшуюся активность.

Если AsyncTask является внутренним классом Activity, вы можете напрямую ссылаться на него и любые переменные / методы-члены.

Если, однако, AsyncTask не является внутренним классом Activity, вам нужно будет передать ссылку Activity на AsyncTask. Когда вы это сделаете, одна потенциальная проблема, которая может возникнуть, заключается в том, что AsyncTask сохранит ссылку для Activity, пока AsyncTask не завершит свою работу в фоновом потоке. Если действие

завершено или убито до того, как будет выполняться фоновый поток AsyncTask, AsyncTask все равно будет ссылаться на Activity, поэтому сбор мусора невозможен.

В результате это приведет к утечке памяти.

Чтобы этого не произошло, используйте ссылку WeakReference в AsyncTask вместо прямой ссылки на Activity.

Вот пример AsyncTask, который использует 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();
        }
    }
}
```

Вызов AsyncTask из Activity:

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

Вызов AsyncTask из фрагмента:

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

Порядок исполнения

При первом вводе `AsyncTasks` выполнялись последовательно на одном фоновом потоке.

Начиная с `DONUT` , это было изменено на пул потоков, позволяющий нескольким задачам работать параллельно. Начиная с `HONEYCOMB` , задачи выполняются в одном потоке, чтобы избежать ошибок обычного приложения, вызванных параллельным выполнением.

Если вы действительно хотите выполнить параллельное выполнение, вы можете вызвать `executeOnExecutor(java.util.concurrent.Executor, Object[])` с помощью `THREAD_POOL_EXECUTOR` .

> SERIAL_EXECUTOR -> Исполнитель, выполняющий задачи по очереди в последовательном порядке.
>
> THREAD_POOL_EXECUTOR -> Исполнитель, который может использоваться для одновременного выполнения задач.

образец :

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

AsyncTask: последовательное выполнение и параллельное выполнение задачи

AsyncTask является абстрактным классом и не наследует класс `Thread` . Он имеет **абстрактный** метод `doInBackground(Params... params)` , который переопределяется для выполнения задачи. Этот метод вызывается из `AsyncTask.call()` .

Исполнитель является частью пакета `java.util.concurrent` .

Кроме того, AsyncTask содержит 2 `Executor`

`THREAD_POOL_EXECUTOR`

Он использует рабочие потоки для выполнения задач параллельно.

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

`SERIAL_EXECUTOR`

Он выполняет задачу поочередно, то есть один за другим.

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

Оба `Executor` s является **статическим,** следовательно , только один `THREAD_POOL_EXECUTOR` и один `SerialExecutor` объекты существуют, но вы можете создать несколько `AsyncTask`

объектов.

Поэтому, если вы попытаетесь выполнить несколько фоновых задач с помощью стандартного Executor (`SerialExecutor`), эта задача будет очереди и будет выполняться последовательно.

Если вы попытаетесь выполнить несколько фоновых задач с помощью `THREAD_POOL_EXECUTOR` , они будут выполняться параллельно.

Пример:

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

Выполните несколько раз кнопку «Пуск», чтобы запустить задачу и увидеть результат.

Задача Выполняется в пуле потоков (1)

Для выполнения каждой задачи требуется 1000 мс.

При t = 36s задачи 2, 3 и 4 ставятся в очередь и начинают выполняться также потому, что они выполняются параллельно.

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

Комментарий `Task Executed in thread pool` (1) и раскомментировании `Task executed Serially` (2).

Выполните несколько раз кнопку «Пуск», чтобы запустить задачу и увидеть результат.

Он выполняет задачу последовательно, поэтому каждая задача запускается после выполнения текущей задачи. Следовательно, когда выполнение задачи 1 завершается, в фоновом режиме запускается только Task 2. Наоборот.

```
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
```

Прочитайте AsyncTask онлайн: https://riptutorial.com/ru/android/topic/117/asynctask

глава 22: AudioManager

Examples

Запрос временного фокуса аудио

```
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);
        }
    }
}
```

Запрос фокуса аудио

```
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.
        }
    }
}
```

Прочитайте AudioManager онлайн: https://riptutorial.com/ru/android/topic/6798/audiomanager

глава 23: AutoCompleteTextView

замечания

Если вы хотите предлагать пользователю предложения, когда они печатают в редактируемом текстовом поле, вы можете использовать `AutoCompleteTextView` . Он автоматически предоставляет предложения, когда пользователь печатает. Список предложений отображается в выпадающем меню, из которого пользователь может выбрать один, чтобы заменить содержимое поля редактирования.

Examples

Простой, жестко закодированный AutoCompleteTextView

Дизайн (макет 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" />
```

Найти представление в коде после `setContentView()` (или его фрагмент или пользовательский эквивалент представления):

```
final AutoCompleteTextView myAutoCompleteTextView =
    (AutoCompleteTextView) findViewById(R.id.autoCompleteTextView1);
```

Предоставлять жестко кодированные данные через адаптер:

```
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);
```

Совет. Хотя предпочтительным способом было бы предоставлять данные через `Loader` какого-то типа вместо жестко закодированного списка, подобного этому.

Автозаполнение с помощью CustomAdapter, ClickListener и фильтра

Основной макет: 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>
```

Строка строки 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;
    }
}
```

Модельный класс: 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;
    }
}
```

Класс адаптера: 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();
                }
            }
        }
    };
}
```

Прочитайте AutoCompleteTextView онлайн:
https://riptutorial.com/ru/android/topic/5300/autocompletetextview

глава 24: Bluetooth и Bluetooth LE API

замечания

Bluetooth Classic доступен с Android 2.0 (API уровня 5) и выше. Bluetooth LE доступен с Android 4.3 (API уровня 18) и выше.

Examples

права доступа

Добавьте это разрешение в файл манифеста, чтобы использовать функции Bluetooth в приложении:

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

Если вам нужно инициировать обнаружение устройства или управлять настройками Bluetooth, вам также необходимо добавить это разрешение:

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

Для целевого Android API уровня 23 и выше потребуется доступ к местоположению:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<!-- OR -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

* Также см. Раздел « Разрешения» для получения более подробной информации о том, как правильно использовать разрешения.

Проверьте, включено ли Bluetooth

```
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
        }
    }
}
```

Обеспечить доступность устройства

```
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
        }
    }
}
```

Поиск поблизости устройств Bluetooth

Сначала объявите `BluetoothAdapter` .

```
BluetoothAdapter mBluetoothAdapter;
```

Теперь создайте `BroadcastReceiver` для `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());
    }
  }
};
```

Зарегистрируйте `BroadcastReceiver`

```
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);
```

Затем начните открывать соседние устройства Bluetooth, вызвав `startDiscovery`

```
mBluetoothAdapter.startDiscovery();
```

Не забудьте `onDestroy` регистрацию `BroadcastReceiver` внутри `onDestroy`

```
unregisterReceiver(mReceiver);
```

Подключение к устройству Bluetooth

После того, как вы получили BluetoothDevice, вы можете общаться с ним. Этот вид связи выполняется с использованием входных данных сокетов / выходных потоков:

Это основные шаги для установления связи Bluetooth:

1) Инициализировать разъем:

```
 private BluetoothSocket _socket;
 //...
 public InitializeSocket(BluetoothDevice device){
    try {
        _socket = device.createRfcommSocketToServiceRecord(<Your app UDID>);
    } catch (IOException e) {
        //Error
    }
  }
```

2) Подключите к гнезду:

```
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) Получение сокетов Входные / выходные потоки

```
private InputStream _inStream;
private OutputStream _outStream;
//....
try {
    _inStream = _socket.getInputStream();
    _outStream =  _socket.getOutputStream();
} catch (IOException e) {
   //Error
}
```

Входной поток - используется как входящий канал данных (прием данных с подключенного устройства)

Выходной поток - используется как исходящий канал данных (отправка данных на подключенное устройство)

По завершении третьего шага мы можем получать и отправлять данные между обоими устройствами с использованием ранее инициализированных потоков:

1) Получение данных (чтение из входного потока сокета)

```
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) Отправка данных (Запись в выходной поток)

```
public void write(byte[] bytes) {
    try {
        _outStream.write(bytes);
    } catch (IOException e) {
        //Error
    }
}
```

- Разумеется, функциональность соединения, чтения и записи должна выполняться в выделенном потоке.
- Сокеты и объекты Stream должны быть

Найдите поблизости устройства Bluetooth Low Energy

API-интерфейс BluetoothLE был представлен в API 18. Однако способ сканирования устройств изменился в API 21. Поиск устройств должен начинаться с определения идентификатора UUID службы, который должен быть отсканирован (либо официально принятые 16-разрядные UUID, либо собственные) , Этот пример иллюстрирует, как сделать независимый от API способ поиска устройств BLE:

1. Создайте модель устройства 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. Определение интерфейса сканирования Bluetooth:

```
public interface ScanningAdapter {

    void startScanning(String name, String[] uuids);
    void stopScanning();
    List<BTDevice> getFoundDeviceList();
}
```

3. Создать класс фабрики сканирования:

```
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. Создайте заводскую реализацию для каждого 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. Найдите список устройств, позвонив по телефону:

```
scanningFactory.startScanning({uuidlist});

wait few seconds...

List<BTDevice> bluetoothDeviceList = scanningFactory.getFoundDeviceList();
```

Прочитайте Bluetooth и Bluetooth LE API онлайн:
https://riptutorial.com/ru/android/topic/2462/bluetooth-и-bluetooth-le-api

глава 25: BottomNavigationView

Вступление

В течение некоторого времени представление «Нижняя навигация» находится в руководстве по разработке материалов, но нам было нелегко реализовать его в наших приложениях.

В некоторых приложениях созданы собственные решения, в то время как другие полагаются на сторонние библиотеки с открытым исходным кодом, чтобы выполнить эту работу.

Теперь библиотека поддержки дизайна увидит добавление этой нижней навигационной панели, давайте погрузиться в то, как мы можем ее использовать!

замечания

Представляет стандартную нижнюю навигационную панель для приложения. Это реализация навигации по дну материального дизайна.

Ссылки:

- Официальный Джавадок

Examples

Основная реализация

Чтобы добавить `BottomNavigationView` выполните следующие действия:

1. Добавьте в свой `build.gradle` **зависимость** :

```
compile 'com.android.support:design:25.1.0'
```

2. Добавьте `BottomNavigationView` в **свой макет** :

```
<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. Создайте **меню,** чтобы заполнить вид:

```
<!-- 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. Прикрепите **слушателя** к событиям щелчка:

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

Демо-код Checkout в BottomNavigation-Demo

Настройка BottomNavigationView

Примечание. Я предполагаю, что вы знаете, как использовать ***`BottomNavigationView`*** *.*

В этом примере я объясню, как добавить селектор для **`BottomNavigationView`** . Таким образом, вы можете указать в пользовательском интерфейсе иконки и тексты.

Создайте `bottom_navigation_view_selector.xml` как

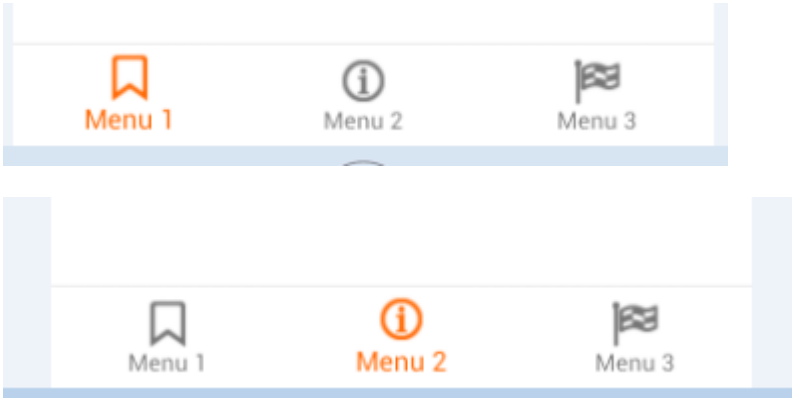
```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item  android:color="@color/bottom_nv_menu_selected" android:state_checked="true" />
    <item android:color="@color/bottom_nv_menu_default" />
</selector>
```

И используйте ниже атрибуты в **`BottomNavigationView`** в файле макета

```
app:itemIconTint="@drawable/bottom_navigation_view_selector"
app:itemTextColor="@drawable/bottom_navigation_view_selector"
```

> В приведенном выше примере я использовал тот же селектор `bottom_navigation_view_selector` для `app:itemIconTint` и `app:itemTextColor` как для сохранения цвета текста, так и значка. Но если ваш дизайн имеет разный цвет для текста и значка, вы можете определить 2 разных селектора и использовать их.

Результат будет похож на приведенный ниже



Управление включенными / отключенными состояниями

Создать селектор для включения / выключения пункта меню.

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" />
```

Разрешение более 3 меню

Этот пример является строго временным решением, поскольку в настоящее время нет способа отключить поведение, известное как ShiftMode.

Создайте функцию как таковую.

```
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);
    }
}
```

Это отключает поведение Shifting меню, когда количество элементов превышает 3 нс.

ИСПОЛЬЗОВАНИЕ

```
BottomNavigationView navView = (BottomNavigationView)
findViewById(R.id.bottom_navigation_bar);
disableMenuShiftMode(navView);
```

Proguard Issue : добавьте следующий конфигурационный файл строки proguard, а также, это не сработает.

```
-keepclassmembers class android.support.design.internal.BottomNavigationMenuView {
boolean mShiftingMode;
}
```

Кроме того, вы можете создать класс и получить доступ к этому методу. Посмотреть оригинальный ответ

ПРИМЕЧАНИЕ . Это **HOTFIX на** основе отражения, пожалуйста, обновите это, как только библиотека поддержки Google будет обновлена прямым вызовом функции.

Прочитайте BottomNavigationView онлайн:
https://riptutorial.com/ru/android/topic/7565/bottomnavigationview

глава 26: BroadcastReceiver

Вступление

BroadcastReceiver (приемник) - это компонент Android, который позволяет вам регистрироваться для системных или прикладных событий. Все зарегистрированные ресиверы для события сообщаются в среду выполнения Android после этого события.

например, широковещательная передача, объявляющая, что экран выключен, батарея разряжена или снимок сделан.

Приложения также могут инициировать трансляции, например, чтобы другие приложения знали, что некоторые данные были загружены на устройство и доступны для них.

Examples

Введение в широковещательный приемник

Приемник Broadcast - это компонент Android, который позволяет вам регистрироваться для событий системы или приложений.

Приемник может быть зарегистрирован через файл `AndroidManifest.xml` или динамически с помощью `Context.registerReceiver()` .

```
public class MyReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      //Your implementation goes here.
   }
}
```

Здесь я `ACTION_BOOT_COMPLETED` пример `ACTION_BOOT_COMPLETED` который запускается системой после завершения процесса загрузки Android.

Вы можете зарегистрировать получателя в файле манифеста следующим образом:

```
<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>
```

Теперь устройство будет загружено, будет `onReceive()` метод `onReceive()` , после чего вы сможете выполнить свою работу (например, запустить службу, запустить будильник).

Основы BroadcastReceiver

BroadcastReceivers используются для получения трансляционных намерений , которые отправляются ОС Android, другими приложениями или в одном приложении.

Каждый Intent создается с помощью *Intent Filter* , для которого требуется *действие* String. В Настройке можно настроить дополнительную информацию.

Аналогично, BroadcastReceivers регистрируются для получения намерений с определенным фильтром намерений. Они могут быть зарегистрированы программно:

```
mContext.registerReceiver(new BroadcastReceiver() {
    @Override
   public void onReceive(Context context, Intent intent) {
      //Your implementation goes here.
   }
}, new IntentFilter("Some Action"));
```

или в файле `AndroidManifest.xml` :

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="Some Action"/>
    </intent-filter>
</receiver>
```

Чтобы получить Intent, установите действие на что-то, задокументированное операционной системой Android, другим приложением или API или в вашем собственном приложении, используя `sendBroadcast` :

```
mContext.sendBroadcast(new Intent("Some Action"));
```

Кроме того, Intent может содержать информацию, такую как строки, примитивы и *Parcelables* , которые можно просмотреть в `onReceive` .

Использование LocalBroadcastManager

LocalBroadcastManager используется для отправки широковещательных Intents в приложении, не подвергая их нежелательные слушатель.

Использование *LocalBroadcastManager* более эффективно и безопаснее, чем использование `context.sendBroadcast()` напрямую, потому что вам не нужно беспокоиться о каких-либо трансляциях, подделанных другими приложениями, которые могут представлять угрозу безопасности.

Вот простой пример отправки и получения местных трансляций:

```
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);
```

Приемник Bluetooth Broadcast

добавить разрешение в файл манифеста

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

В вашем фрагменте (или активности)

- Добавить метод приемника

```
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;
    }
};
```

Регистрация трансляции

- Вызовите этот метод onResume ()

```
private void registerBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    manager.registerReceiver(mBluetoothStatusChangedReceiver, new
IntentFilter(Constants.BROADCAST_BLUETOOTH_STATE));
}
```

Отменить регистрацию трансляции

- Вызовите этот метод onPause ()

```
private void unregisterBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    // Beacon□□□
    manager.unregisterReceiver(mBluetoothStatusChangedReceiver);
}
```

Включение и отключение широковещательного приемника программно

Чтобы включить или отключить `BroadcastReceiver` , нам нужно получить ссылку на `PackageManager` и нам нужен объект `ComponentName` содержащий класс получателя, который мы хотим включить / отключить:

```
ComponentName componentName = new ComponentName(context, MyBroadcastReceiver.class);
PackageManager packageManager = context.getPackageManager();
```

Теперь мы можем вызвать следующий метод для включения `BroadcastReceiver` :

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP);
```

Или мы можем вместо этого использовать `COMPONENT_ENABLED_STATE_DISABLED` чтобы отключить приемник:

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
    PackageManager.DONT_KILL_APP);
```

BroadcastReceiver для обработки событий BOOT_COMPLETED

Пример ниже показывает, как создать `BroadcastReceiver` который может принимать события `BOOT_COMPLETED` . Таким образом, вы можете запустить `Service` или запустить `Activity` как только устройство будет включено.

Кроме того, вы можете использовать события `BOOT_COMPLETED` для восстановления ваших аварийных сигналов, так как они уничтожаются при выключении устройства.

ПРИМЕЧАНИЕ . Пользователь должен запустить приложение хотя бы один раз, прежде чем вы сможете принять действие `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
            }
        }
    }
}
```

Пример LocalBroadcastManager

BroadcastReceiver - это в основном механизм для передачи Intents через ОС для выполнения определенных действий. Классическое определение

> «Приемник Broadcast - это компонент Android, который позволяет вам регистрироваться для событий системы или приложений».

LocalBroadcastManager - это способ отправки или получения широковещательных сообщений в процессе подачи заявки. Этот механизм имеет много преимуществ

1. поскольку данные остаются внутри прикладного процесса, данные не могут быть пропущены.
2. LocalBroadcasts разрешаются быстрее, поскольку разрешение обычной широковещательной рассылки происходит во время работы по всей ОС.

Простой пример LocalBroastManager:

SenderActivity

```
Intent intent = new Intent("anEvent");
intent.putExtra("key", "This is an event");
LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
```

ReceiverActivity

1. Зарегистрировать приемник

```
LocalBroadcastManager.getInstance(this).registerReceiver(aLBReceiver,
              new IntentFilter("anEvent"));
```

2. Конкретный объект для выполнения действий при вызове получателя

```
private BroadcastReceiver aLBReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        // perform action here.
    }
};
```

3. отмените регистрацию, когда представление больше не видно.

```
@Override
protected void onPause() {
    // Unregister since the activity is about to be closed.
    LocalBroadcastManager.getInstance(this).unregisterReceiver(aLBReceiver);
    super.onDestroy();
}
```

Сообщать о двух действиях через пользовательский широковещательный приемник

Вы можете сообщить два действия, чтобы Activity A мог быть уведомлен о событии, происходящем в Activity B.

Активность 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
    }
};
```

Активность B

```
final String eventName = "your.package.goes.here.EVENT";

private void launchEvent() {
    Intent eventIntent = new Intent(eventName);
    this.sendBroadcast(eventIntent);
}
```

Конечно, вы можете добавить дополнительную информацию в эфир, добавляя дополнительные функции к намерению, которое передается между действиями. Не добавляется, чтобы привести пример как можно более простым.

Важная трансляция

Если мы используем метод sendStickyBroadcast (намерение), соответствующее намерение является липким, то есть намерение, которое вы отправляете, остается после завершения трансляции. StickyBroadcast, как следует из названия, является механизмом для чтения данных из трансляции после завершения трансляции. Это можно использовать в сценарии, в котором вы можете проверить `Activity's onCreate()` значение ключа в намерении до того, как эта активность была запущена.

```
Intent intent = new Intent("com.org.action");
intent.putExtra("anIntegerKey", 0);
sendStickyBroadcast(intent);
```

Использование упорядоченных трансляций

Заказываемые трансляции используются, когда вам нужно указать приоритет для

широковещательных слушателей.

В этом примере `firstReceiver` будет получать широковещательную передачу всегда до `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);
```

Кроме того, широковещательный приемник может прервать упорядоченную передачу:

```
@Override
public void onReceive(final Context context, final Intent intent) {
    abortBroadcast();
}
```

в этом случае все приемники с более низким приоритетом не получат широковещательное сообщение.

Android остановил состояние

Начиная с Android 3.1, все приложения после установки находятся в остановленном состоянии. При остановленном состоянии приложение не будет запускаться по какой-либо причине, кроме как при ручном запуске какого-либо действия, или в **явном** намерении, которое адресовано активности, службе или широковещательной передаче.

При написании системного приложения, которое напрямую устанавливает APK, учтите, что недавно установленный APP не получит никаких трансляций, пока не перейдет в состояние без остановок.

Легкий способ активации приложения - отправить явное сообщение в это приложение. поскольку большинство приложений реализуют `INSTALL_REFERRER` , мы можем использовать его как точку подключения

Сканируйте манифест установленного приложения и отправьте явное вещание каждому получателю:

```
Intent intent = new Intent();
intent.addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES);
intent.setComponent(new ComponentName(packageName, fullClassName));
sendBroadcast(intent);
```

Прочитайте BroadcastReceiver онлайн:
https://riptutorial.com/ru/android/topic/1460/broadcastreceiver

глава 27: CardView

Вступление

FrameLayout с закругленным фоном угла и тенью.

CardView использует свойство высоты на Lollipop для теней и возвращается к пользовательской эмулируемой теневой реализации на старых платформах.

Из-за дороговизны скругления округлых углов на платформах перед Lollipop CardView не обрезает своих детей, которые пересекаются с закругленными углами. Вместо этого он добавляет отступы, чтобы избежать такого пересечения (см. SetPreventCornerOverlap (boolean), чтобы изменить это поведение).

параметры

параметр	подробности
cardBackgroundColor	Цвет фона для CardView.
cardCornerRadius	Угловой радиус для CardView.
cardElevation	Высота для CardView.
cardMaxElevation	Максимальная высота для CardView.
cardPreventCornerOverlap	Добавьте дополнение к CardView на v20 и ранее, чтобы предотвратить пересечения между содержимым карты и закругленными углами.
cardUseCompatPadding	Добавьте дополнение в API v21 +, чтобы иметь те же самые измерения с предыдущими версиями. Может быть логическим значением, таким как «true» или «false».
contentPadding	Внутренняя прокладка между краями Карты и дочерними элементами CardView.
contentPaddingBottom	Внутренняя прокладка между нижним краем Карты и дочерними элементами CardView.
contentPaddingLeft	Внутренняя прокладка между левым краем Карты и дочерними элементами CardView.
contentPaddingRight	Высота для CardView.

параметр	подробности
cardElevation	Внутренняя прокладка между правым краем Карты и дочерними элементами CardView.
contentPaddingTop	Внутренняя прокладка между верхним краем Карты и дочерними элементами CardView.

замечания

`CardView` использует реальную высоту и динамические тени на Lollipop (API 21) и выше. Однако, прежде чем Lollipop `CardView` вернется к реализации программной тени.

Если вы пытаетесь сделать `ImageView` `CardView` в округленные углы `CardView` , вы можете заметить, что он не выглядит корректным перед Lollipop (API 21). Чтобы исправить это, вы должны вызвать `setPreventCornerOverlap(false)` в `CardView` или добавить `app:cardPreventCornerOverlap="false"` в ваш макет.

Перед использованием `CardView` вам необходимо добавить зависимость библиотеки поддержки в файле `build.gradle` :

```
dependencies{
    compile 'com.android.support:cardview-v7:25.2.0'
}
```

Номер последней версии можно найти здесь

Официальная документация:

https://developer.android.com/reference/android/support/v7/widget/CardView.html
https://developer.android.com/training/material/lists-cards.html

Examples

Начало работы с CardView

`CardView` является членом библиотеки поддержки Android и предоставляет макет для карт.

Чтобы добавить `CardView` в свой проект, добавьте следующую строку в ваши зависимости `build.gradle` .

```
compile 'com.android.support:cardview-v7:25.1.1'
```

Номер последней версии можно найти здесь

В своем макете вы можете добавить следующее, чтобы получить карту.

```
<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>
```

Затем вы можете добавить другие макеты внутри, и они будут включены в карту.

Кроме того, CardView может быть заполнен любым элементом пользовательского интерфейса и управляться из кода .

```
<?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>
```

Настройка CardView

CardView обеспечивает высоту по умолчанию и угловой радиус, так что карты имеют согласованный внешний вид на платформах.

Вы можете настроить эти значения по умолчанию, используя эти атрибуты в файле xml:

1. `card_view:cardElevation` атрибут `card_view:cardElevation` добавляет отметку в CardView.
2. `card_view:cardBackgroundColor` атрибут `card_view:cardBackgroundColor` используется для настройки цвета фона фона CardView (вы можете дать любой цвет).
3. `card_view:cardCornerRadius` атрибут `card_view:cardCornerRadius` используется для кривой 4 ребер CardView
4. `card_view:contentPadding` атрибут `card_view:contentPadding` добавляет отступы между карточкой и `card_view:contentPadding` элементами карты

Примечание. Card_view - это пространство имен, определенное в верхнем представлении макета родителя. xmlns: card_view = " http://schemas.android.com/apk/res-auto "

Вот пример:

```
<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>
```

Вы также можете делать это программно, используя:

```
card.setCardBackgroundColor(....);
card.setCardElevation(...);
card.setRadius(....);
card.setContentPadding();
```

Проверьте официальный javadoc на наличие дополнительных свойств.

Добавление анимации Ripple

Чтобы включить анимацию пульсации в CardView, добавьте следующие атрибуты:

```
<android.support.v7.widget.CardView
  ...
  android:clickable="true"
  android:foreground="?android:attr/selectableItemBackground">
  ...
</android.support.v7.widget.CardView>
```

Использование изображений в качестве фона в CardView (проблемы с устройством перед Lollipop)

При использовании Image / Color в качестве фона в CardView вы можете получить небольшие белые прокладки (если по умолчанию цвет карты белый) по краям. Это происходит из-за закругленных углов по умолчанию в окне карты. Вот как избежать этих полей в устройствах с предварительным лечением.

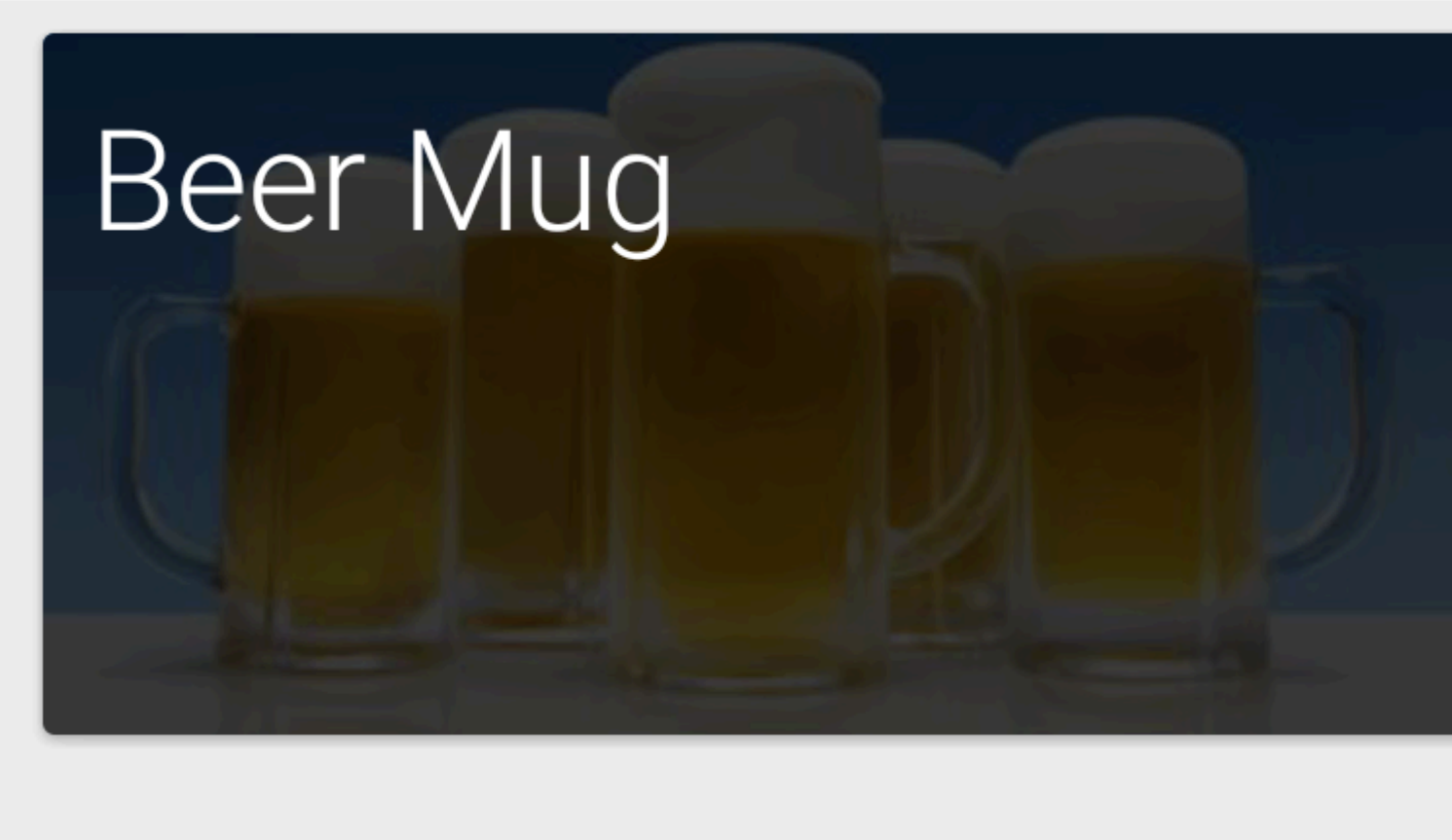
Нам нужно использовать атрибут `card_view:cardPreventCornerOverlap="false"` в CardView. 1). В XML используйте следующий фрагмент.

```
<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>
```

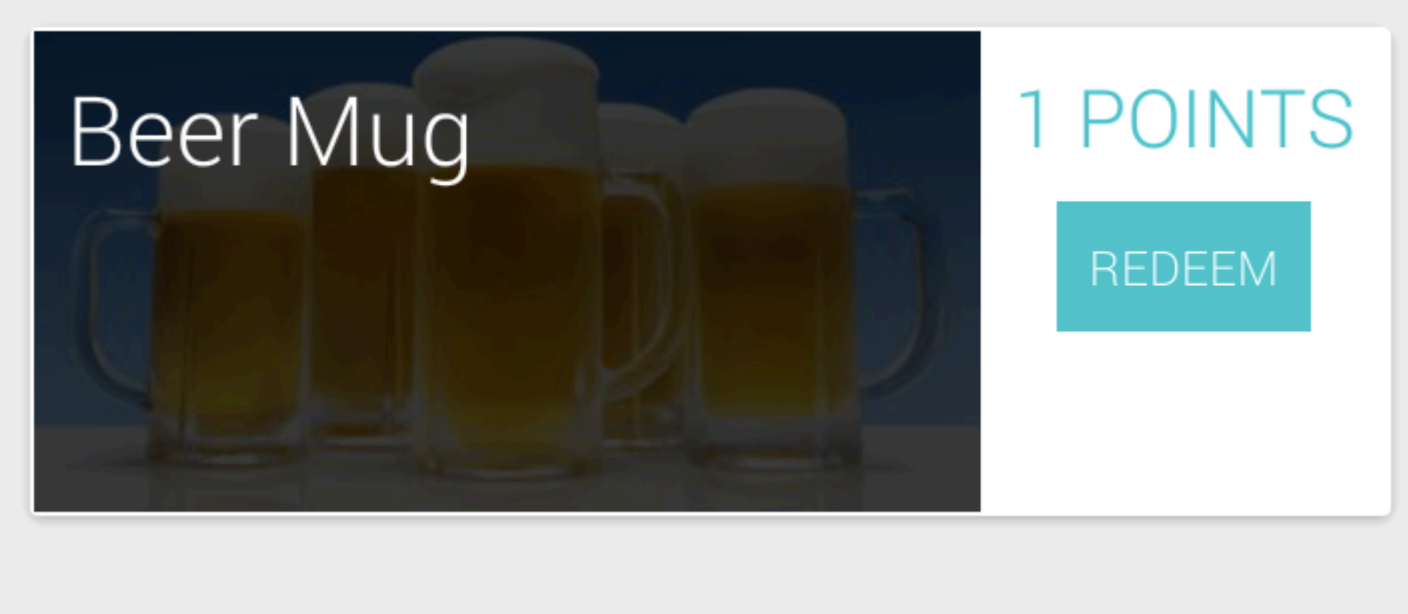
2. В Java, как этот `cardView.setPreventCornerOverlap(false)` .

При этом удаляется нежелательное дополнение на краях карты. Вот некоторые визуальные примеры, связанные с этой реализацией.

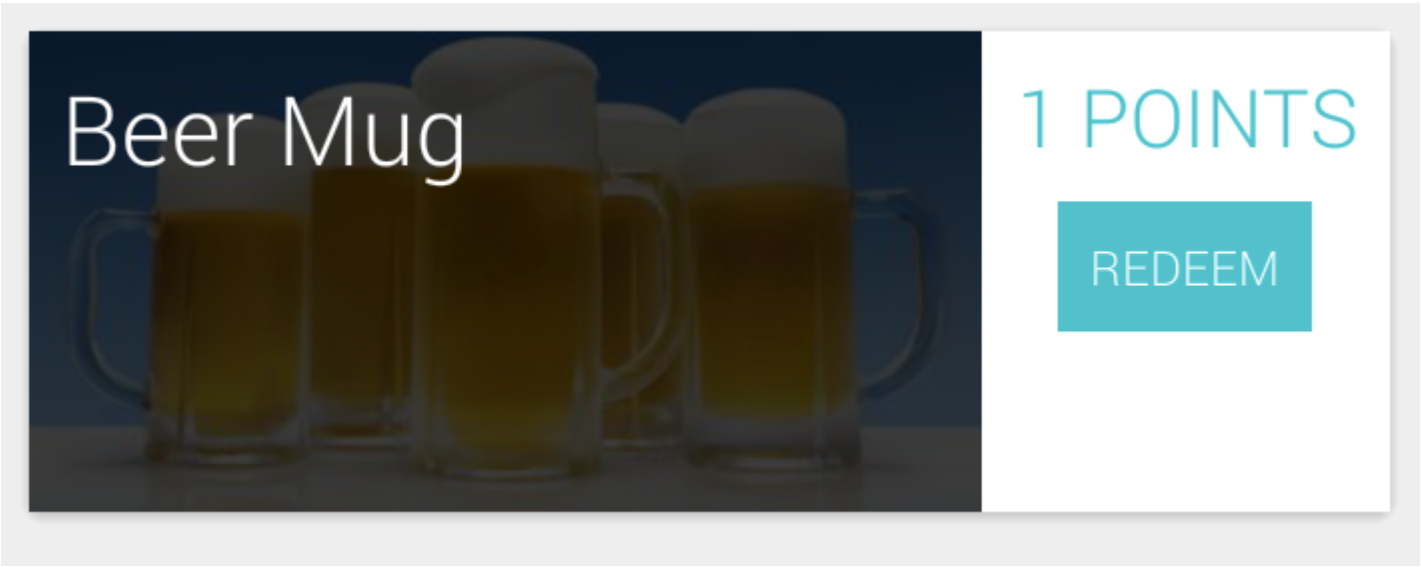
1 карта с фоном изображения в API 21 (отлично)



2 Карточка с фоном изображения в API 19 без атрибута (обратите внимание на прокладки вокруг изображения)



3 FIXED Card с фоном изображения в API 19 с атрибутом
`cardView.setPreventCornerOverlap(false)` (проблема теперь исправлена)



Также читайте об этом на Документации здесь
Оригинальный пост SOF здесь

Цвет фона Animate CardView с 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);
}
```

Прочитайте CardView онлайн: https://riptutorial.com/ru/android/topic/726/cardview

глава 28: CleverTap

Вступление

Быстрые хаки для аналитики и взаимодействия SDK, предоставляемые CleverTap - Android

замечания

Получите свои учетные данные CleverTap с https://clevertap.com .

Examples

Получить экземпляр SDK для записи событий

```
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
}
```

Установка уровня отладки

В своем пользовательском классе приложения переопределите метод `onCreate()` , добавьте `onCreate()` строку:

```
CleverTapAPI.setDebugLevel(1);
```

Прочитайте CleverTap онлайн: https://riptutorial.com/ru/android/topic/9337/clevertap

глава 29: ConstraintLayout

Вступление

`ConstraintLayout` - это `ViewGroup` которая позволяет `ViewGroup` и `ViewGroup` размеры виджетов. Он совместим с Android 2.3 (API уровня 9) и выше.

Он позволяет создавать большие и сложные макеты с иерархией с плоским представлением. Он похож на `RelativeLayout` на то, что все представления выложены в соответствии с отношениями между представлениями сестры и родительским макетом, но он более гибкий, чем `RelativeLayout` и более простой в использовании с редактором макетов Android Studio.

Синтаксис

- **ConstraintLayout**
    - public void addView (просмотр дочернего объекта, индекс int, параметры ViewGroup.LayoutParams)
    - public ConstraintLayout.LayoutParams generateLayoutParams (AttributeSet attrs)
    - public void onViewAdded (View view)
    - public void onViewRemoved (View view)
    - public void removeView (Просмотреть представление)
    - public void requestLayout ()
    - protected boolean checkLayoutParams (параметры ViewGroup.LayoutParams)
    - protected ConstraintLayout.LayoutParams generateDefaultLayoutParams ()
    - protected ViewGroup.LayoutParams generateLayoutParams (параметры ViewGroup.LayoutParams)
    - protected void onLayout (boolean changed, 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)

параметры

параметр	подробности
ребенок	`View` который будет добавлен в макет
индекс	Индекс `View` в иерархии макета
Титулы	`LayoutParams` `View`
ATTRS	`AttributeSet` , который определяет `LayoutParams`
Посмотреть	`View` , которое было добавлено или удалено
изменено	Указывает, изменил ли этот `View` размер или положение
оставил	Левое положение относительно родительского `View`
Топ	Верхнее положение относительно родительского `View`
право	Правильное положение относительно родительского `View`
низ	Нижняя позиция относительно родительского `View`
widthMeasureSpec	Требования к горизонтальному пространству, предъявляемые родительским `View`
heightMeasureSpec	Требования к вертикальному пространству, предъявляемые родительским `View`
LayoutDirection	-
	-
widthAttr	-
heightAttr	-

замечания

В Google IO 2016 Google анонсировала новый план Android под названием ConstraintLayout. Обратите внимание, потому что в настоящее время этот макет является **бета-релизом** .

Больше о шаблоне ограничения:

https://codelabs.developers.google.com/codelabs/constraint-layout/index.html

Examples

Добавление ConstraintLayout к вашему проекту

Для работы с ConstraintLayout вам потребуется Android Studio версии 2.2 или новее и иметь как минимум 32 (или выше) поддержки Android Support Repository.

1. Добавьте библиотеку Constraint Layout в качестве зависимости в файле `build.gradle` :

```
dependencies {
   compile 'com.android.support.constraint:constraint-layout:1.0.2'
}
```

2. Проект синхронизации

Чтобы добавить новый макет ограничения в свой проект:

1. **Щелкните правой кнопкой мыши** каталог макета модуля и выберите « `New > XML > Layout XML.`
2. Введите **имя** макета и введите `"android.support.constraint.ConstraintLayout"` для корневого тега.
3. Нажмите « **Готово**» .

В противном случае просто добавьте в файл макета:

```
<?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>
```

Цепи

Поскольку `ConstraintLayout` alpha 9, доступны **цепочки** . **Цепь** представляет собой набор представлений внутри `ConstraintLayout` , которые связаны между собой двунаправленно, т. Е. **A,** связанный с **B** с ограничением, и **B,** связанный с **A** с другим ограничением.

Пример:

```
<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>
```

В этом примере два вида расположены один под другим, и оба они центрированы по вертикали. Вы можете изменить вертикальное положение этих видов, отрегулировав **смещение** цепи. Добавьте следующий код в первый элемент цепочки:

```
app:layout_constraintVertical_bias="0.2"
```

В вертикальной цепочке первый элемент представляет собой самый верхний вид, а в горизонтальной цепочке это самый левый вид. Первый элемент определяет поведение всей цепочки.

Цепи - это новая функция и часто обновляются. Вот официальная Android-документация по цепочкам.

Прочитайте ConstraintLayout онлайн:
https://riptutorial.com/ru/android/topic/5076/constraintlayout

глава 30: ConstraintSet

Вступление

Этот класс позволяет вам программно определить набор ограничений, которые будут использоваться с `ConstraintLayout` . Он позволяет создавать и сохранять ограничения и применять их к существующему `ConstraintLayout` .

Examples

ConstraintSet с программным обеспечением ContraintLayout

```
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
        }
    }
}
```

Прочитайте ConstraintSet онлайн: https://riptutorial.com/ru/android/topic/9334/constraintset

глава 31: ExoPlayer

Examples

Добавить ExoPlayer в проект

Через jCenter

включая следующее в файле build.gradle вашего проекта:

```
compile 'com.google.android.exoplayer:exoplayer:rX.X.X'
```

где rX.XX - ваша предпочтительная версия. Последнюю версию см. В выпусках проекта. Для получения дополнительной информации см. Проект на Bintray .

Использование ExoPlayer

Создайте экземпляр своего ExoPlayer:

```
exoPlayer = ExoPlayer.Factory.newInstance(RENDERER_COUNT, minBufferMs, minRebufferMs);
```

Чтобы воспроизводить только аудио, вы можете использовать следующие значения:

```
RENDERER_COUNT = 1 //since you want to render simple audio
minBufferMs = 1000
minRebufferMs = 5000
```

Оба значения буфера могут быть изменены в соответствии с вашими требованиями.

Теперь вам нужно создать DataSource. Когда вы хотите потоковое mp3, вы можете использовать DefaultUriDataSource. Вы должны передать Контекст и UserAgent. Чтобы упростить воспроизведение локального файла и передать null в качестве userAgent:

```
DataSource dataSource = new DefaultUriDataSource(context, null);
```

Затем создайте sampleSource:

```
ExtractorSampleSource sampleSource = new ExtractorSampleSource(
                uri, dataSource, new Mp3Extractor(), RENDERER_COUNT, requestedBufferSize);
```

uri указывает на ваш файл, в качестве Extractor вы можете использовать простой Mp3Extractor по умолчанию, если хотите воспроизвести mp3. requestBufferSize может быть изменен в соответствии с вашими требованиями. Например, используйте 5000.

Теперь вы можете создать свой рендерер звуковой дорожки, используя источник примера, следующим образом:

```
MediaCodecAudioTrackRenderer audioRenderer = new MediaCodecAudioTrackRenderer(sampleSource);
```

Наконец, позвоните в свой экземпляр exoPlayer:

```
exoPlayer.prepare(audioRenderer);
```

Для запуска воспроизведения:

```
exoPlayer.setPlayWhenReady(true);
```

Основные шаги для воспроизведения видео и аудио с использованием стандартных реализаций 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!
```

Прочитайте ExoPlayer онлайн: https://riptutorial.com/ru/android/topic/6248/exoplayer

глава 32: Facebook SDK для Android

Синтаксис

- **newInstance** : создать отдельный экземпляр класса помощника Facebook.
- **loginUser** : Войти в систему.
- **signOut** : выйти из системы.
- **getCallbackManager** : для получения обратного вызова для Facebook.
- **getLoginCallback** : для получения обратного вызова для входа.
- **getKeyHash** : генерировать Facebook-хеш.

параметры

параметр	подробности
ТЕГ	Строка, используемая при регистрации
FacebookSignInHelper	Статическая ссылка на помощника facebook
CallbackManager	Обратный вызов для операций с facebook
Деятельность	Контекст
PERMISSION_LOGIN	Массив, содержащий все разрешения, необходимые для входа в систему для входа в facebook.
loginCallback	Обратный вызов для входа в facebook

Examples

Как добавить Facebook Войти в Android

Добавьте ниже зависимости от вашего `build.gradle`

```
  // Facebook login
    compile 'com.facebook.android:facebook-android-sdk:4.21.1'
```

Добавьте ниже вспомогательный класс в ваш служебный пакет:

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

Добавьте в свою команду код ниже:

```
FacebookSignInHelper facebookSignInHelper =
FacebookSignInHelper.newInstance(LoginActivity.this, fireBaseAuthHelper);
facebookSignInHelper.loginUser();
```

Добавьте ниже код в свой `OnActivityResult` :

```
 facebookSignInHelper.getCallbackManager().onActivityResult(requestCode, resultCode, data);
```

Настройка разрешений для доступа к данным из профиля Facebook

Если вы хотите получить информацию о профиле пользователя в Facebook, вам

необходимо установить разрешения для него:

```
loginButton = (LoginButton)findViewById(R.id.login_button);

loginButton.setReadPermissions(Arrays.asList("email", "user_about_me"));
```

Вы можете добавлять дополнительные разрешения, такие как список друзей, сообщения, фотографии и т. Д. Просто выберите правильное разрешение и добавьте его в приведенный выше список.

Примечание. Вам не нужно устанавливать какие-либо явные разрешения для доступа к общедоступному профилю (имя, фамилия, идентификатор, пол и т. Д.).

Создайте свою собственную кнопку для входа в Facebook

После того, как вы впервые добавите логин / регистрацию в Facebook, кнопка выглядит примерно так:

В большинстве случаев это не соответствует дизайнерским спецификациям вашего приложения. И вот как вы можете настроить его:

```
<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>
```

Просто заверните оригинальный `com.facebook.login.widget.LoginButton` в `FrameLayout` и `FrameLayout` его видимость.

Затем добавьте свою настраиваемую кнопку в тот же `FrameLayout` . Я добавил некоторые образцы спецификаций. Вы всегда можете сделать свой собственный рисованный фон для кнопки facebook и установить его в качестве фона кнопки.

Последнее, что мы делаем, - это просто щелкнуть мышью по моей пользовательской кнопке, нажав на кнопку 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();
    }
}
```

Большой! Теперь кнопка выглядит примерно так:

Минималистическое руководство по внедрению Facebook / регистрации

1. Вы должны настроить предварительные условия .
2. Добавьте активность Facebook в файл *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. Добавьте кнопку входа в свой XML-файл макета:

   ```
   <com.facebook.login.widget.LoginButton
       android:id="@+id/login_button"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content" />
   ```

4. Теперь у вас есть кнопка Facebook. Если пользователь нажмет на него, окно входа в систему Facebook появится поверх экрана приложения. Здесь пользователь может заполнить свои учетные данные и нажать кнопку « *Вход* » . Если учетные данные верны, диалог предоставляет соответствующие разрешения, и обратный вызов отправляется в исходное действие, содержащее кнопку. Следующий код показывает, как вы можете получить этот обратный вызов:

```
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.
    }
});
```

Выход из Facebook

Facebook SDK 4.0 и далее, вот как мы выходим из системы:

```
com.facebook.login.LoginManager.getInstance().logOut();
```

Для версий до версии 4.0 выход из системы пропадает, явно очистив токен доступа:

```
Session session = Session.getActiveSession();
session.closeAndClearTokenInformation();
```

Прочитайте Facebook SDK для Android онлайн:
https://riptutorial.com/ru/android/topic/3919/facebook-sdk-для-android

глава 33: Fastjson

Вступление

Fastjson - это библиотека Java, которая может быть использована для преобразования объектов Java в их представление JSON. Его также можно использовать для преобразования строки JSON в эквивалентный объект Java.

Особенности Fastjson:

Обеспечивайте максимальную производительность на стороне сервера и андроид-клиента

Предоставьте простые `toJSONString()` и `parseObject()` для преобразования объектов Java в JSON и наоборот

Разрешить преобразование ранее немодифицируемых объектов в JSON и обратно

Обширная поддержка Java Generics

Синтаксис

- Разбор объекта (текст строки)
- JSONObject parseObject (текст строки)
- T parseObject (текст строки, класс <T> clazz)
- JSONArray parseArray (текст строки)
- <T> Список <T> parseArray (текст строки, класс <T> clazz)
- String toJSONString (объект объекта)
- String toJSONString (объект Object, boolean prettyFormat)
- Объект toJSON (объект javaObject)

Examples

Разбор JSON с Fastjson

Вы можете посмотреть пример в библиотеке Fastjson

шифровать

```
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);
```

Выход

```
{"id":0,"name":"admin","users":[{"id":2,"name":"guest"},{"id":3,"name":"root"}]}
```

раскодировать

```
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;
    }
}
```

Преобразование данных типа Map в строку JSON

Код

```
    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);
```

Выход

```
{1:"No.1",2:"No.2",3:[{"id":2,"name":"Liu"},{"id":3,"name":"Yue"}]}
```

Прочитайте Fastjson онлайн: https://riptutorial.com/ru/android/topic/10865/fastjson

глава 34: Fastlane

замечания

Fastlane - это инструмент для разработчиков iOS, Mac и Android для автоматизации утомительных задач, таких как создание скриншотов, настройка профилей подготовки и выпуск приложения.

Документы: https://docs.fastlane.tools/

Исходный код: https://github.com/fastlane/fastlane

Examples

Fastfile для создания и загрузки нескольких вариантов для бета-версии Crashlytics

Это пример настройки **Fastfile** для приложения с несколькими **вкусами** . Это дает вам возможность создавать и внедрять все вкусы или один аромат. После развертывания он сообщает **Slack** о статусе развертывания и отправляет уведомление тестировщикам в бета-тестировании группой тестировщиков Crashlytics.

Чтобы создавать и развертывать все варианты, используйте:

```
fastlane android beta
```

Для создания одного APK и развертывания используйте:

```
fastlane android beta app:flavorName
```

Используя один файл Fastlane, вы можете управлять приложениями iOS, Android и Mac. Если вы используете этот файл только для одной `platform` приложения, не требуется.

Как это устроено

1. аргумент `android` говорит fastlane, что мы будем использовать `:android` платформу `:android` .
2. Внутри платформы `:android` вы можете иметь несколько полос движения. В настоящее время у меня есть только `:beta` лейн. Второй аргумент из приведенной выше команды указывает полосу, которую мы хотим использовать.
3. `options[:app]`
4. Есть две задачи **Gradle** . Во-первых, он работает с `gradle clean` . Если вы предоставили аромат с ключом `app` , fastfile запускает `gradle assembleReleaseFlavor` . В

противном случае он запускает `gradle assembleRelease` для создания всех `gradle assembleRelease` .

5. Если мы строим для всех ароматов, массив сгенерированных имен файлов APK хранится внутри `SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS` . Мы используем это для циклического создания сгенерированных файлов и развертывания их в **Beta с помощью Crashlytics** . поля `notifications` и `groups` являются необязательными. Они используются для уведомления тестеров, зарегистрированных для приложения на **бета-версии Crashlytics** .
6. Если вы знакомы с Crashlytics, вы можете знать, что для активации приложения на портале вы должны запустить его на устройстве и использовать его в первую очередь. В противном случае Crashlytics будет считать приложение неактивным и выдает ошибку. В этом случае я фиксирую его и сообщаю **Slack** как провал, поэтому вы узнаете, какое приложение неактивно.
7. Если развертывание будет успешным, **fastlane** отправит сообщение успеха **Slack** .
8. `#{/([^\/]*)$/.match(apk)}` это регулярное выражение используется для получения имени аромата из пути APK. Вы можете удалить его, если он не работает для вас.
9. `get_version_name` и `get_version_code` - два плагина **Fastlane** для получения имени и кода версии приложения. Вы должны установить эти драгоценные камни, если хотите, или их можно удалить. Подробнее о плагинах здесь.
10. Оператор `else` будет выполнен, если вы создаете и развертываете один APK. Нам не нужно предоставлять `apk_path` для Crashlytics, так как у нас есть только одно приложение.
11. `error do` блок в конце используется для получения уведомления, если что-то еще не работает во время выполнения.

Заметка

Не забудьте заменить `SLACK_URL` , `API_TOKEN` , `GROUP_NAME` и `BUILD_SECRET` своими собственными учетными данными.

```
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 для создания и установки всех вариантов для данного типа сборки на устройство

Добавьте эту полосу в свой **Fastfile** и запустите `fastlane installAll type:{BUILD_TYPE}` в командной строке. Замените `BUILD_TYPE` на тип сборки, который вы хотите построить.

Например: `fastlane installAll type:Debug`

Эта команда построит все варианты данного типа и установит его на ваше устройство. В настоящее время он не работает, если у вас установлено более одного устройства. Убедитесь, что у вас есть только один. В будущем я планирую добавить опцию для выбора целевого устройства.

```
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
```

Прочитайте Fastlane онлайн: https://riptutorial.com/ru/android/topic/8215/fastlane

глава 35: FileIO с Android

Вступление

Чтение и запись файлов в Android не отличается от чтения и записи файлов в стандартной Java. Можно использовать тот же пакет `java.io` Тем не менее, есть некоторые особенности, связанные с папками, где вам разрешено писать, разрешения вообще и работа MTP.

замечания

Android предоставляет средства для обмена файлами между несколькими приложениями, как это описано здесь . Это не требуется, если есть только одно приложение, которое создает и использует файл.

Android предоставляет альтернативные варианты хранения, такие как общие и частные настройки, сохраненные пакеты и встроенная база данных. В некоторых случаях это лучший выбор, чем просто использование простых файлов.

Активность Android имеет несколько конкретных методов, которые выглядят как замена стандартных методов ввода файлов в формате Java. Например, вместо этого для `File.delete()` вы можете вызывать `Context.deleteFile()` , и вместо того, чтобы применять `File.listFiles()` рекурсивно, вы можете вызвать `Context.fileList()` чтобы получить список всех ваших конкретных файлов приложения несколько код. Однако они не обеспечивают дополнительную функциональность, кроме стандартного пакета `java.io`

Examples

Получение рабочей папки

Вы можете получить свою рабочую папку, вызвав метод `getFilesDir()` в своей деятельности (Activity - это основной класс вашего приложения, который наследуется от Context. См. Здесь). Чтение не отличается. Доступ к этой папке будет иметь только ваше приложение.

Ваша деятельность может содержать следующий код, например:

```
File myFolder = getFilesDir();
File myFile = new File(myFolder, "myData.bin");
```

Написание исходного массива байтов

```
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()
```

В этом коде нет ничего особенного для Android. Если вы пишете много небольших значений, используйте BufferedOutputStream, чтобы уменьшить износ внутреннего SSD устройства.

Сериализация объекта

Старая хорошая сериализация объектов Java доступна для вас в Android. вы можете определить классы Serializable, например:

```
class Cirle implements Serializable {
  final int radius;
  final String name;

  Circle(int radius, int name) {
    this.radius = radius;
    this.name = name;
  }
}
```

а затем записать в 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()
```

Сериализация объектов Java может быть либо идеальной, либо действительно плохим выбором, в зависимости от того, что вы хотите с ней сделать - вне рамок этого учебника и иногда основывается на мнениях. Прежде всего, ознакомьтесь с версией, если вы решите ее использовать.

Запись на внешнее хранилище (SD-карта)

Вы также можете читать и записывать с / на карту памяти (SD-карту), которая присутствует на многих устройствах Android. Доступ к файлам в этом месте могут получить другие программы, а также непосредственно пользователь после подключения устройства к ПК через USB-кабель и включения протокола MTP.

Поиск местоположения SD-карты несколько более проблематичен. Класс Environment содержит статические методы для получения «внешних каталогов», которые обычно

должны находиться внутри SD-карты, а также информация, если SD-карта существует вообще и доступна для записи. Этот вопрос содержит ценные ответы, как убедиться, что правильное местоположение будет найдено.

Доступ к внешнему хранилищу требует разрешений в вашем проявлении Android:

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

Для более старых версий разрешения на размещение Android достаточно установить эти разрешения в манифест (пользователь должен утвердить во время установки). Однако начиная с Android 6.0 Android запрашивает у пользователя одобрение во время первого доступа, и вы должны поддержать этот новый подход. В противном случае доступ запрещается независимо от вашего манифеста.

В Android 6.0 сначала нужно проверить разрешение, а затем, если не предоставлено, запросите его. Примеры кода можно найти внутри этого вопроса SO .

Решение проблемы «Невидимые файлы MTP».

Если вы создаете файлы для экспорта через USB-кабель на рабочий стол с использованием протокола MTP, может возникнуть проблема, что вновь созданные файлы не будут сразу видны в проводнике файлов, работающем на подключенном настольном ПК. Чтобы сделать новые файлы видимыми, вам необходимо вызвать 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)));
```

Этот код вызова MediaScannerConnection работает только для файлов, а не для каталогов. Проблема описана в этом отчете об ошибке Android . Это может быть исправлено для некоторых версий в будущем или на некоторых устройствах.

Работа с большими файлами

Маленькие файлы обрабатываются за долю секунды, и вы можете читать / записывать их вместо кода, в котором это необходимо. Однако, если файл больше или медленнее обрабатывается, вам может потребоваться использовать AsyncTask в Android для работы с файлом в фоновом режиме:

```
    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
        }
    }
}
```

а потом

```
new FileOperation().execute("Some parameters");
```

Этот вопрос SO содержит полный пример создания и вызова AsyncTask. Также см. Вопрос об обработке ошибок о том, как обрабатывать IOExceptions и другие ошибки.

Прочитайте FileIO с Android онлайн: https://riptutorial.com/ru/android/topic/8689/fileio-c-android

глава 36: FileProvider

Examples

Совместное использование файла

В этом примере вы узнаете, как делиться файлом с другими приложениями. Мы будем использовать pdf-файл в этом примере, хотя код работает и с любым другим форматом.

Дорожная карта:

Укажите каталоги, в которых размещаются файлы, которые вы хотите разделить.

Для обмена файлами мы будем использовать FileProvider, класс, обеспечивающий безопасное совместное использование файлов между приложениями. FileProvider может делиться только файлами в предопределенных каталогах, поэтому давайте их определим.

1. Создайте новый XML-файл, который будет содержать пути, например *res / xml / filepaths.xml*

2. Добавить пути

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <files-path name="pdf_folder" path="documents/"/>
</paths>
```

Определите FileProvider и свяжите его с файловыми путями

Это делается в манифесте:

```
<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>
```

Создайте URI для файла

Чтобы поделиться файлом, мы должны предоставить идентификатор файла. Это делается с помощью 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);
```

Как вы можете видеть в коде, мы сначала создаем новый класс File, представляющий файл. Чтобы получить URI, мы попросим FileProvider получить нас. Второй аргумент важен: он передает полномочия FileProvider. Он должен быть равен авторитету FileProvider, определенному в манифесте.

Поделитесь файлом с другими приложениями

Мы используем ShareCompat для совместного использования файла с другими приложениями:

```
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);
```

Выбор - это меню, из которого пользователь может выбрать, с каким приложением он хочет поделиться файлом. Флаг Intent.FLAG_GRANT_READ_URI_PERMISSION необходим

для предоставления разрешения на временное чтение для URI.

Прочитайте FileProvider онлайн: https://riptutorial.com/ru/android/topic/6266/fileprovider

глава 37: Firebase

Вступление

Firebase - это платформа для мобильных и веб-приложений с инструментами и инфраструктурой, призванная помочь разработчикам создавать высококачественные приложения.

Характеристики

Firebase Cloud Messaging, Firebase Auth, база данных реального времени, хранилище Firebase, хостинг Firebase, лаборатория тестирования Firebase для Android, отчеты о сбоях Firebase.

замечания

Firebase - Расширенная документация:

Существует еще один тег, где вы можете найти другие темы и примеры использования Firebase.

Другие связанные темы:

- База данных Firebase Realtime
- Инфраструктура приложений Firebase
- Пожарная авария
- Пожарная безопасность Firebase

Examples

Создание пользователя 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;
    }
}
```

Войти Пользователь Firebase с электронной почтой и паролем

```
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));
    }
}
```

Отправить пароль для сброса пароля 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();
                        }
                    }
                });
    }
}
```

Обновление электронной почты пользователей 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());
    }
}
```

Изменить пароль

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

Повторно аутентифицировать пользователя 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();
    }
}
```

Операции хранения в Firebase

В этом примере вы сможете выполнить следующие операции:

1. Подключение к хранилищу Firebase
2. Создайте каталог с именем "images"
3. Загрузите файл в каталог изображений
4. Загрузите файл из каталога изображений
5. Удаление файла из каталога изображений

```
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();
    }
}
```

По умолчанию в правилах хранения Firebase применяется ограничение на аутентификацию. Если пользователь аутентифицирован, только тогда он может выполнять операции с Firebase Storage, иначе он не сможет. Я отключил часть аутентификации в этой демонстрации, обновив правила хранения. Раньше правила выглядели так:

```
 service firebase.storage {
   match /b/**something**.appspot.com/o {
     match /{allPaths=**} {
```

```
      allow read, write: if request.auth != null;
    }
  }
}
```

Но я отказался пропустить аутентификацию:

```
service firebase.storage {
  match /b/**something**.appspot.com/o {
    match /{allPaths=**} {
      allow read, write;
    }
  }
}
```

Пожарная безопасность Firebase

Прежде всего, вам нужно настроить проект, добавив Firebase в свой Android-проект, следуя шагам, описанным в этом разделе .

Настройка Firebase и FCM SDK

Добавьте зависимость FCM к файлу `build.gradle` уровне `build.gradle`

```
dependencies {
 compile 'com.google.firebase:firebase-messaging:11.0.4'
}
```

И в самом низу (это важно) добавьте:

```
// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

Отредактируйте манифест приложения

Добавьте в манифест вашего приложения следующее:

- Служба, которая расширяет `FirebaseMessagingService` . Это необходимо, если вы хотите выполнять обработку сообщений за пределами приема уведомлений в приложениях в фоновом режиме.
- Служба, которая расширяет `FirebaseInstanceIdService` для обработки создания, вращения и обновления регистрационных токенов.

Например:

```
    <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>
```

Вот простые реализации двух служб.

Чтобы получить текущий токен регистрации, расширьте класс `FirebaseInstanceIdService` и переопределите метод `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
    }
}
```

Чтобы получать сообщения, используйте службу, которая расширяет `FirebaseMessagingService` и переопределяет метод `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
    }
```

в **Firebase** может сгруппировать пользователя по своему поведению, например, « AppVersion», «бесплатный пользователь», «купить пользователя» или какие-либо конкретные правила », а затем отправить уведомление определенной группе, отправив функцию **темы** в fireBase.
зарегистрировать пользователя в теме

```
FirebaseMessaging.getInstance().subscribeToTopic("Free");
```

затем в консоли FireBase отправьте уведомление по названию темы

Дополнительная информация в специальном разделе Firebase Cloud Messaging .

Добавить Firebase в свой Android-проект

Вот упрощенные шаги (на основе официальной документации), необходимые для создания проекта Firebase и подключения его с Android-приложением.

Добавить Firebase в приложение

1. Создайте проект Firebase в консоли Firebase и нажмите « **Создать новый проект**» .
2. Нажмите « **Добавить Firebase» в приложение Android** и выполните шаги настройки.
3. При появлении запроса введите имя своего **приложения** .
   Важно ввести полное имя пакета, используемое вашим приложением; это можно установить только при добавлении приложения в проект Firebase.
4. В конце вы загрузите файл `google-services.json` . Вы можете загрузить этот файл снова в любое время.
5. Если вы еще этого не сделали, скопируйте файл `google-services.json` в папку модуля вашего проекта, как правило, `app/` .

Следующим шагом является добавление SDK для интеграции библиотек Firebase в проект.

Добавить SDK

Чтобы интегрировать библиотеки Firebase в один из ваших собственных проектов, вам нужно выполнить несколько основных задач для подготовки проекта Android Studio.

Возможно, вы уже сделали это как часть добавления Firebase в свое приложение.

1. Добавьте правила в файл `build.gradle` уровне `build.gradle` , чтобы включить **плагин google-services** :

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.1.0'
    }
}
```

Затем в своем модуле Gradle (обычно это `app/build.gradle`) добавьте строку плагина apply в нижней части файла, чтобы включить плагин 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'
```

Последний шаг - добавить зависимости для Firebase SDK, используя одну или несколько **библиотек, доступных** для различных функций Firebase.

Линия зависимостей градля	обслуживание
com.google.firebase: firebase-ядро: 11.0.4	аналитика
com.google.firebase: firebase-база данных: 11.0.4	База данных реального времени
com.google.firebase: firebase-хранения: 11.0.4	Место хранения
com.google.firebase: firebase-аварии: 11.0.4	Отчеты о сбоях
com.google.firebase: firebase-аутентификации: 11.0.4	Аутентификация
com.google.firebase: firebase-сообщения: 11.0.4	Облачные сообщения / уведомления
com.google.firebase: firebase-конфигурации: 11.0.4	Удаленная настройка

Линия зависимостей градля	обслуживание
com.google.firebase: firebase-инвайты: 11.0.4	Приглашения / Динамические ссылки
com.google.firebase: firebase-объявления: 11.0.4	AdMob
com.google.android.gms: игры-сервисы appindexing: 11.0.4	Индексирование приложений

База данных реального времени Firebase: как установить / получить данные

Примечание. Давайте установим некоторую анонимную аутентификацию для примера.

```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

Как только это будет сделано, создайте ребенка, отредактировав адрес своей базы данных. Например:

https://your-project.firebaseio.com/ **https://your-project.firebaseio.com/chat**

Мы поместим данные в это местоположение с нашего Android-устройства. Вам **не нужно** создавать структуру базы данных (вкладки, поля ... и т. Д.), Она будет автоматически создана, когда вы отправите объект Java в Firebase!

Создайте объект Java, который содержит все атрибуты, которые вы хотите отправить в базу данных:

```
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;
    }
```

```
}
```

Затем в вашей деятельности:

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

Чтобы отправить значение:

```
ChatMessage msg = new ChatMessage("user1", "Hello World!");
reference.push().setValue(msg);
```

Для получения изменений, которые происходят в базе данных:

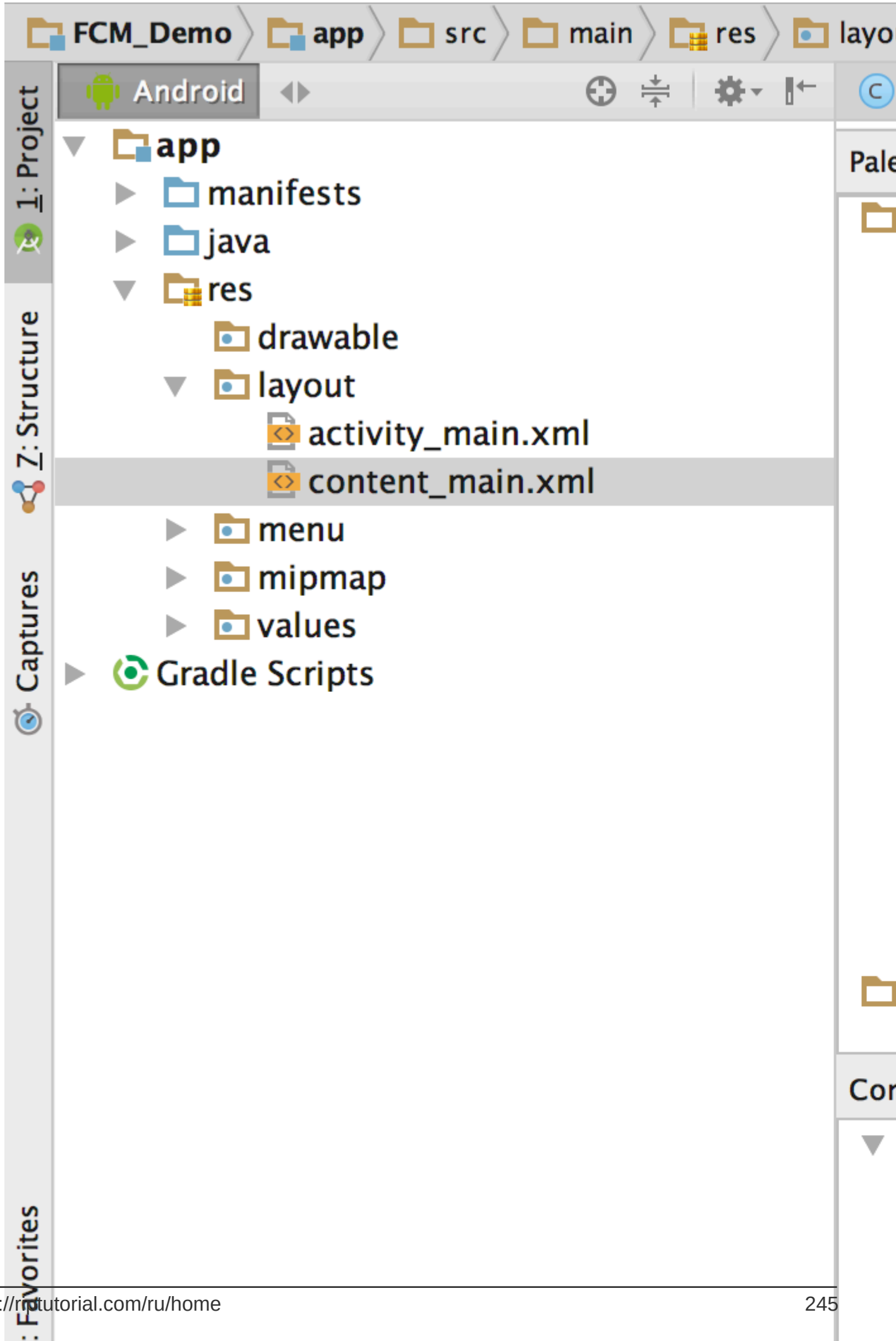
```
reference.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        ChatMessage msg = dataSnapshot.getValue(ChatMessage.class);
        Log.d(TAG, msg.getUsername()+" "+msg.getMessage());
    }

    public void onChildChanged(DataSnapshot dataSnapshot, String s) {}
    public void onChildRemoved(DataSnapshot dataSnapshot) {}
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {}
    public void onCancelled(DatabaseError databaseError) {}
});
```

Демонстрация уведомлений на основе FCM

В этом примере показано, как использовать платформу Firebase Cloud Messaging (FCM). FCM является преемником Google Cloud Messaging (GCM). Он не требует разрешения C2D_MESSAGE от пользователей приложения.

Шаги по интеграции FCM заключаются в следующем.

1. Создайте образец проекта hello world в Android Studio. Экран студии Android будет выглядеть следующим образом.


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

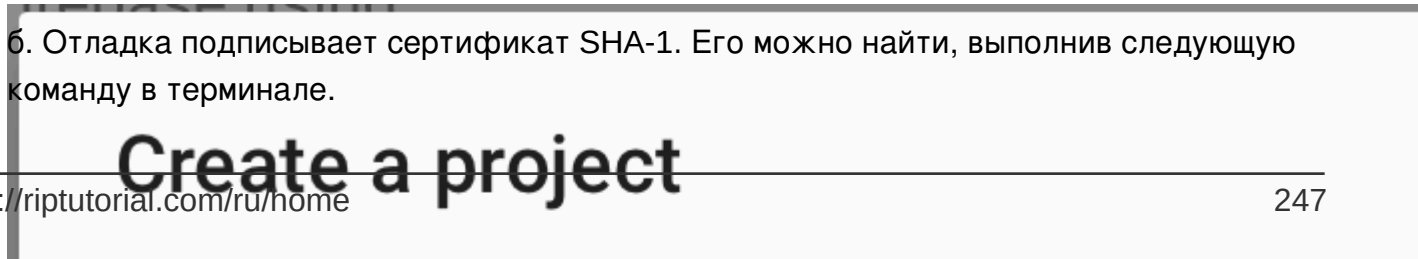
2. Следующий шаг - настроить проект firebase. Перейдите на страницу https://console.firebase.google.com и создайте проект с одинаковым именем, чтобы вы могли легко отслеживать его.

https://console.firebase.google.com и создайте проект с одинаковым именем, чтобы вы могли легко отслеживать его.

3. Теперь пришло время добавить firebase к вашему образцу проекта Android, который вы только что создали. Вам понадобится имя пакета вашего проекта и сертификат подписи отладки SHA-1 (необязательно).

   а. Имя пакета. Его можно найти в файле XML манифеста Android.

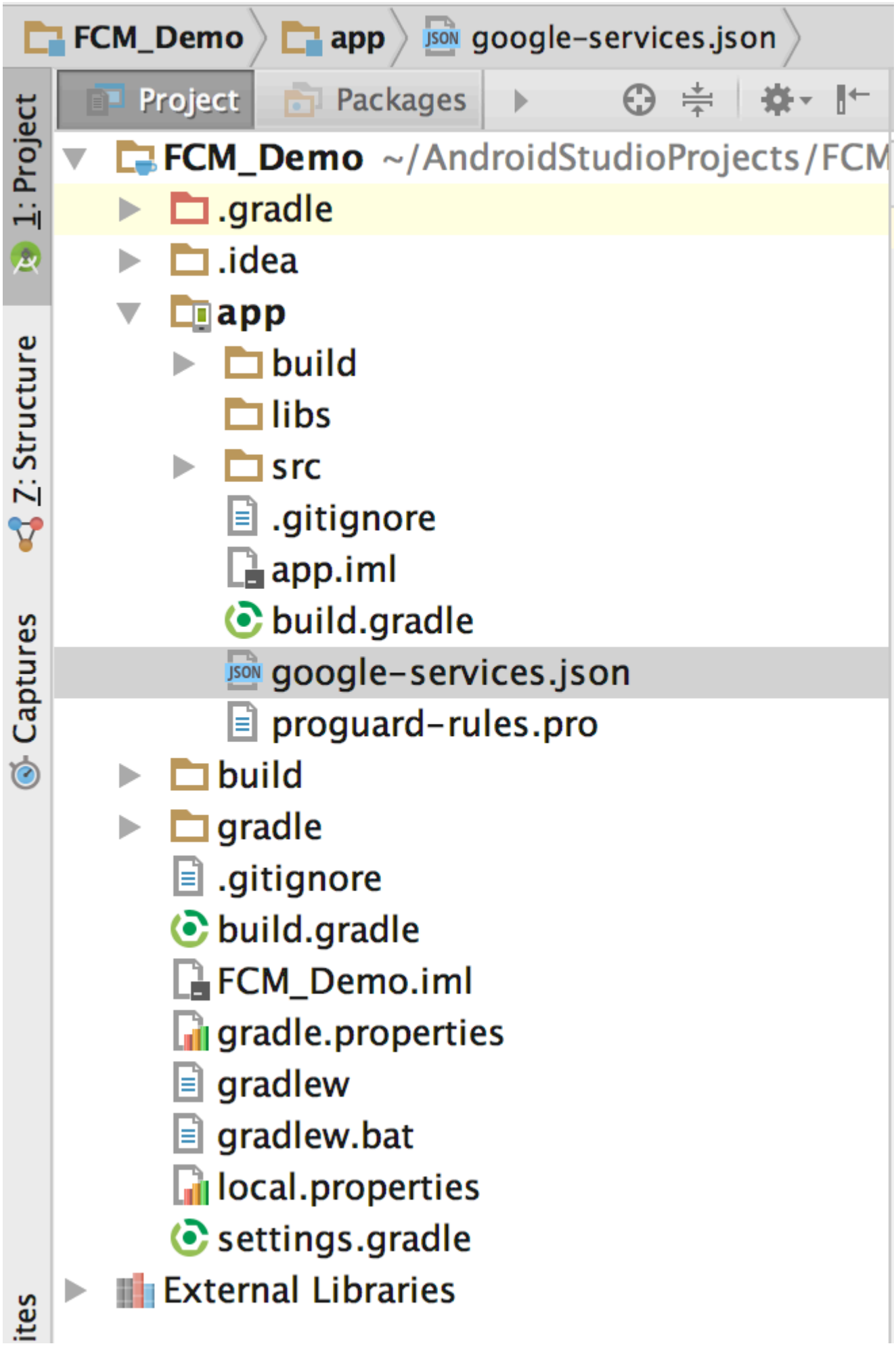
   б. Отладка подписывает сертификат SHA-1. Его можно найти, выполнив следующую команду в терминале.



```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -
keypass android
```

Введите эту информацию в консоли firebase и добавьте приложение в проект firebase. Как только вы нажмете кнопку добавления приложения, ваш браузер автоматически загрузит файл JSON с именем «google-services.json».

4. Теперь скопируйте файл google-services.json, который вы только что загрузили в корневой каталог вашего приложения 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. Следуйте инструкциям, приведенным на консоли firebase, по мере продвижения вперед. а. Добавьте следующую строку кода на свой уровень проекта build.gradle

```
dependencies{ classpath 'com.google.gms:google-services:3.1.0' .....
```

б. Добавьте следующую строку кода в конец вашего уровня сборки 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. Студия Android попросит вас синхронизировать проект. Нажмите «Синхронизировать сейчас».

6. Следующая задача - добавить две службы. а. Один расширяющий FirebaseMessagingService с фильтром намерения следующим образом

```
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
```

б. Один расширяет FirebaseInstanceIDService.

```
<intent-filter>
    <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
</intent-filter>
```

7. Код FirebaseMessagingService должен выглядеть так.

```
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 должен выглядеть так.

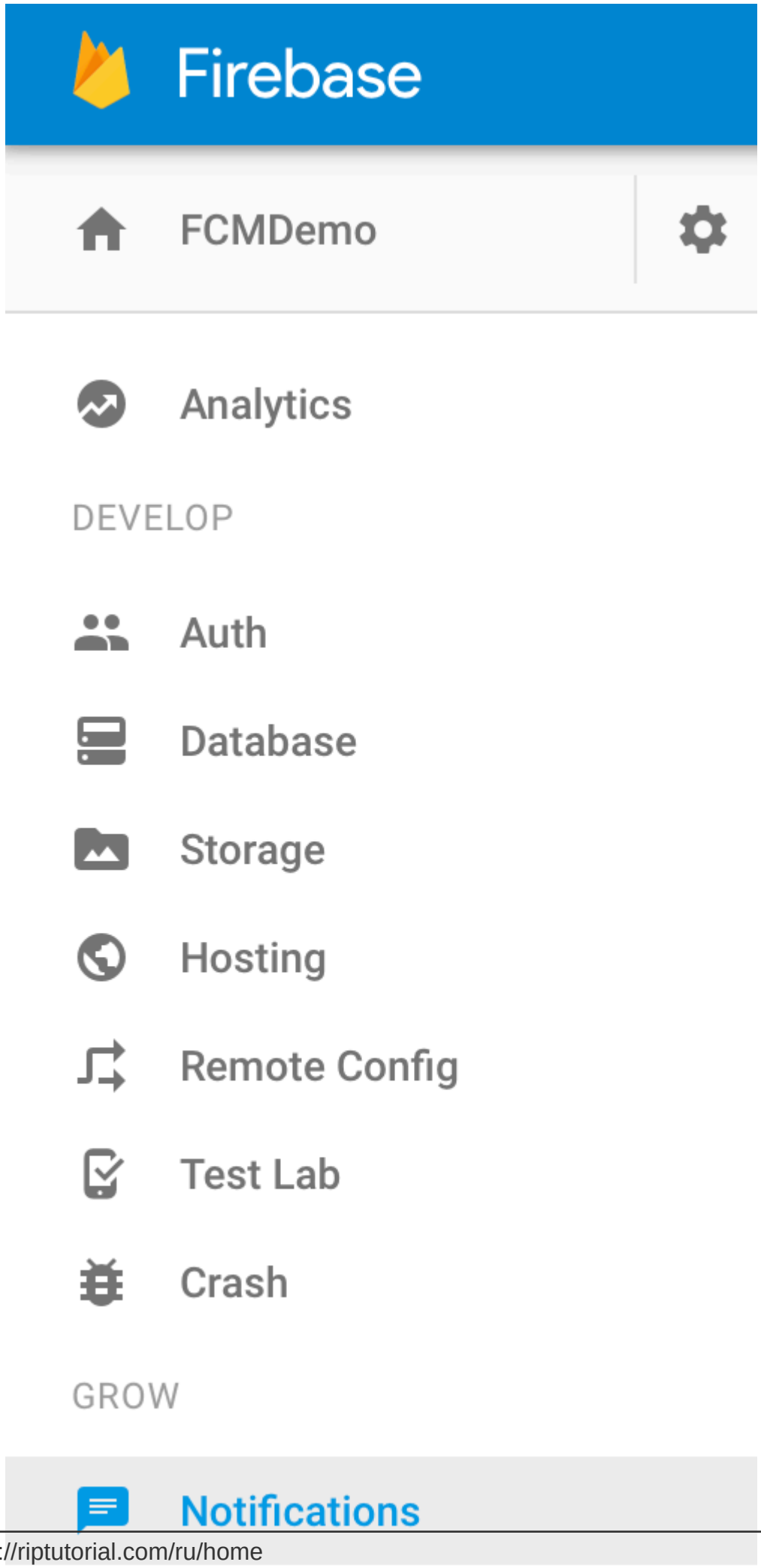
```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

import com.google.firebase.iid.FirebaseInstanceIdService;

public class MyFirebaseInstanceIDService extends FirebaseInstanceIdService {
```

```
    public MyFirebaseInstanceIDService() {
    }
}
```

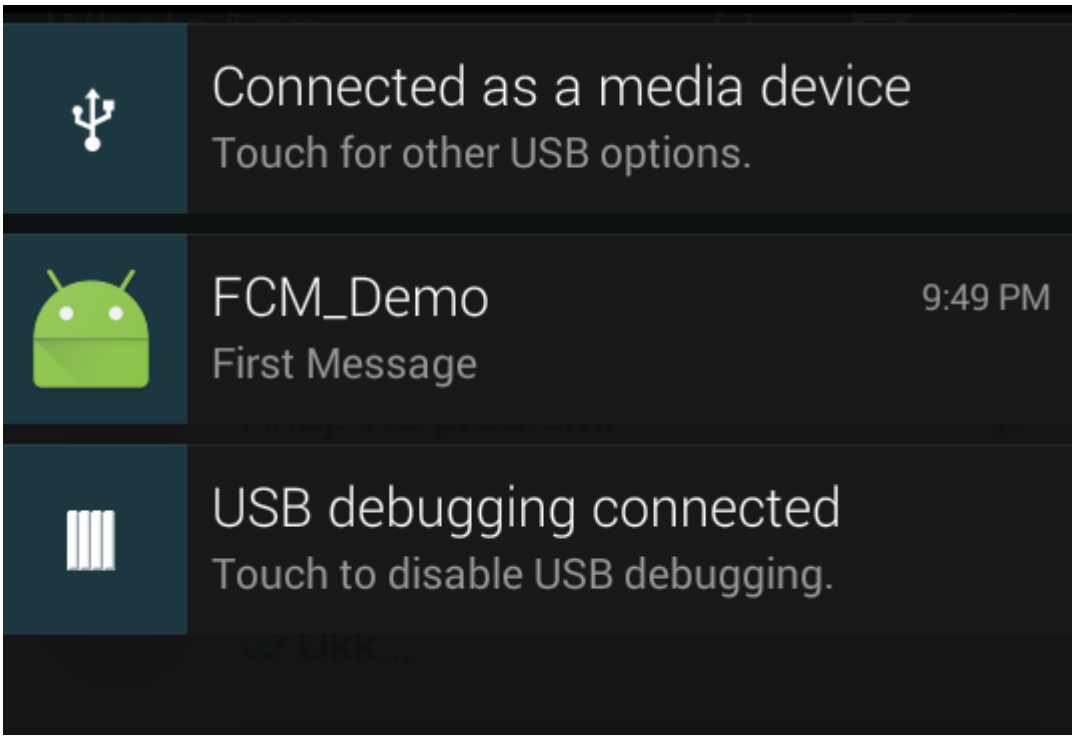
9. Теперь пришло время захватить токен регистрации устройства. Добавьте следующую строку кода в метод onCreate MainActivity.

```
String token = FirebaseInstanceId.getInstance().getToken();
Log.d("FCMAPP", "Token is "+token);
```

10. Когда у нас есть токен доступа, мы можем использовать консоль firebase для отправки уведомления. Запустите приложение на телефоне Android.


Firebase
FCMDemo
Analytics
DEVELOP
Auth
Database
Storage
Hosting
Remote Config
Test Lab
Crash
GROW
Notifications

и пользовательский интерфейс поможет вам отправить ваше первое сообщение. Firebase предлагает функциональные возможности для отправки сообщений на одно устройство (используя идентификатор токена устройства, который мы зафиксировали) или всех пользователей, использующих наше приложение или для определенной группы пользователей. После отправки вашего первого сообщения ваш мобильный экран должен выглядеть следующим образом.



Спасибо

Выйти из Firebase

Инициализация переменной

```
private GoogleApiClient mGoogleApiClient;
```

Вы должны написать этот код в методе onCreate () всего, что когда u выведет кнопку выключения.

```
 mGoogleApiClient = new GoogleApiClient.Builder(this)
            .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener
*/)
            .addApi(Auth.GOOGLE_SIGN_IN_API)
            .build();
```

Поместите ниже код на кнопку выписки.

```
Auth.GoogleSignInApi.signOut(mGoogleApiClient).setResultCallback(
                    new ResultCallback<Status>() {
                        @Override
```

```
                        public void onResult(Status status) {
                            FirebaseAuth.getInstance().signOut();
                            Intent i1 = new Intent(MainActivity.this,
GoogleSignInActivity.class);
                            startActivity(i1);
                            Toast.makeText(MainActivity.this, "Logout Successfully!",
Toast.LENGTH_SHORT).show();
                        }
                    });
```

Прочитайте Firebase онлайн: https://riptutorial.com/ru/android/topic/3843/firebase

глава 38: FloatingActionButton

Вступление

Кнопка плавающего действия используется для особого типа продвигаемого действия, по умолчанию оно анимируется на экране как расширяющийся кусок материала. Значок внутри него может быть анимированным, также FAB может двигаться иначе, чем другие элементы пользовательского интерфейса из-за их относительной важности. Кнопка плавающего действия представляет основное действие в приложении, которое может просто инициировать действие или перемещаться где-нибудь.

параметры

параметр	подробность
`android.support.design:elevation`	Значение высоты для FAB. Может быть ссылкой на другой ресурс в форме «@ [+] [package:] type / name» или атрибут темы в форме «? [Package:] type / name».
`android.support.design:fabSize`	Размер для FAB.
`android.support.design:rippleColor`	Цвет пульсации для FAB.
`android.support.design:useCompatPadding`	Включить прошивку.

замечания

Кнопки с плавающим действием используются для специального типа продвигаемого действия. Они отличаются кружком плавающим значком над UI и имеют специальные поведения движения, связанные с морфинг, запуска и передающей точкой привязки.

Для отображения наиболее частого действия рекомендуется использовать только одну кнопку плавающего действия.

Перед использованием `FloatingActionButton` вы должны добавить зависимость библиотеки поддержки дизайна в файле `build.gradle` :

```
dependencies {
    compile 'com.android.support:design:25.1.0'
}
```

Официальная документация:

https://developer.android.com/reference/android/support/design/widget/FloatingActionButton.html

Технические характеристики материалов:

https://material.google.com/components/buttons-floating-action-button.html

Examples

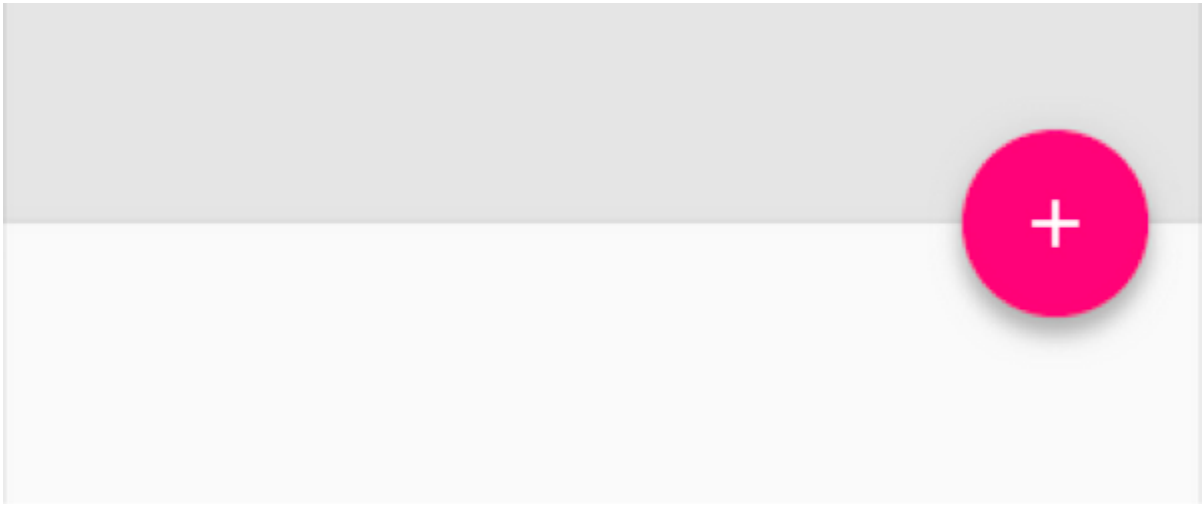
Как добавить FAB к макету

Чтобы использовать FloatingActionButton, просто добавьте зависимость в файле `build.gradle` как описано в разделе примечаний.

Затем добавьте в макет:

```
<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" />
```

Пример:



цвет

Цвет фона этого представления по умолчанию соответствует цвету вашей темы.

В приведенном выше изображении, если `src` указывает только на значок + (по умолчанию 24x24 dp), чтобы получить *фоновый цвет* полного круга, вы можете использовать `app:backgroundTint="@color/your_colour"`

Если вы хотите изменить цвет кода, который вы можете использовать,

```
myFab.setBackgroundTintList(ColorStateList.valueOf(your color in int));
```

Если вы хотите изменить цвет FAB в нажатом состоянии, используйте

```
mFab.setRippleColor(your color in int);
```

позиционирование

Рекомендуется установить минимум 16dp с края на мобильном устройстве и минимум 24dp на планшет / рабочий стол.

Примечание . После того, как вы установите src, за исключением того, чтобы покрыть всю область `FloatingActionButton` убедитесь, что у вас есть правильный размер этого изображения, чтобы получить лучший результат.

Размер круга по умолчанию - 56 x 56dp

Размер мини-круга: 40 x 40dp

Если вы хотите изменить только значок «Интерьер», используйте значок 24 x 24dp для размера по умолчанию

Показать и скрыть FloatingActionButton на прокрутке

Чтобы показать и скрыть `FloatingActionButton` с анимацией по умолчанию, просто вызовите методы `show()` и `hide()` . Хорошая практика - сохранять `FloatingActionButton` в макете Activity вместо того, чтобы помещать его во Фрагмент, это позволяет работать с анимацией по умолчанию при показе и скрытии.

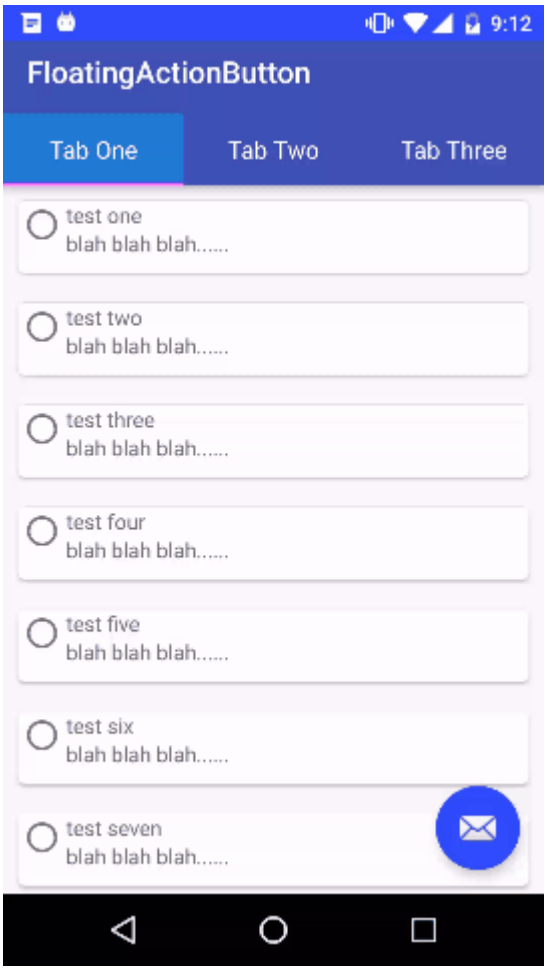
Вот пример с `ViewPager` :

- Три вкладки
- Показать `FloatingActionButton` для первой и третьей вкладки
- Скрыть `FloatingActionButton` на средней вкладке

```
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();
                }
            }
        });

    }
}
```

Результат:



Показать и скрыть FloatingActionButton на прокрутке

Начиная с версии Support Library версии 22.2.1, можно показать и скрыть FloatingActionButton из поведения прокрутки с помощью подкласса `FloatingActionButton.Behavior` который использует методы `show()` и `hide()` .

Обратите внимание, что это работает только с CoordinatorLayout в сочетании с внутренними представлениями, которые поддерживают вложенную прокрутку, например RecyclerView и NestedScrollView .

Этот класс `ScrollAwareFABBehavior` поставляется с Android Guides на Codepath (cc-wiki с требуемой атрибуцией)

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

В макете FloatingActionButton xml укажите `app:layout_behavior` с полным именем класса `ScrollAwareFABBehavior` :

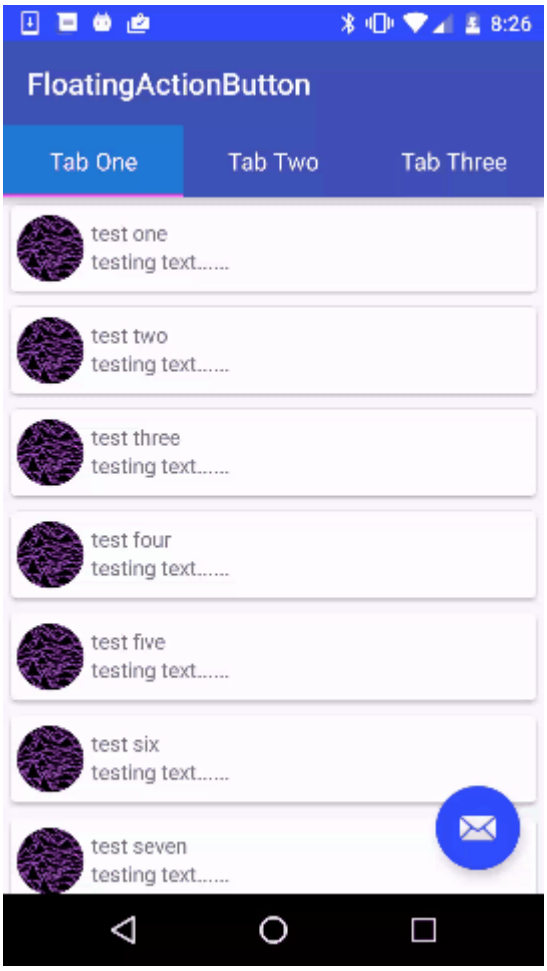
```
app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
```

Например, с помощью этого макета:

```
<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>
```

Вот результат:



Настройка поведения FloatingActionButton

Вы можете установить поведение FAB в XML.

Например:

```
<android.support.design.widget.FloatingActionButton
   app:layout_behavior=".MyBehavior" />
```

Или вы можете программно использовать:

```
CoordinatorLayout.LayoutParams p = (CoordinatorLayout.LayoutParams) fab.getLayoutParams();
p.setBehavior(xxxx);
fab.setLayoutParams(p);
```

Прочитайте FloatingActionButton онлайн:
https://riptutorial.com/ru/android/topic/2979/floatingactionbutton

глава 39: Genymotion для android

Вступление

Genymotion - это быстрый сторонний эмулятор, который можно использовать вместо стандартного Android-эмулятора. В некоторых случаях это так же хорошо или лучше, чем на реальных устройствах!

Examples

Установка Genymotion, бесплатная версия

Шаг 1 - установка `VirtualBox`

Загрузите и установите VirtualBox в соответствии с вашей операционной системой. , требуется запустить `Genymotion` .

Шаг 2 - загрузка `Genymotion`

Перейдите на страницу загрузки `Genymotion` и загрузите `Genymotion` соответствии с вашей операционной системой.

Примечание. Вам нужно будет создать новую учетную запись или войти в свою учетную запись.

Шаг 3 - Установка `Genymotion`

если в `Linux` то обратитесь к этому ответу , чтобы установить и запустить `.bin` файл.

Шаг 4 - Установка эмуляторов `Genymotion`

- запустить `Genymotion`
- Нажмите кнопку «Добавить» (в верхней панели).
- Войдите в свою учетную запись, и вы сможете просматривать доступные эмуляторы.
- выберите и установите то, что вам нужно.

Шаг 5 - Интеграция `genymotion` с `Android Studio`

`Genymotion` , можно интегрировать с `Android Studio` через плагин, здесь шаги по установке в `Android Studio`

- перейдите в раздел «Файл / Настройки» (для Windows и Linux) или в Android Studio / Preferences (для Mac OS X)
- Выберите «Плагины» и нажмите «Обзор репозиториев».
- Щелкните правой кнопкой мыши на Genymotion и нажмите «Загрузить и установите».

Теперь вы можете увидеть значок плагина, см. Это изображение

Обратите внимание: вы можете отобразить панель инструментов, нажав «Просмотр»> «Панель инструментов».

Шаг 6 - Запуск `Genymotion` из `Android Studio`

- перейдите в раздел «Файл / Настройки» (для Windows и Linux) или в Android Studio / Preferences (для Mac OS X)
- перейдите в раздел Другие настройки / Genymotion и добавьте путь `Genymotion's` папке `Genymotion's` и примените свои изменения.

Теперь вы можете запустить эмулятор `Genymotion's` нажав значок плагина и выбрав установленный эмулятор, а затем нажмите кнопку запуска!

Рамка Google для Genymotion

Если разработчики хотят тестировать Карты Google или любые другие сервисы Google, такие как Gmail, Youtube, Google диск и т. Д., То сначала им нужно установить Google framework на Genymotion. Вот шаги: -

4.4 Киткат
5.0 Lollipop
5.1 Lollipop
6.0 Зефир
7,0 Нуга
7.1 Нуга (патч для веб-просмотра)

1. Скачать по ссылке выше
2. Просто перетащите и отпустите загруженный zip-файл в genymotion и перезапустите
3. Добавьте учетную запись google и загрузите «Google Play Музыка» и запустите.

Ссылка:-
Вопрос о переполнении стека по этой теме

Прочитайте Genymotion для android онлайн:

https://riptutorial.com/ru/android/topic/9245/genymotion-для-android

глава 40: Google Maps API v2 для Android

параметры

параметр	подробности
Google Map	`GoogleMap` - это объект, который получен в `onMapReady()`
MarkerOptions	`MarkerOptions` является классом строителя `Marker` и используется для добавления одного маркера к карте.

замечания

Требования

1. Установлен SDK Google Play Services.
2. Учетная запись консоли Google.
3. Ключ API Карт Google, полученный в Google Консоли.

Examples

Активность по умолчанию в Google Map

Этот код действия предоставит базовую функциональность для включения Карты Google с помощью SupportMapFragment.

API Google Maps V2 включает в себя совершенно новый способ загрузки карт.

Теперь теперь необходимо реализовать интерфейс **OnMapReadyCallBack** , который поставляется с переопределением метода **onMapReady (),** который выполняется каждый раз, когда мы запускаем **SupportMapFragment** . **getMapAsync (OnMapReadyCallback)** ; и звонок успешно завершен.

Карты используют **маркеры** , **полигоны** и **полилинии** для отображения интерактивной информации пользователю.

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));
    }
}
```

Обратите внимание, что приведенный выше код раздувает макет, который имеет поддержку SupportMapFragment, вложенную в макет контейнера, определенный с идентификатором `R.id.map` . Файл макета показан ниже:

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>
```

Пользовательские стили Google Map

Стиль карты

Карты Google поставляются с набором различных стилей, которые будут применяться, используя этот код:

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

Различные стили карт:

Нормальный

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

Типичная дорожная карта. Показаны дороги, некоторые искусственные черты и важные природные объекты, такие как реки. Дорожки и ярлыки функций также видны.

Гибридный

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

Добавлены данные спутниковой фотографии с дорожными картами. Дорожки и ярлыки функций также видны.

спутник

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

Данные спутниковой фотографии. Дорожные знаки и ярлыки функций не видны.

местность

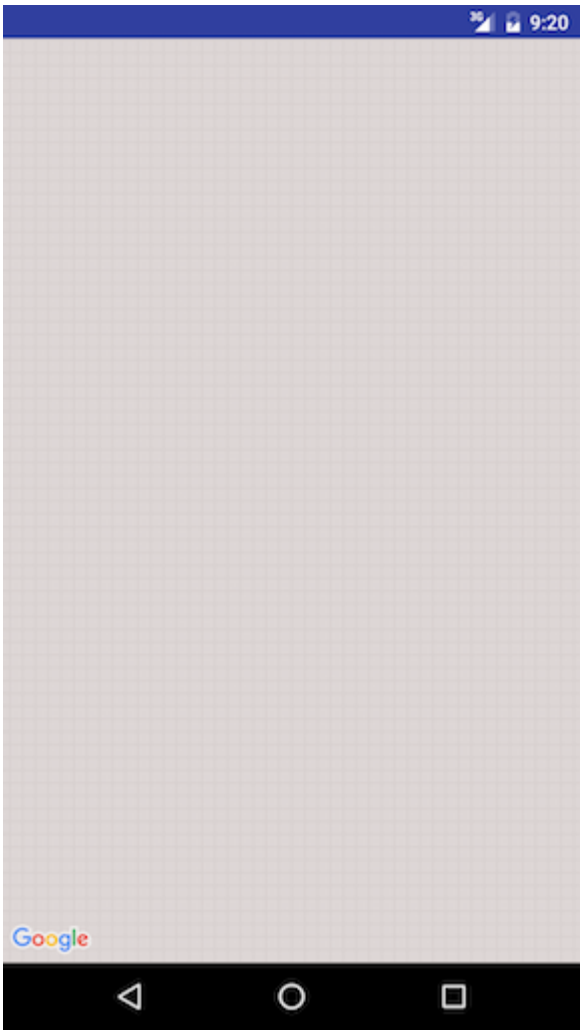
```
map.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

Топографические данные. Карта включает в себя цвета, контурные линии и метки и перспективное затенение. Некоторые дороги и этикетки также видны.

Никто

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

Нет плиток. Карта будет отображаться как пустая сетка без загрузки плиток.



ПРОЧИЕ ВАРИАНТЫ СТИЛЯ

Внутренние карты

При высоких уровнях масштабирования на карте будут отображаться планы этажей для внутренних помещений. Они называются закрытыми картами и отображаются только для типов «нормальных» и «спутниковых» карт.

чтобы включить или отключить внутренние карты, вот как это делается:

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

Мы можем добавлять пользовательские стили к картам.

В методе onMapReady добавьте следующий фрагмент кода

```
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);
  }
```

под папкой *res* создайте имя папки *raw* и добавьте стили json-файла. Пример файла style.json

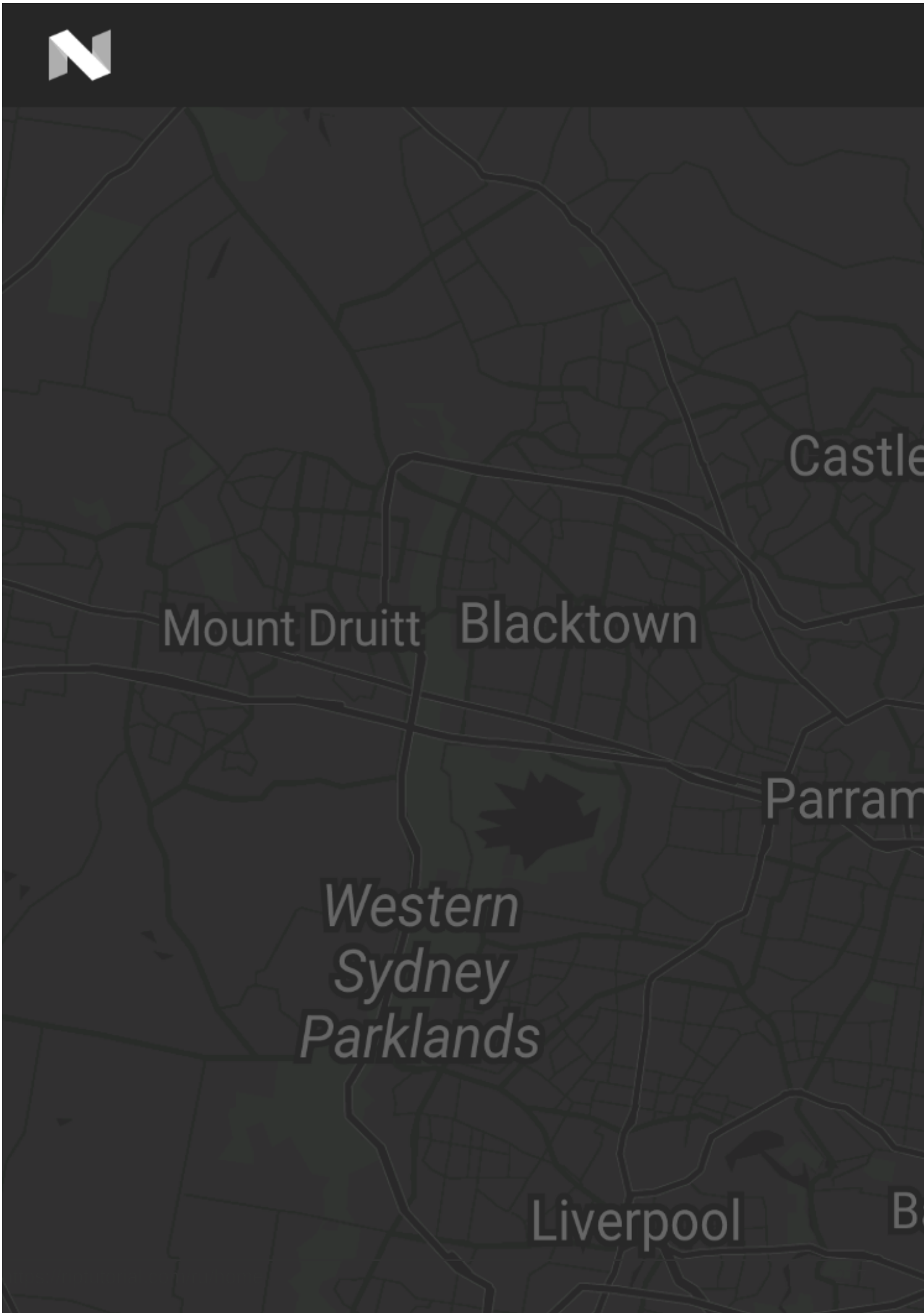
```
  [
{
  "featureType": "all",
  "elementType": "geometry",
  "stylers": [
    {
      "color": "#242f3e"
    }
  ]
},
{
  "featureType": "all",
  "elementType": "labels.text.stroke",
  "stylers": [
    {
      "lightness": -80
    }
  ]
},
{
  "featureType": "administrative",
  "elementType": "labels.text.fill",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

Для создания стилей json-файла нажмите эту ссылку


Castle
Mount Druitt
Blacktown
Parram
Western
Sydney
Parklands
Liverpool

мы можем сделать это таким образом.

`MyLocation` держателя `MyLocation` :

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

Вот метод, который бы взял список объектов `MyLocation` и `MyLocation` маркер для каждого из них:

```
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));
 }
}
```

Примечание. Для целей этого примера `mMap` является переменной-членом класса Activity, где мы присвоили ей ссылку на карту, полученную в переопределении `onMapReady()` .

MapView: внедрение GoogleMap в существующую компоновку

Можно рассматривать GoogleMap как представление Android, если мы используем предоставленный класс MapView. Его использование очень похоже на MapFragment.

В вашем макете используйте MapView следующим образом:

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

Для работы необходимо реализовать интерфейс OnMapReadyCallback для работы:

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

Показать текущее местоположение на карте Google

Вот полный класс активности, который помещает маркер в текущее местоположение, а также перемещает камеру в текущую позицию.

Здесь происходит несколько вещей:

- Проверить разрешение на размещение
- После предоставления разрешения на размещение вызовите `setMyLocationEnabled()`, создайте GoogleApiClient и подключите его
- После подключения GoogleApiClient запросите обновления местоположения

```
public class MapLocationActivity extends AppCompatActivity
        implements OnMapReadyCallback,
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        LocationListener {

    GoogleMap mGoogleMap;
    SupportMapFragment mapFrag;
    LocationRequest mLocationRequest;
    GoogleApiClient mGoogleApiClient;
    Location mLastLocation;
    Marker mCurrLocationMarker;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        getSupportActionBar().setTitle("Map Location Activity");

        mapFrag = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
        mapFrag.getMapAsync(this);
    }

    @Override
    public void onPause() {
        super.onPause();

        //stop location updates when Activity is no longer active
        if (mGoogleApiClient != null) {
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
        }
    }

    @Override
    public void onMapReady(GoogleMap googleMap)
    {
```

```
        mGoogleMap=googleMap;
        mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);

        //Initialize Google Play Services
        if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)
                    == PackageManager.PERMISSION_GRANTED) {
                //Location Permission already granted
                buildGoogleApiClient();
                mGoogleMap.setMyLocationEnabled(true);
            } else {
                //Request Location Permission
                checkLocationPermission();
            }
        }
        else {
            buildGoogleApiClient();
            mGoogleMap.setMyLocationEnabled(true);
        }
    }

    protected synchronized void buildGoogleApiClient() {
        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
        mGoogleApiClient.connect();
    }

    @Override
    public void onConnected(Bundle bundle) {
        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(1000);
        mLocationRequest.setFastestInterval(1000);
        mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
        }
    }

    @Override
    public void onConnectionSuspended(int i) {}

    @Override
    public void onConnectionFailed(ConnectionResult connectionResult) {}

    @Override
    public void onLocationChanged(Location location)
    {
        mLastLocation = location;
        if (mCurrLocationMarker != null) {
            mCurrLocationMarker.remove();
        }

        //Place current location marker
        LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
```

```
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.position(latLng);
        markerOptions.title("Current Position");

markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_MAGENTA));

        mCurrLocationMarker = mGoogleMap.addMarker(markerOptions);

        //move map camera
        mGoogleMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mGoogleMap.animateCamera(CameraUpdateFactory.zoomTo(11));

        //stop location updates
        if (mGoogleApiClient != null) {
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
        }
    }

    public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;
    private void checkLocationPermission() {
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED) {

            // Should we show an explanation?
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)) {

                // Show an explanation to the user *asynchronously* -- don't block
                // this thread waiting for the user's response! After the user
                // sees the explanation, try again to request the permission.
                new AlertDialog.Builder(this)
                        .setTitle("Location Permission Needed")
                        .setMessage("This app needs the Location permission, please accept to
use location functionality")
                        .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialogInterface, int i) {
                                //Prompt the user once explanation has been shown
                                ActivityCompat.requestPermissions(MapLocationActivity.this,
                                        new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                                        MY_PERMISSIONS_REQUEST_LOCATION );
                            }
                        })
                        .create()
                        .show();


            } else {
                // No explanation needed, we can request the permission.
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                        MY_PERMISSIONS_REQUEST_LOCATION );
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           String permissions[], int[] grantResults) {
        switch (requestCode) {
```

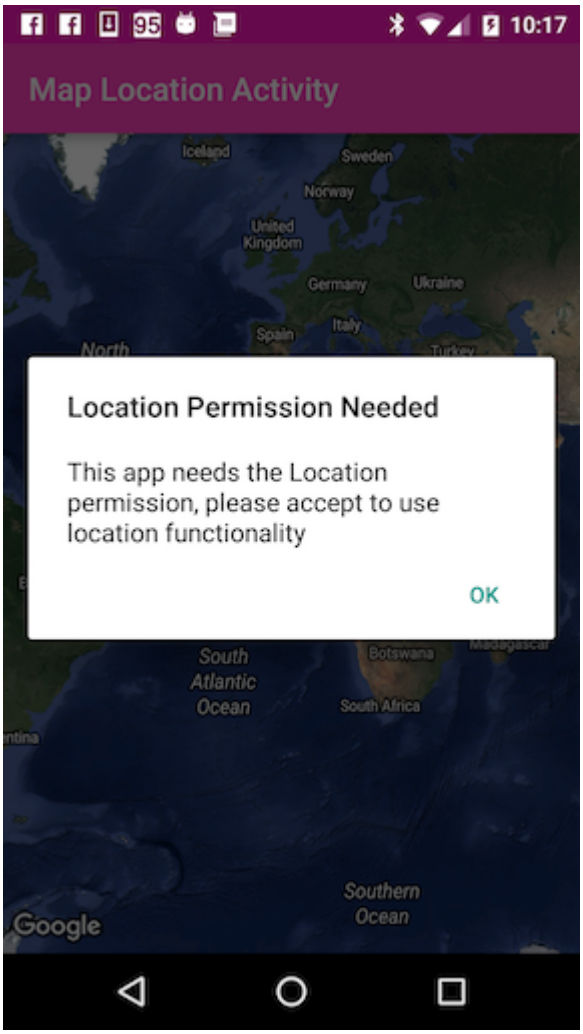
```
            case MY_PERMISSIONS_REQUEST_LOCATION: {
                // If request is cancelled, the result arrays are empty.
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

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

                        if (mGoogleApiClient == null) {
                            buildGoogleApiClient();
                        }
                        mGoogleMap.setMyLocationEnabled(true);
                    }

                } else {

                    // permission denied, boo! Disable the
                    // functionality that depends on this permission.
                    Toast.makeText(this, "permission denied", Toast.LENGTH_LONG).show();
                }
                return;
            }

            // other 'case' lines to check for other
            // permissions this app might request
        }
    }

}
```

activity_main.xml:

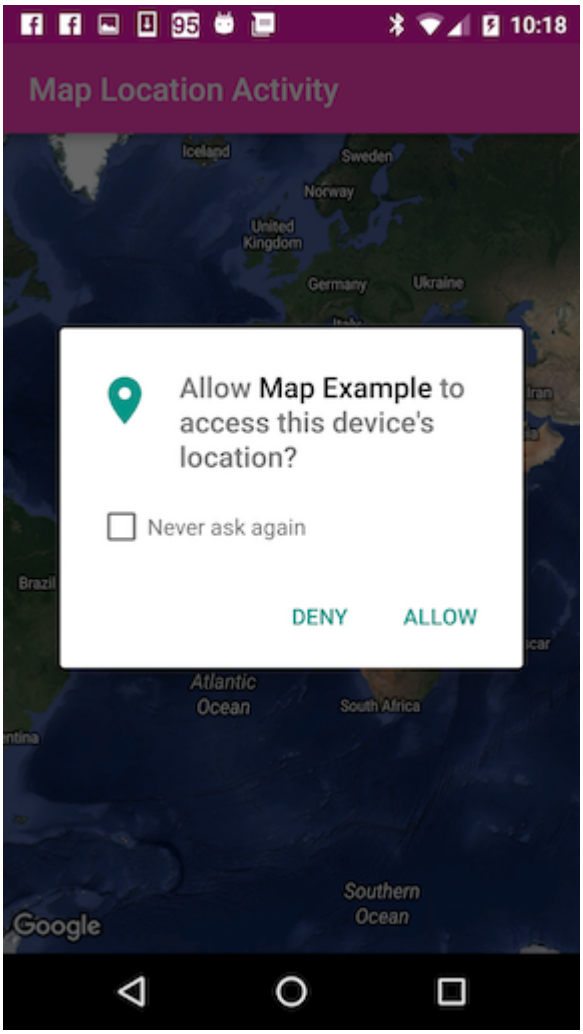
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:map="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/map"
        tools:context="com.example.app.MapLocationActivity"
        android:name="com.google.android.gms.maps.SupportMapFragment"/>

</LinearLayout>
```

Результат:

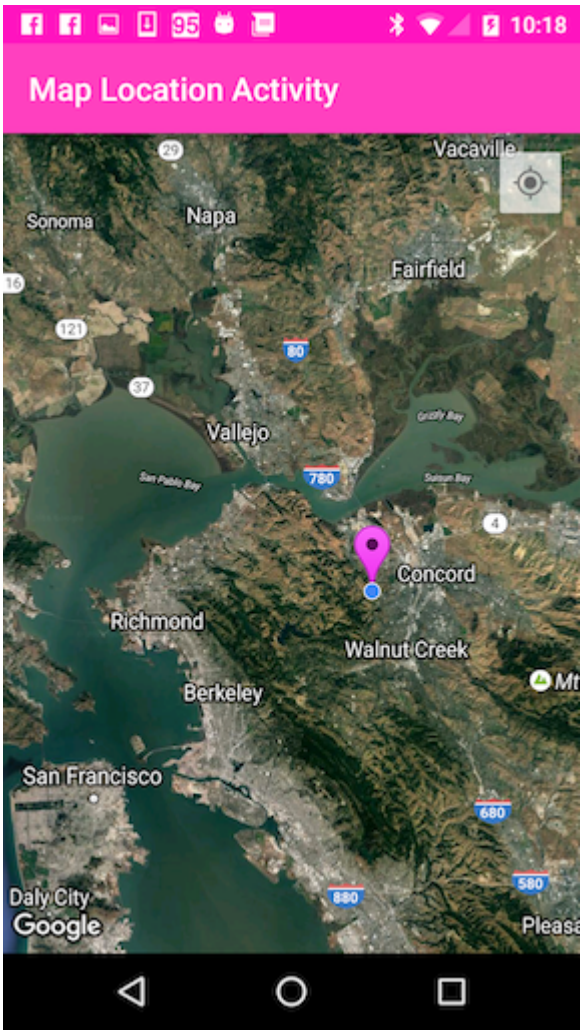
Покажите объяснение, если это необходимо для Marshmallow и Nougat, используя AlertDialog (этот случай случается, когда пользователь ранее отказал в запросе на разрешение или предоставил разрешение, а затем отозвал его в настройках):



Подскажите пользователю о разрешении местоположения в Marshmallow и Nougat, вызвав `ActivityCompat.requestPermissions()` :



Переместите камеру в текущее местоположение и поместите маркер при предоставлении разрешения на размещение:



Получение SH1-Отпечатка вашего файла хранилища сертификатов

Чтобы получить ключ API Карт Google для вашего сертификата, вы должны предоставить консоль API SH1-отпечатку своего хранилища отладки / выпуска.

Вы можете получить хранилище ключей, используя программу **keytool JDK,** как описано здесь в документах.

Другой подход - получить программный отпечаток пальца, выполнив этот фрагмент с вашим приложением, подписанным с сертификатом отладки / выпуска, и напечатайте хэш в журнале.

```
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());
}
```

Не запускайте Карты Google при нажатии на карту (режим Lite)

Когда карта Google отображается в режиме Lite, нажатие на карте откроет приложение Google Maps. Чтобы отключить эту функцию, вы должны вызвать `setClickable(false)` в `MapView` , *например* :

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

UISettings

Используя `UISettings` , можно изменить внешний вид Карты Google.

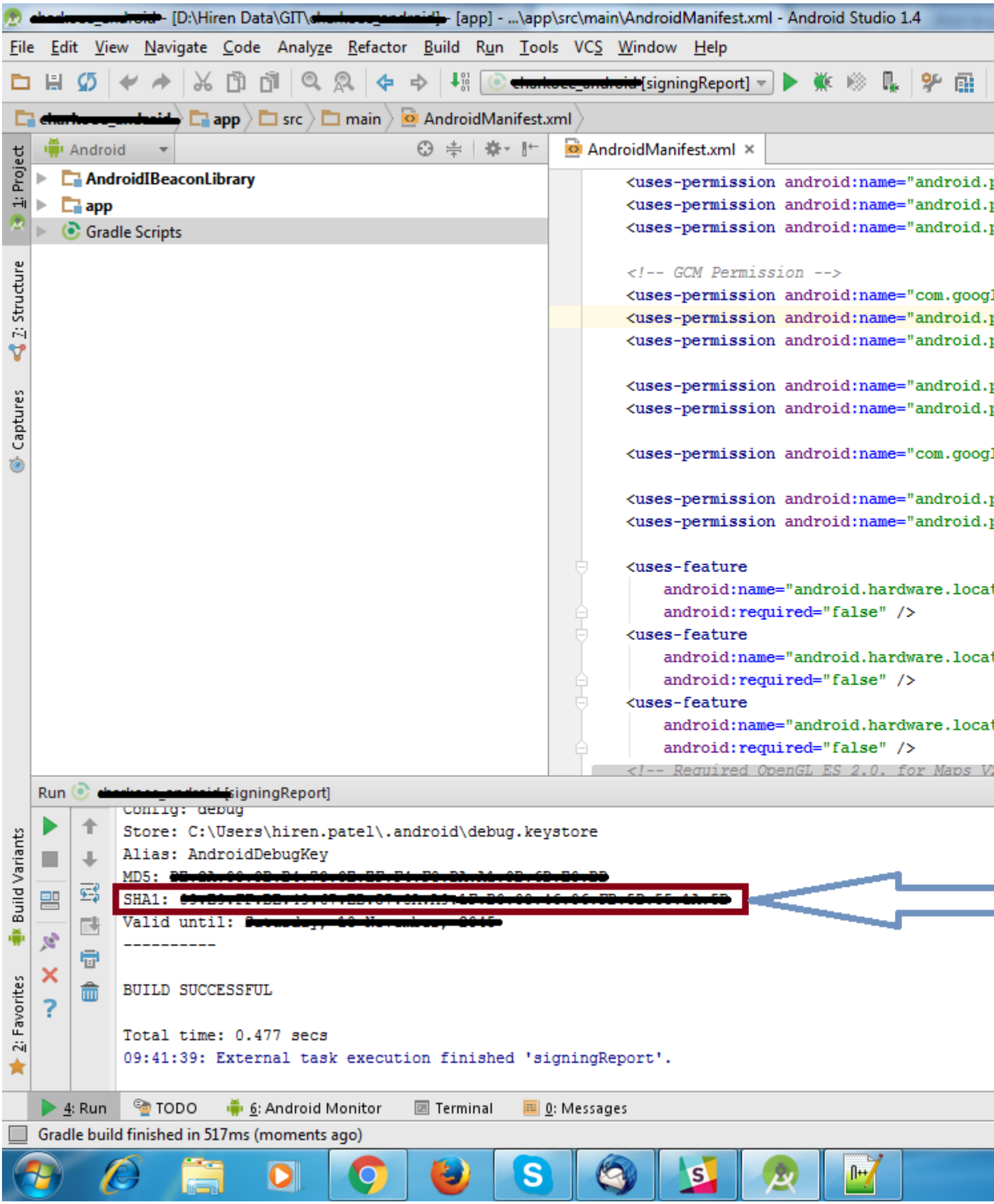
Вот пример некоторых общих настроек:

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

Результат:

Получить отладочный отпечаток SHA1

1. Открыть Android Studio
2. Открыть проект
3. Нажмите на Gradle (с правой стороны, вы увидите **Gradle Bar**)
4. Нажмите «Обновить» (нажмите «Обновить» с **панели «Грейдл»** , вы увидите «Сценарии **списка** градиентов» вашего проекта)
5. Нажмите на свой проект (**список** имен вашего проекта (root))
6. Нажмите «Задачи»
7. Нажмите на android
8. Двойной щелчок на signReport (вы получите **SHA1** и **MD5** в строке **запуска**)



InfoWindow Click Listener

Ниже приведен пример того, как определить другое действие для каждого события кликов

InfoWindow маркера.

Используйте HashMap, в котором идентификатор маркера является ключом, а значение - соответствующее действие, которое оно должно предпринять при щелчке по InfoWindow.

Затем используйте `OnInfoWindowClickListener` для обработки события пользователя, щелкнувшего InfoWindow, и используйте HashMap, чтобы определить, какое действие нужно предпринять.

В этом простом примере мы откроем другое действие, основанное на том, что было нажато InfoWindow Маркера.

Объявите HashMap в качестве переменной экземпляра Activity или Fragment:

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

Затем, каждый раз, когда вы добавляете маркер, делайте запись в HashMap с идентификатором маркера и действием, которое оно должно предпринять, когда нажимается InfoWindow.

Например, добавление двух маркеров и определение действия для каждого из них:

```
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");
```

В InfoWindow нажмите прослушиватель, получите действие из HashMap и откройте соответствующее действие, основанное на действии маркера:

```
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);
   }
 }
```

```
});
```

Примечание. Если код находится в фрагменте, замените MainActivity.this на getActivity ().

Изменение смещения

Изменяя значения mappoint *x* и *y по* мере необходимости, вы можете изменить смещение карты google, по умолчанию она будет находиться в центре отображения карты. Позвоните ниже метода, где вы хотите его изменить! Лучше использовать его внутри вашего `onLocationChanged` например `onLocationChanged` `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

        }
```

Прочитайте Google Maps API v2 для Android онлайн:
https://riptutorial.com/ru/android/topic/170/google-maps-api-v2-для-android

глава 41: Google Play магазин

Examples

Откройте список покупок в Google Play для своего приложения.

В следующем фрагменте кода показано, как безопасно открыть список покупок в Google Play Store. Обычно вы хотите использовать его, предлагая пользователю оставить отзыв для своего приложения.

```
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);
}
```

Примечание . В случае, если приложение установлено, код открывается в Google Play Store. В противном случае он просто откроет веб-браузер.

Откройте Google Play Store со списком всех приложений из учетной записи издателя

Вы можете добавить кнопку «Обзор наших других приложений» в своем приложении, в котором перечислены все ваши (издатели) приложения в приложении 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);
    }
}
```

Прочитайте Google Play магазин онлайн: https://riptutorial.com/ru/android/topic/10900/google-play-магазин

глава 42: Gradle для Android

Вступление

Gradle - это система сборки на основе JVM, которая позволяет разработчикам писать высокоуровневые сценарии, которые могут использоваться для автоматизации процесса компиляции и создания приложений. Это гибкая система на основе плагинов, которая позволяет автоматизировать различные аспекты процесса сборки; включая компиляцию и `.jar` , загрузку и управление внешними зависимостями, ввод полей в `AndroidManifest` или использование определенных версий SDK.

Синтаксис

- `apply plugin` : плагины, которые обычно должны использоваться только `'com.android.application'` или `'com.android.library'` .
- `android` : основная конфигурация вашего приложения
    - `compileSdkVersion` : скомпилированная версия SDK
    - `buildToolsVersion` : версия инструмента сборки
    - `defaultConfig` : настройки по умолчанию, которые могут быть перезаписаны вкусами и типами сборки
        - `applicationId` : идентификатор приложения, который вы используете, например, в PlayStore, в основном совпадает с именем вашего пакета
        - `minSdkVersion` : минимальная требуемая версия SDK
        - `targetSdkVersion` : SDK-версия, с которой вы компилируете (всегда должна быть новая)
        - `versionCode` : внутренний номер версии, который должен быть больше при каждом обновлении
        - `versionName` : номер версии, которую пользователь может видеть на странице сведений о приложении
    - `buildTypes` : см. где-то еще (TODO)
- `dependencies` : maven или локальные зависимости вашего приложения
    - `compile` одну зависимость
    - `testCompile` : зависимость для модульных или интеграционных тестов

замечания

Смотрите также

- Официальная страница gradle
- Как настроить градиентные сборки
- Плагин android для градиента
- Android Gradle DSL

Gradle for Android - Расширенная документация:

Существует еще один тег, где вы можете найти больше тем и примеров использования градиента в Android.
http://www.riptutorial.com/topic/2092

Examples

Базовый файл build.gradle

Это пример файла `build.gradle` умолчанию в модуле.

```
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 (язык, специфичный для домена)

Каждый блок в указанном выше файле называется `DSL` (специфичным для домена языком).

Плагины

В первой строке `apply plugin: 'com.android.application'` , применяет плагин Android для Gradle к сборке и блокирует блок `android {}` для объявления вариантов сборки для Android.

Для **Android-приложения** :

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

Для **Android-библиотеки** :

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

Понимание DSL в приведенном выше образце

Вторая часть, блок `android {...}` - это Android `DSL` который содержит информацию о вашем проекте.

Например, вы можете установить `compileSdkVersion` который определяет уровень API Android, который должен использоваться Gradle для компиляции вашего приложения. В `defaultConfig` содержатся значения по умолчанию для вашего манифеста. Вы можете `override` их с помощью продуктов Flavors .

Вы можете найти больше информации в этих примерах:

- DSL для модуля приложения
- Типы строений
- Ароматизаторы продуктов
- Настройки подписи

зависимости

Блок `dependencies` определяется вне блока `android {...}` : это означает, что он не определен плагином Android, но это стандартный Gradle.
Блок `dependencies` определяет, какие внешние библиотеки (как правило, библиотеки Android, но библиотеки Java также действительны), которые вы хотите включить в свое приложение. Gradle автоматически загрузит эти зависимости для вас (если локальная копия отсутствует), вам просто нужно добавить похожие строки `compile` если вы хотите добавить другую библиотеку.

Давайте посмотрим на одну из представленных здесь строк:

```
compile 'com.android.support:design:25.3.1'
```

Эта строка в основном говорит

> добавьте зависимость от библиотеки дизайна поддержки Android к моему проекту.

Gradle гарантирует, что библиотека будет загружена и представлена, чтобы вы могли использовать ее в своем приложении, а ее код также будет включен в ваше приложение.

Если вы знакомы с Maven, этот синтаксис - это *GroupId* , двоеточие, *ArtifactId* , другой двоеточие, а затем версия зависимости, которую вы хотите включить, что дает вам полный контроль над версиями.

Хотя можно указать версии артефакта, используя знак плюса (+), лучше всего избегать этого; это может привести к проблемам, если библиотека будет обновлена с нарушением изменений без вашего ведома, что, вероятно, приведет к сбоям в вашем приложении.

Вы можете добавить различные зависимости:

- локальные двоичные зависимости
- зависимости модулей
- удаленные зависимости

Особое внимание должно быть уделено плоской зависимости aar .

Вы можете найти более подробную информацию в этой теме.

Обратите внимание на **-v7 в *appcompat-v7***

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

Это просто означает, что эта **библиотека** (`appcompat`) совместима с уровнем API Android 7 и `appcompat` .

Примечание о **junit: junit: 4.12**

Это тестовая зависимость для модульного тестирования.

Указание зависимостей, характерных для разных конфигураций компоновки

Вы можете указать, что зависимость должна использоваться только для определенной конфигурации сборки или вы можете определить различные зависимости для типов сборки или продуктов (например, отладка, тестирование или выпуск) с помощью `debugCompile` , `testCompile` или `releaseCompile` вместо обычного `compile` ,

Это полезно для хранения зависимостей, связанных с тестированием и отладкой, из вашей сборки релиза, что позволит вашей версии `APK` как можно более тонкой и поможет гарантировать, что любая информация об отладке не может использоваться для получения внутренней информации о вашем приложении.

---

signingConfig

`signingConfig` позволяет вам настроить Gradle для включения информации о `keystore` и убедиться, что APK, построенный с использованием этих конфигураций, подписан и готов к выпуску Play Store.

Здесь вы можете найти специальную тему .

Примечание . Не рекомендуется сохранять учетные данные подписи в вашем файле Gradle. Чтобы удалить настройки подписи, просто опустите часть `signingConfigs` .
Вы можете указать их по-разному:

- сохранение во внешнем файле
- сохраняя их при настройке переменных среды .

См. Эту тему для получения более подробной информации: Подпишите APK без раскрытия пароля хранилища ключей .

Вы можете найти дополнительную информацию о Gradle для Android в теме Gradle .

Определение вкусов продукта

`build.gradle` продукта определены в файле `build.gradle` внутри блока `android { ... }` как

показано ниже.

```
...
android {
    ...
    productFlavors {
        free {
            applicationId "com.example.app.free"
            versionName "1.0-free"
        }
        paid {
            applicationId "com.example.app.paid"
            versionName "1.0-paid"
        }
    }
}
```

Делая это, у нас теперь есть два дополнительных продукта: `free` и `paid` . Каждый из них может иметь свою собственную конфигурацию и атрибуты. Например, оба наших новых вкуса имеют отдельное имя `applicationId` и `versionName` чем наш существующий `main` аромат (доступный по умолчанию, поэтому не показан здесь).

Добавление зависимых от продукта вкусовых зависимостей

Зависимости могут быть добавлены для конкретного вкуса продукта , аналогично тому, как они могут быть добавлены для конкретных конфигураций сборки.

В этом примере предположим, что мы уже определили два продукта, называемых `free` и `paid` (подробнее об определении вкусов здесь).
Затем мы можем добавить зависимость AdMob для `free` аромата и библиотеку Picasso для `paid` например:

```
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'
}
...
```

Добавление ресурсов, специфичных для продукта

Ресурсы могут быть добавлены для конкретного продукта .

В этом примере предположим, что мы уже определили два продукта, называемых `free` и `paid` . Чтобы добавить ресурсы, специфичные для продукта, мы создаем дополнительные папки ресурсов вместе с папкой `main/res` , которые затем можно добавить к ресурсам, как обычно. В этом примере мы определим строку, `status` , для каждого продукта:

*/ 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>
```

Строки `status` специфичные для вкуса продукта, переопределяют значение `status` в `main` ароматизаторе.

Определение и использование полей конфигурации сборки

BuildConfigField

Gradle позволяет `buildConfigField` определять константы. Эти константы будут доступны во время выполнения в качестве статических полей класса `BuildConfig` . Это можно использовать для создания ароматов , определяя все поля в блоке `defaultConfig` , а затем переопределяя их для отдельных `defaultConfig` сборки по мере необходимости.

В этом примере определяется дата сборки и флаги сборки для производства, а не теста:

```
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'
        }
    }
}
```

Автоматически созданное <имя_пакета>. `BuildConfig` .java в папке gen содержит следующие поля на основе указанной выше директивы:

```
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";
}
```

Определенные поля теперь можно использовать в приложении во время выполнения, `BuildConfig` к сгенерированному классу `BuildConfig` :

```
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

`resValue` в `productFlavors` создает значение ресурса. Это может быть любой тип ресурсов (`string` , `dimen` , `color` и т. Д.). Это похоже на определение ресурса в соответствующем

файле: например, определение строки в файле `strings.xml` . Преимущество состоит в том, что тот, который определен в градиенте, может быть изменен на основе вашего productFlavor / buildVariant. Чтобы получить доступ к значению, напишите тот же код, как если бы вы получали доступ к res из файла ресурсов:

```
getResources().getString(R.string.app_name)
```

Важно то, что ресурсы, определенные таким образом, не могут изменять существующие ресурсы, определенные в файлах. Они могут создавать только новые значения ресурсов.

Некоторым библиотекам (например, API Android для Google Maps) требуется ключ API, указанный в манифесте в качестве тега `meta-data` . Если для отладочной и производственной сборки нужны разные ключи, укажите манифест заполнителя, заполненный Gradle.

В вашем файле `AndroidManifest.xml` :

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="${MAPS_API_KEY}"/>
```

Затем установите соответствующее поле в файле `build.gradle` :

```
android {
    defaultConfig {
        ...
        // Your development key
        manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
    }

    productFlavors {
        prod {
            // Your production key
            manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
        }
    }
}
```

Система сборки Android автоматически генерирует несколько полей и помещает их в `BuildConfig.java` . Эти поля:

поле	Описание
`DEBUG`	`Boolean` сообщение, если приложение находится в режиме отладки или выпуска
`APPLICATION_ID`	`String` содержащая идентификатор приложения (например, `com.example.app`)

поле	Описание
`BUILD_TYPE`	`String` содержащая тип сборки приложения (обычно либо `debug` либо `release`)
`FLAVOR`	`String` содержащая особый аромат сборки
`VERSION_CODE`	`int` содержащий номер версии (сборки). Это то же самое , как `versionCode` в `build.gradle` или `versionCode` в `AndroidManifest.xml`
`VERSION_NAME`	`String` содержащая имя версии (сборки). Это то же самое , как `versionName` в `build.gradle` или `versionName` в `AndroidManifest.xml`

В дополнение к вышесказанному, если вы определили несколько параметров вкуса, то каждое измерение будет иметь свое значение. Например, если у вас есть два размера аромата для `color` и `size` вас также будут следующие переменные:

поле	Описание
`FLAVOR_color`	`String` содержащая значение для цветового оттенка.
`FLAVOR_size`	`String` содержащая значение для аромата размера.

Централизация зависимостей через файл «dependencies.gradle»

При работе с многомодульными проектами полезно централизовать зависимости в одном месте, а не распространять их во многих файлах сборки, особенно для обычных библиотек, таких как библиотеки поддержки Android и библиотеки Firebase .

Одним из рекомендуемых способов является разделение файлов сборки Gradle с одним `build.gradle` на модуль, а также с одним в корне проекта и другим для зависимостей, например:

```
root
  +- gradleScript/
  |     dependencies.gradle
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
```

Затем все ваши зависимости могут быть расположены в `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}",
    ];
}
```

Который может быть применен из этого файла в файле верхнего уровня `build.gradle` следующим образом:

```
// Load dependencies
apply from: 'gradleScript/dependencies.gradle'
```

И В `module1/build.gradle` :

```
// Module build file
dependencies {
    // ...
    compile supportDependencies.appCompat
    compile supportDependencies.design
    compile firebaseDependencies.crash
}
```

Другой подход

Менее подробный подход для централизации версий зависимостей библиотек может быть достигнут путем объявления номера версии как переменной один раз и использования ее повсюду.

В рабочей области root `build.gradle` добавьте следующее:

```
ext.v = [
    supportVersion:'24.1.1',
```

```
]
```

И в каждом модуле, который использует одну и ту же библиотеку, добавьте необходимые библиотеки

```
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}"
```

Структура каталогов для ресурсов, специфичных для вкуса

Различные варианты создания приложений могут содержать разные ресурсы. Чтобы создать ресурс, специфичный для вкуса, создайте каталог с наименьшим именем вашего аромата в каталоге `src` и добавьте свои ресурсы таким же образом, как вы обычно.

Например, если у вас есть аромат `Development` и вы хотите предоставить отдельный значок запуска, вы должны создать каталог `src/development/res/drawable-mdpi` а внутри этого каталога создайте файл `ic_launcher.png` с вашим значком, зависящим от конкретной разработки.

Структура каталогов будет выглядеть так:

```
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'
```

(Конечно, в этом случае вы также создадите значки для drawable-hdpi, drawable-xhdpi и *т . Д.*).

Почему в проекте Android Studio есть два файла build.gradle?

`<PROJECT_ROOT>\app\build.gradle` специфичен для **модуля приложения** .

`<PROJECT_ROOT>\build.gradle` - это **«файл сборки верхнего уровня»,** где вы можете добавить параметры конфигурации, общие для всех подпроектов / модулей.

Если вы используете другой модуль в своем проекте, в качестве локальной библиотеки у вас будет другой файл `build.gradle` : `<PROJECT_ROOT>\module\build.gradle`

В файле верхнего уровня вы можете указать общие свойства как блок buildscript или некоторые общие свойства.

```
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"
}
```

В `app\build.gradle` вы определяете только свойства для модуля:

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


android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion
}

dependencies {
    //.....
}
```

Выполнение скрипта оболочки из градиента

Сценарий оболочки - это очень универсальный способ расширить вашу сборку до практически всего, что вы можете придумать.

В качестве примера, вот простой скрипт для компиляции файлов protobuf и добавления результирующих java-файлов в исходный каталог для дальнейшей компиляции:

```
def compilePb() {
    exec {
        // NOTICE: gradle will fail if there's an error in the protoc file...
        executable "../pbScript.sh"
    }
}

project.afterEvaluate {
    compilePb()
}
```

Сценарий оболочки «pbScript.sh» для этого примера, расположенный в корневой папке проекта:

```
#!/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
```

Отладка ошибок Gradle

Ниже приведена отрывок из Gradle. Что такое ненулевое значение выхода и как его исправить? , см. его для полного обсуждения.

Предположим, вы разрабатываете приложение, и вы получаете некоторую ошибку Gradle, которая появляется, как правило, будет выглядеть так.

```
: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
```

Вы ищете здесь, в StackOverflow, для своей проблемы, и люди говорят, чтобы очистить и перестроить ваш проект, или включить MultiDex , и когда вы попробуете это, это просто не устраняет проблему.

Есть способы получить дополнительную информацию , но сам вывод Gradle должен указывать на фактическую ошибку в нескольких строках над этим сообщением между: `module:someTask FAILED` и последним `:module:someOtherTask` который прошел. Поэтому, если вы зададите вопрос о своей ошибке, отредактируйте свои вопросы, чтобы включить больше контекста в ошибку.

Таким образом, вы получаете «ненулевое значение выхода». Ну, это число является хорошим показателем того, что вы должны попытаться исправить. Вот несколько наиболее часто встречающихся случаев.

- `1` является просто общим кодом ошибки, и ошибка, скорее всего, в выходе Gradle
- `2` похоже, связано с перекрывающимися зависимостями или неправильной конфигурацией проекта.
- `3` похоже, связано со слишком большим количеством зависимостей или проблемой памяти.

Общие решения для вышеизложенного (после попытки «Очистить и перестроить проект»):

- `1`

- Обратите внимание на указанную ошибку. Как правило, это ошибка времени компиляции, то есть некоторая часть кода в вашем проекте недопустима. Это включает в себя как XML, так и Java для Android-проекта.
- `2` и `3` - Многие ответы здесь дают вам возможность включить multidex . Хотя это может решить проблему, это, скорее всего, обходной путь. Если вы не понимаете, почему вы его используете (см. Ссылку), вам, вероятно, это не понадобится. Общие решения включают в себя сокращение чрезмерного использования зависимостей библиотек (например, от всех сервисов Google Play, когда вам нужно использовать только одну библиотеку, например, Карты или Вход в систему).

Указание различных идентификаторов приложений для типов и типов продуктов

Вы можете указать разные идентификаторы приложений или имена пакетов для каждого типа `buildType` или `productFlavor` используя атрибут конфигурации **applicationIdSuffix** :

Пример суффикса `applicationId` для каждого типа `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"
    }
}
```

В результате мы `applicationIds` :

- com.package.android для `release`
- com.package.android. **dev** для `development`
- com.package.android. **qa** для `testing`

Это можно сделать и для `productFlavors` :

```
productFlavors {
```

```
    free {
        applicationIdSuffix ".free"
    }
    paid {
        applicationIdSuffix ".paid"
    }
}
```

Результирующее `applicationIds` будет:

- com.package.android. **бесплатно** для `free` вкуса
- com.package.android. **заплатил** за `paid` аромат

Подписать APK без раскрытия пароля хранилища ключей

Вы можете определить конфигурацию подписи, чтобы подписать apk в файле `build.gradle` используя следующие свойства:

- `storeFile` : файл хранилища ключей
- `storePassword` : пароль хранилища ключей
- `keyAlias` : ключевое имя псевдонима
- `keyPassword` : пароль псевдонима ключа

Во многих случаях вам может потребоваться избежать такой информации в файле `build.gradle` .

Способ A: Настроить подпись выпуска с помощью файла keystore.properties.

Можно сконфигурировать `build.gradle` вашего приложения, чтобы он считывал вашу информацию о настройке `build.gradle` из файла свойств, такого как `keystore.properties` .

Настройка подписи так же выгодна, потому что:

- Ваша информация о настройке `build.gradle` отделена от файла `build.gradle`
- Вам не нужно вмешиваться во время процесса подписания, чтобы предоставить пароли для файла хранилища ключей
- Вы можете легко исключить файл `keystore.properties` из управления версиями

Во-первых, создайте файл с именем `keystore.properties` в корневом каталоге вашего проекта с таким содержимым (заменив значения собственными):

```
storeFile=keystore.jks
storePassword=storePassword
keyAlias=keyAlias
keyPassword=keyPassword
```

Теперь в файле `build.gradle` вашего приложения `signingConfigs` блок `signingConfigs` следующим образом:

```
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']
            }
        }
    }
}
```

Это действительно все, что нужно, **но не забудьте исключить как ваш файл `keystore.properties` файл `keystore.properties` из управления версиями** .

Несколько вещей, чтобы отметить:

- `storeFile` путь , указанный в `keystore.properties` файл должен быть относительно вашего приложения `build.gradle` файла. В этом примере предполагается, что файл хранилища ключей находится в том же каталоге, что и файл `build.gradle` приложения.
- В этом примере файл `keystore.properties` в корне проекта. Если вы поместите его в другое место, обязательно измените значение в `rootProject.file('keystore.properties')` на ваше местоположение по отношению к корню вашего проекта.

Метод B: используя переменную окружения

То же самое можно сделать и без файла свойств, что затрудняет поиск пароля:

```
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
    }
}
```

Переменная среды `"ps"` может быть глобальной, но более безопасный подход может заключаться в добавлении ее в оболочку Android Studio.
В linux это можно сделать, отредактировав `Desktop Entry` Studio `Desktop Entry`

```
Exec=sh -c "export ps=myPassword123 ; /path/to/studio.sh"
```

Вы можете найти более подробную информацию в этой теме .

Версии ваших сборников через файл «version.properties»

Вы можете использовать Gradle для автоматического увеличения вашей версии пакета каждый раз, когда вы его создаете. Для этого создайте файл `version.properties` в том же каталоге, что и ваш `build.gradle` со следующим содержимым:

```
VERSION_MAJOR=0
VERSION_MINOR=1
VERSION_BUILD=1
```

(Изменение значений для основного и второстепенного, как вы сочтете нужным). Затем в `build.gradle` добавьте следующий код в раздел `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)
  }
}
```

Информацию можно получить в Java в виде строки `BuildConfig.VERSION_NAME` для полного { major}. { `BuildConfig.VERSION_NAME` }. {Build} числа и как целое `BuildConfig.VERSION_CODE` только для номера сборки.

Изменение имени файла apk и добавление имени версии:

Это код для изменения имени файла прикладного приложения (.apk). Имя можно настроить, присвоив другому значению `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 + "]");
        }
    }
}
```

Отключить сжатие изображения для меньшего размера файла APK

Если вы оптимизируете все изображения вручную, отключите APT Cruncher для меньшего размера файла APK.

```
android {

    aaptOptions {
        cruncherEnabled = false
    }
}
```

Включить Proguard с помощью gradle

Для включения конфигураций Proguard для вашего приложения вам необходимо включить его в файл уровня градиента на уровне модуля. Вы должны установить значение `minifyEnabled true` .

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

```
    }
```

Вышеприведенный код применит ваши конфигурации Proguard, содержащиеся в стандартном Android SDK в сочетании с файлом «proguard-rules.pro» на вашем модуле, в ваш выпущенный apk.

Включить поддержку экспериментального NDK-плагина для Gradle и AndroidStudio

Включите и настройте экспериментальный плагин Gradle для улучшения поддержки NDK AndroidStudio. Убедитесь, что вы выполняете следующие требования:

- Gradle 2.10 (для этого примера)
- Android NDK r10 или новее
- Android SDK с инструментами построения v19.0.0 или новее

Настроить файл MyApp / build.gradle

Отредактируйте строку dependencies.classpath в файле build.gradle, например

```
classpath 'com.android.tools.build:gradle:2.1.2'
```

в

```
classpath 'com.android.tools.build:gradle-experimental:0.7.2'
```

(v0.7.2 была последней версией на момент написания. Проверьте последнюю версию самостоятельно и соответствующим образом адаптируйте свою линию)

Файл build.gradle должен выглядеть примерно так:

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle-experimental:0.7.2'
    }
}

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
```

```
}
```

Настроить файл MyApp / app / build.gradle

Отредактируйте файл build.gradle, чтобы он выглядел примерно так, как показано в следующем примере. Номера версий могут отличаться.

```
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'])
}
```

Синхронизируйте и проверьте, нет ли ошибок в файлах Gradle перед продолжением.

Проверить, включен ли плагин

Сначала убедитесь, что вы загрузили модуль Android NDK. Затем создайте новое приложение в AndroidStudio и добавьте следующее в файл ActivityMain:

```
public class MainActivity implements Activity {
    onCreate() {
        // Pregenerated code. Not important here
    }
    static {
        System.loadLibrary("myLib");
    }
    public static native String getString();
}
```

Часть `getString()` должна быть выделена красным цветом, говоря, что соответствующая функция JNI не найдена. Наведите указатель мыши на вызов функции, пока не появится красная лампочка. Щелкните лампу и выберите `create function JNI_...` Это должно сгенерировать файл myLib.c в каталоге myApp / app / src / main / jni с правильным вызовом функции JNI. Он должен выглядеть примерно так:

```
#include <jni.h>

JNIEXPORT jstring JNICALL
Java_com_example_mydomain_myapp_MainActivity_getString(JNIEnv *env, jobject instance)
{
    // TODO

    return (*env)->NewStringUTF(env, returnValue);
}
```

Если это не так, то плагин не был настроен правильно или NDK не был загружен

Показать все задачи проекта 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
```

Немедленно удалить «неприглаженный» apk

Если вам не нужны автоматически сгенерированные файлы apk с `unaligned` суффиксом (которого вы, вероятно, нет), вы можете добавить следующий код для файла `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()
        }
    }
  }
}
```

Отсюда

Игнорирование варианта сборки

По некоторым причинам вы можете игнорировать свои варианты сборки. Например: у вас есть «макет» вкуса продукта, и вы используете его только для целей отладки, таких как тестирование единиц / контрольно-измерительных приборов.

Давайте проигнорируем вариант **mockRelease** из нашего проекта. Откройте файл **build.gradle** и напишите:

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

Просмотр дерева зависимостей

Используйте зависимости задачи. В зависимости от того, как настроены ваши модули, это могут быть `./gradlew dependencies` или посмотреть зависимости использования приложения

модуля `./gradlew :app:dependencies`

Пример, следующий за файлом 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'
}
```

будет отображать следующий график:

```
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 (*)

. . .
```

Здесь вы можете увидеть, что проект напрямую включает в себя `com.android.support:design` версию 23.2.1, которая сама приносит `com.android.support:support-v4` с версией 23.2.1. Однако `com.google.android.gms:play-services` самих `com.google.android.gms:play-services` есть зависимость от той же `support-v4` но с более старой версией 21.0.0, которая является конфликтом, обнаруженным методом gradle.

`(*)` используются, когда gradle пропускает поддерево, потому что эти зависимости уже были указаны ранее.

Используйте gradle.properties для центральной версии номера / buildconfigurations

Вы можете определить центральную конфигурационную информацию в

- отдельный град включает файл. Централизация зависимостей через файл « dependencies.gradle»
- автономный файл свойств Версии ваших сборников через файл «version.properties»

или сделать это с корневым файлом `gradle.properties`

структура проекта

```
root
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
  +- gradle.properties
```

глобальная настройка для всех подмодулей в 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
```

использование в подмодуле

```
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 {
    ...
}
```

Примечание. Если вы хотите опубликовать свое приложение в магазине приложений F-Droid, вы должны использовать магические числа в файле gradle, потому что еще один робот f-droid не может читать текущую версию nnnner для обнаружения / проверки изменений версии.

Отображение информации о подписке

В некоторых случаях (например, при получении ключа API Google) вам нужно найти отпечаток своего хранилища ключей. У Gradle есть удобная задача, отображающая всю информацию о подписке, включая отпечатки ключей:

```
./gradlew signingReport
```

Это примерный результат:

```
: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
----------
```

Определение типов сборки

Вы можете создавать и настраивать типы сборки в файле `build.gradle` уровне `build.gradle` внутри блока `android {}` .

```
    android {
        ...
        defaultConfig {...}

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

```
            debug {
                applicationIdSuffix ".debug"
            }
        }
    }
```

Прочитайте Gradle для Android онлайн: https://riptutorial.com/ru/android/topic/95/gradle-для-android

глава 43: GreenBot EventBus

Синтаксис

- @Subscribe (threadMode = ThreadMode.POSTING) public void onEvent (событие EventClass) {}

параметры

Ветвь дискуссии	Описание
`ThreadMode.POSTING`	Будет вызван в тот же поток, что событие было опубликовано. Это режим "по умолчанию".
`ThreadMode.MAIN`	Будет вызван в основной поток пользовательского интерфейса.
`ThreadMode.BACKGROUND`	Будет вызван фоновый поток. Если поток проводки не является основным потоком, он будет использоваться. Если отправлено в основной поток, `EventBus` имеет один фоновый поток, который он будет использовать.
`ThreadMode.ASYNC`	Будет вызван в свою собственную нить.

Examples

Создание объекта Event

Для отправки и получения событий нам сначала нужен объект Event. Объекты событий - фактически простые POJO.

```
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;
    }
}
```

Получение событий

Для получения событий вам необходимо зарегистрировать свой класс на `EventBus` .

```
@Override
public void onStart() {
    super.onStart();
    EventBus.getDefault().register(this);
}

@Override
public void onStop() {
   EventBus.getDefault().unregister(this);
   super.onStop();
}
```

А затем подписаться на события.

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void handleEvent(ArbitaryEvent event) {
    Toast.makeText(getActivity(), "Event type: "+event.getEventType(),
Toast.LENGTH_SHORT).show();
}
```

Отправка событий

Отправка событий так же просто, как создание объекта Event, а затем его публикация.

```
EventBus.getDefault().post(new ArbitaryEvent(ArbitaryEvent.TYPE_1));
```

Прохождение простого события

Первое, что нам нужно сделать, добавить EventBus в файл градиента нашего модуля:

```
dependencies {
    ...
    compile 'org.greenrobot:eventbus:3.0.0'
    ...
}
```

Теперь нам нужно создать модель для нашего мероприятия. Он может содержать все, что мы хотим передать. Сейчас мы просто создадим пустой класс.

```
public class DeviceConnectedEvent
{
}
```

Теперь мы можем добавить код в нашу `Activity` которая будет регистрироваться в EventBus и подписаться на это событие.

```
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
    }
}
```

В этой `Activity` мы получаем экземпляр `EventBus` в `onCreate()` метода. Мы регистрируем / `onStart()` регистрацию событий в `onStart()` / `onStop()` . Важно помнить о том, чтобы отменить регистрацию, когда ваш слушатель теряет объем, или вы можете пропустить свою `Activity` .

Наконец, мы определяем метод, который мы хотим вызвать с событием. `@Subscribe` сообщает EventBus, какие методы он может искать для обработки событий. У вас должен быть хотя бы один метод, аннотированный с помощью `@Subscribe` для регистрации в EventBus, иначе он выдает исключение. В аннотации мы определяем режим потока. Это сообщает EventBus, какой поток вызывает метод. Это очень удобный способ передачи информации из фонового потока в поток пользовательского интерфейса! Это именно то, что мы делаем здесь. `ThreadMode.MAIN` означает, что этот метод будет вызываться в основном потоке пользовательского интерфейса Android, поэтому вам безопасно делать какие-либо манипуляции с пользовательским интерфейсом здесь, что вам нужно. Имя метода не имеет значения. `@Subscribe` только думать, что аннотация `@Subscribe` , которую ищет EventBus, является типом аргумента. Пока тип совпадает, он будет вызываться, когда событие отправлено.

Последнее, что нам нужно сделать, чтобы отправить событие. Этот код будет в нашей `Service` .

```
EventBus.getDefault().post(new DeviceConnectedEvent());
```

Вот и все! EventBus возьмет этот DeviceConnectedEvent и просмотрит зарегистрированные зарегистрированные слушатели, просмотрит методы, которые они подписали, и найдите те, которые принимают DeviceConnectedEvent в качестве аргумента, и вызовите их в потоке, который они хотят вызвать.

Прочитайте GreenBot EventBus онлайн: https://riptutorial.com/ru/android/topic/3551/greenbot-eventbus

глава 44: GreenDAO

Вступление

GreenDAO - это библиотека объектно-реляционного сопоставления, которая помогает разработчикам использовать базы данных SQLite для постоянного локального хранилища.

Examples

Вспомогательные методы для запросов SELECT, INSERT, DELETE, UPDATE

В этом примере показан вспомогательный класс, содержащий полезные методы при выполнении запросов для данных. Каждый метод использует Java Generic для того, чтобы быть очень гибким.

```
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();
}
```

Создание объекта с помощью GreenDAO 3.X, который имеет составной первичный ключ

При создании модели для таблицы с составным первичным ключом требуется дополнительная работа над объектом для модели Entity для соблюдения этих ограничений.

В следующем примере таблица SQL и Entity демонстрируют структуру для хранения обзора, оставленного клиентом для элемента в интернет-магазине. В этом примере мы хотим, `item_id` столбцы `customer_id` и `item_id` составным первичным ключом, позволяя использовать только один просмотр между конкретным клиентом и элементом.

Таблица 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)
);
```

Обычно мы будем использовать аннотации `@Id` и `@Unique` над соответствующими полями класса сущности, однако для составного первичного ключа мы делаем следующее:

1. Добавьте аннотацию `@Index` внутри аннотации `@Index` на уровне `@Entity` . Свойство value содержит список разделенных запятыми полей, составляющих ключ. Используйте `unique` свойство, как показано для обеспечения уникальности ключа.

2. GreenDAO требует, чтобы каждый объект Entity имел `long` или `Long` объект в качестве первичного ключа. Нам все равно нужно добавить это в класс Entity, однако нам не нужно его использовать или беспокоиться об этом, влияющем на нашу реализацию. В приведенном ниже примере он называется `localID`

сущность

```
@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() {}
}
```

Начало работы с GreenDao v3.X

После добавления зависимости библиотеки GreenDao и плагина Gradle нам нужно сначала создать объект сущности.

сущность

Сущность - это простой старый Java-объект *(POJO),* который моделирует некоторые данные в базе данных. GreenDao будет использовать этот класс для создания таблицы в базе данных SQLite и автоматически генерировать вспомогательные классы, которые мы можем использовать для доступа и хранения данных без необходимости писать SQL-запросы.

```
@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...

}
```

Одноразовая настройка GreenDao

Каждый раз, когда запускается приложение, GreenDao необходимо инициализировать. GreenDao предлагает сохранить этот код в классе Application или где-то он будет запускаться только один раз.

```
DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this, "mydatabase", null);
db = helper.getWritableDatabase();
DaoMaster daoMaster = new DaoMaster(db);
DaoSession daoSession = daoMaster.newSession();
```

Классы помощников GreenDao

После создания объекта сущности GreenDao автоматически создает вспомогательные классы, используемые для взаимодействия с базой данных. Они называются аналогично имени созданного объекта сущности, за которым следует `Dao` и извлекаются из объекта `daoSession` .

```
UsersDao usersDao = daoSession.getUsersDao();
```

Теперь можно выполнить множество типичных действий с базой данных с помощью этого объекта Dao с объектом сущности.

запрос

```
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();
```

Вставить

```
Users newUser = new User("John","Doe","jdoe@example.com");
usersDao.insert(newUser);
```

Обновить

```
// Modify a previously retrieved user object and update
user.setLastname("Dole");
usersDao.update(user);
```

удалять

```
// Delete a previously retrieved user object
usersDao.delete(user);
```

Прочитайте GreenDAO онлайн: https://riptutorial.com/ru/android/topic/1345/greendao

глава 45: Gson

Вступление

Gson - это библиотека Java, которая может быть использована для преобразования объектов Java в их представление JSON. Г-н считает эти цели очень важными.

Особенности Gson:

Предоставьте простые `toJson()` и `fromJson()` для преобразования объектов Java в JSON и наоборот

Разрешить преобразование ранее немодифицируемых объектов в JSON и обратно

Обширная поддержка Java Generics

Поддержка произвольно сложных объектов (с глубокими иерархиями наследования и широким использованием родовых типов)

Синтаксис

- Исключительное исключение ()
- Поле FieldNamingStrategyNamingStrategy ()
- <T> T fromJson (JsonElement json, класс <T> classOfT)
- <T> T fromJson (JsonElement json, Тип typeOfT)
- <T> T от Json (читатель JsonReader, тип typeOfT)
- <T> T fromJson (Reader json, Class <T> classOfT)
- <T> T fromJson (Reader json, Тип typeOfT)
- <T> T fromJson (String json, Class <T> classOfT)
- <T> T fromJson (String json, Тип typeOfT)
- <T> Тип Adapter <T> getAdapter (тип <T>)
- <T> TypeAdapter <T> getAdapter (тип TypeToken <T>)
- <T> TypeAdapter <T> getDelegateAdapter (TypeAdapterFactory skipPast, TypeToken <T>)
- JsonReader newJsonReader (считыватель чтения)
- JsonWriter newJsonWriter (писатель-писатель)
- JsonElement toJsonTree (Object src)
- JsonElement toJsonTree (объект src, тип typeOfSrc)
- boolean serializeNulls ()
- boolean htmlSafe ()
- Строка toJson (JsonElement jsonElement)
- Строка toJson (Object src)
- Строка toJson (Object src, Тип typeOfSrc)
- Строка toString ()

- void toJson (Object src, Тип typeOfSrc, Добавочный писатель)
- void toJson (Object src, Тип typeOfSrc, JsonWriter writer)
- void toJson (JsonElement jsonElement, Appendable writer)
- void toJson (JsonElement jsonElement, автор JsonWriter)
- void toJson (Object src, Appendable writer)

Examples

Разбор JSON с Gson

В этом примере показан разбор объекта JSON с использованием библиотеки Gson от Google .

Разбор объектов:

```
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 = "";
    }


}
```

Затем, когда необходимо провести синтаксический анализ, используйте следующее:

```
String robotJson = "{
                        \"version\": \"JellyBean\",
                        \"age\": 3,
                        \"robotName\": \"Droid\"
                    }";

Gson gson = new Gson();
Robot robot = gson.fromJson(robotJson, Robot.class);
```

Разбор списка:

Когда вы извлекаете список объектов JSON, вам часто приходится анализировать их и

преобразовывать в объекты Java.

Строка JSON, которую мы попытаемся преобразовать, следующая:

```
{
  "owned_dogs": [
    {
      "name": "Ron",
      "age": 12,
      "breed": "terrier"
    },
    {
      "name": "Bob",
      "age": 4,
      "breed": "bulldog"
    },
    {
      "name": "Johny",
      "age": 3,
      "breed": "golden retriever"
    }
  ]
}
```

Этот конкретный массив JSON содержит три объекта. В нашем Java-коде мы хотим сопоставить эти объекты с объектами `Dog` . Объект Dog будет выглядеть так:

```
private class Dog {
    public String name;
    public int age;

    @SerializedName("breed")
    public String breedName;
}
```

Чтобы преобразовать массив JSON в `Dog[]` :

```
Dog[] arrayOfDogs = gson.fromJson(jsonArrayString, Dog[].class);
```

Преобразование `Dog[]` в строку JSON:

```
String jsonArray = gson.toJson(arrayOfDogs, Dog[].class);
```

Чтобы преобразовать массив JSON в `ArrayList<Dog>` мы можем сделать следующее:

```
Type typeListOfDogs = new TypeToken<List<Dog>>(){}.getType();
List<Dog> listOfDogs = gson.fromJson(jsonArrayString, typeListOfDogs);
```

Тип объекта `typeListOfDogs` определяет, как будет выглядеть список объектов `Dog` . GSON может использовать этот объект типа для сопоставления массива JSON с правильными значениями.

Альтернативно, преобразование `List<Dog>` в массив JSON может быть выполнено аналогичным образом.

```
String jsonArray = gson.toJson(listOfDogs, typeListOfDogs);
```

Разбор свойств JSON для перечисления с помощью Gson

Если вы хотите разобрать String для перечисления с помощью Gson:

> {"status": "open"}

```
public enum Status {
    @SerializedName("open")
    OPEN,
    @SerializedName("waiting")
    WAITING,
    @SerializedName("confirm")
    CONFIRM,
    @SerializedName("ready")
    READY
}
```

Разбор списка с Гсоном

Способ 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());
```

Это полезно для большинства общих классов контейнеров, поскольку вы не можете получить класс параметризованного типа (т. Е. Вы не можете вызвать `List<String>.class`).

Способ 2

```
public class StringList extends ArrayList<String> { }

...

List<String> members = gson.fromJson(json, StringList.class);
```

Кроме того, вы всегда можете подклассифицировать нужный тип, а затем пройти в этом классе. Однако это не всегда лучшая практика, так как она вернет вам объект типа `StringList` ;

Сериализация / десериализация JSON с помощью AutoValue и Gson

Импортировать в корневой файл gradle

```
classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
```

Импорт в файл приложения 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'
```

Создать объект с autovalue:

```
@AutoValue public abstract class SignIn {
    @SerializedName("signin_token") public abstract String signinToken();
    public abstract String username();

    public static TypeAdapter<SignIn> typeAdapter(Gson gson) {
        return new AutoValue_SignIn.GsonTypeAdapter(gson);
    }

    public static SignIn create(String signin, String username) {
        return new AutoValue_SignIn(signin, username);
    }
}
```

Создайте свой конвертер Gson с помощью вашего GsonBuilder

```
Gson gson = new GsonBuilder()
                .registerTypeAdapterFactory(
                    new AutoValueGsonTypeAdapterFactory())
                .create());
```

Deserialize

```
String myJsonData = "{
    \"signin_token\": \"mySigninToken\",
    \"username\": \"myUsername\" }";
SignIn signInData = gson.fromJson(myJsonData, Signin.class);
```

Сериализация

```
Signin myData = SignIn.create("myTokenData", "myUsername");
String myJsonData = gson.toJson(myData);
```

Использование Gson - отличный способ упростить код Serialization и Deserialization, используя объекты POJO. Побочным эффектом является то, что отражение является дорогостоящим. Вот почему использование AutoValue-Gson для генерации CustomTypeAdapter позволит избежать этой стоимости отражения, оставаясь очень простым для обновления при изменении api.

Разбор JSON для универсального класса с Gson

Предположим, что у нас есть строка JSON:

```
["first","second","third"]
```

Мы можем проанализировать эту строку JSON в массив `String` :

```
Gson gson = new Gson();
String jsonArray = "[\"first\",\"second\",\"third\"]";
String[] strings = gson.fromJson(jsonArray, String[].class);
```

Но если мы хотим разобрать его в объект `List<String>` , мы должны использовать `TypeToken` .

Вот пример:

```
Gson gson = new Gson();
String jsonArray = "[\"first\",\"second\",\"third\"]";
List<String> stringList = gson.fromJson(jsonArray, new TypeToken<List<String>>()
{}.getType());
```

Предположим, что у нас есть два класса:

```
public class Outer<T> {
    public int index;
    public T data;
}

public class Person {
    public String firstName;
    public String lastName;
}
```

и у нас есть строка JSON, которая должна анализироваться на объект `Outer<Person>` .

В этом примере показано, как разбить эту строку JSON на связанный общий объект класса:

```
String json = "......";
Type userType = new TypeToken<Outer<Person>>(){}.getType();
Result<User> userResult = gson.fromJson(json,userType);
```

Если строка JSON должна анализироваться на объект `Outer<List<Person>>` :

```
Type userListType = new TypeToken<Outer<List<Person>>>(){}.getType();
Result<List<User>> userListResult = gson.fromJson(json,userListType);
```

Добавление Gson к вашему проекту

```
dependencies {
    compile 'com.google.code.gson:gson:2.8.1'
}
```

Использовать последнюю версию Gson

Следующая строка будет компилировать последнюю версию gson-библиотеки каждый раз, когда вы компилируете, вам не нужно менять версию.

Плюсы: вы можете использовать новейшие функции, скорость и меньше ошибок.
Минусы: он может нарушить совместимость с вашим кодом.

```
compile 'com.google.code.gson:gson:+'
```

Использование Gson для загрузки JSON-файла с диска.

Это загрузит JSON-файл с диска и преобразует его в заданный тип.

```
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);
}
```

Добавление пользовательского конвертера в Gson

Иногда вам нужно сериализовать или десериализовать некоторые поля в нужном формате, например, ваш бэкэнд может использовать формат «YYYY-MM-dd HH: mm» для дат, и вы хотите, чтобы ваш POJOS использовал класс DateTime в Joda Time.

Чтобы автоматически преобразовать эти строки в объект DateTimes, вы можете использовать собственный конвертер.

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

Чтобы заставить Gson использовать вновь созданный конвертер, вам нужно назначить его при создании объекта 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
```

Чтобы десериализовать дату в этом формате, вам нужно определить поле в формате DateTime:

```
public class SomePojo {
    private DateTime someDate;
}
```

Когда Gson встречает поле типа DateTime, он вызовет ваш конвертер, чтобы десериализовать поле.

Использование Gson в качестве сериализатора с дооснащением

Прежде всего вам нужно добавить `GsonConverterFactory` в файл build.gradle

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

Затем вы должны добавить фабрику преобразователя при создании Retrofit Service:

```
Gson gson = new GsonBuilder().create();
new Retrofit.Builder()
        .baseUrl(someUrl)
        .addConverterFactory(GsonConverterFactory.create(gson))
        .build()
        .create(RetrofitService.class);
```

Вы можете добавить собственные преобразователи при создании объекта Gson, который вы передаете на заводе. Позволяет создавать персонализированные преобразования

типов.

Разбор массива json для универсального класса с использованием Gson

Предположим, что у нас есть 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"
    }
  ]
}
```

Мы можем проанализировать этот массив в пользовательский твиты (содержимое списка твитов), но это проще сделать с `fromJson` метода `fromJson` :

```
Gson gson = new Gson();
String jsonArray = "....";
Tweets tweets = gson.fromJson(jsonArray, Tweets.class);
```

Предположим, что у нас есть два класса:

```
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;
```

```
}
```

и если вам нужно просто разобрать json-массив, вы можете использовать этот код в своем разборе:

```
String tweetsJsonArray = "[{.....},{.....}]"
List<Tweet> tweets = gson.fromJson(tweetsJsonArray, new TypeToken<List<Tweet>>()
{}.getType());
```

Пользовательский десериализатор JSON с использованием Gson

Представьте, что у вас есть все даты во всех ответах в каком-то специальном формате, например `/Date(1465935152)/` и вы хотите применить общее правило для десериализации всех дат Json для экземпляров java `Date` . В этом случае вам нужно реализовать пользовательский Json Deserializer.

Пример json:

```
{
  "id": 1,
  "created_on": "Date(1465935152)",
  "updated_on": "Date(1465968945)",
  "name": "Oleksandr"
}
```

Предположим, что этот класс ниже:

```
class User {
    @SerializedName("id")
    long id;
    @SerializedName("created_on")
    Date createdOn;
    @SerializedName("updated_on")
    Date updatedOn;
    @SerializedName("name")
    String name;
}
```

Пользовательский десериализатор:

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

И использование:

```
Gson gson = new GsonBuilder()
                .registerTypeAdapter(Date.class, new DateDeSerializer())
                .create();
String json = "....";
User user = gson.fromJson(json, User.class);
```

Сериализация и десериализация строк Jackson JSON с типами даты

Это также относится к случаю, когда вы хотите сделать преобразование Gson Date совместимым с Jackson, например.

Джексон обычно сериализует Date на «миллисекунды с эпохи», тогда как Gson использует читаемый формат, такой как `Aug 31, 2016 10:26:17` для представления Date. Это приводит к JsonSyntaxExceptions в Gson, когда вы пытаетесь десериализовать формат даты в формате Jackson.

Чтобы обойти это, вы можете добавить собственный сериализатор и собственный десериализатор:

```
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();
```

Использование Gson с наследованием

Gson не поддерживает наследование из коробки.

Допустим, у нас есть следующая иерархия классов:

```
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;
    }
}
```

И теперь мы хотим сериализовать экземпляр `DerivedClass1` в строку JSON

```
DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;

Gson gson = new Gson();
String derivedClass1Json = gson.toJson(derivedClass1);
```

Теперь, в другом месте, мы получаем эту строку json и хотим ее десериализовать, но во время компиляции мы знаем только, что это должен быть экземпляр `BaseClass` :

```
BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());
```

Но GSON не знает, что `derivedClass1Json` был первоначально экземпляром `DerivedClass1` , поэтому он будет распечатывать 10.

Как это решить?

Вам нужно создать собственный `JsonDeserializer` , который обрабатывает такие случаи. Решение не совсем чистое, но я не мог придумать лучшего.

Сначала добавьте следующее поле в базовый класс

```
@SerializedName("type")
private String typeName;
```

И инициализируйте его в конструкторе базового класса

```
public BaseClass() {
    typeName = getClass().getName();
}
```

Теперь добавьте следующий класс:

```
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);
 }
}
```

Все, что осталось сделать, это все перехватить -

```
GsonBuilder builder = new GsonBuilder();
 builder
 .registerTypeAdapter(BaseClass.class, new JsonDeserializerWithInheritance<BaseClass>());
 Gson gson = builder.create();
```

И теперь, запустив следующий код,

```
 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());
```

Распечатает 5.

Прочитайте Gson онлайн: https://riptutorial.com/ru/android/topic/4158/gson

глава 46: HttpURLConnection

Синтаксис

- аннотация void disconnect ()
- abstract boolean usingProxy ()
- static boolean getFollowRedirects ()
- static void setFollowRedirects (boolean set)
- Строка getHeaderField (int n)
- Строка getHeaderFieldKey (int n)
- Строка getRequestMethod ()
- Строка getResponseMessage ()
- int getResponseCode ()
- long getHeaderFieldDate (String name, long Default)
- boolean getInstanceFollowRedirects ()
- Разрешение getPermission ()
- InputStream getErrorStream ()
- void setChunkedStreamingMode (int chunklen)
- void setFixedLengthStreamingMode (int contentLength)
- void setFixedLengthStreamingMode (long contentLength)
- void setInstanceFollowRedirects (boolean followRedirects)
- void setRequestMethod (метод String)

замечания

`HttpURLConnection` - это стандартный HTTP-клиент для Android, используемый для отправки и получения данных через Интернет. Это конкретная реализация URLConnection для HTTP (RFC 2616).

Examples

Создание HttpURLConnection

Чтобы создать новый HTTP HTTP Client `HttpURLConnection` , вызовите `openConnection()` в экземпляре URL. Поскольку `openConnection()` возвращает `URLConnection` , вам необходимо явно `openConnection()` возвращаемое значение.

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
// do something with the connection
```

Если вы создаете новый `URL` , вам также придется обрабатывать исключения, связанные с

разбором URL-адресов.

```
try {
    URL url = new URL("http://example.com");
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    // do something with the connection
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

Когда тело ответа прочитано и соединение больше не требуется, соединение должно быть закрыто вызовом функции `disconnect()` .

Вот пример:

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
try {
    // do something with the connection
} finally {
    connection.disconnect();
}
```

Отправка запроса HTTP GET

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

try {
    BufferedReader br = new BufferedReader(new
InputStreamReader(connection.getInputStream()));

    // read the input stream
    // in this case, I simply read the first line of the stream
    String line = br.readLine();
    Log.d("HTTP-GET", line);

} finally {
    connection.disconnect();
}
```

Обратите внимание, что исключения не обрабатываются в приведенном выше примере. Полный пример, включая (тривиальную) обработку исключений, будет:

```
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();
    }
}
```

Чтение тела запроса HTTP GET

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

try {
    BufferedReader br = new BufferedReader(new
InputStreamReader(connection.getInputStream()));

    // use a string builder to bufferize the response body
    // read from the input strea.
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line).append('\n');
    }

    // use the string builder directly,
    // or convert it into a String
    String body = sb.toString();

    Log.d("HTTP-GET", body);

} finally {
    connection.disconnect();
}
```

Обратите внимание, что исключения не обрабатываются в приведенном выше примере.

Использовать HttpURLConnection для multipart / form-data

Создание пользовательского класса для вызова запроса MttpURLConnection 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;
    }
}
```

Используйте его (Async)

```
    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;
                }
```

Отправка запроса HTTP POST с параметрами

Используйте HashMap для хранения параметров, которые должны быть отправлены на сервер через параметры POST:

```
HashMap<String, String> params;
```

После заполнения `params` HashMap создайте StringBuilder, который будет использоваться для отправки их на сервер:

```
    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++;
    }
```

Затем создайте HttpURLConnection, откройте соединение и отправьте параметры 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();
}
```

Затем получите результат, который сервер отправляет обратно:

```
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();
    }
}
```

Загрузить (POST) файл с использованием HttpURLConnection

Довольно часто необходимо отправить / загрузить файл на удаленный сервер, например изображение, видео, аудио или резервную копию базы данных приложения на удаленный частный сервер. Предполагая, что сервер ожидает запроса POST с контентом, вот простой пример того, как выполнить эту задачу в Android.

Загрузка файлов отправляется с использованием POST-запросов `multipart/form-data` . Это очень легко реализовать:

```
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
}
```

Конечно, исключения должны быть пойманы или объявлены как брошенные. Пара указывает на этот код:

1. `postTarget` - это целевой URL-адрес POST; `oauthToken` - токен аутентификации; `fileDescription` - это описание файла, которое отправляется как значение `description` поля; `file` - это файл, который нужно отправить - он имеет тип `java.io.File` - если у вас есть путь к файлу, вы можете использовать `new File(filePath)` .
2. Он устанавливает заголовок `Authorization` для аутентификации oAuth
3. Он использует Apache Common `FileUtil` для чтения файла в массив байтов - если у вас уже есть содержимое файла в байтовом массиве или каким-то другим способом в памяти, тогда нет необходимости его читать.

Многоцелевой класс HttpURLConnection для обработки всех типов HTTP-запросов

Следующий класс может использоваться как один класс, который может обрабатывать `GET` , `POST` , `PUT` , `PATCH` и другие запросы:

```
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();
    }
}
```

использование

Используйте любой из заданных конструкторов класса в зависимости от того, нужно ли отправлять данные `POST` или любые дополнительные заголовки.

Метод `onComplete()` вызывается, когда выборка данных завершена. Данные возвращаются как объект класса `APIResponseObject` , который имеет код состояния, содержащий код состояния HTTP для запроса и строку, содержащую ответ. Вы можете проанализировать этот ответ в своем классе, то есть XML или JSON.

Вызовите `execute()` для объекта класса для выполнения запроса, как показано в следующем примере:

```
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();
}
```

Прочитайте HttpURLConnection онлайн:
https://riptutorial.com/ru/android/topic/781/httpurlconnection

глава 47: ImageView

Вступление

ImageView (`android.widget.ImageView`) представляет собой представление для отображения и управления ресурсами изображения, такими как Drawables и Bitmaps.

Некоторые эффекты, обсуждаемые в этом разделе, могут быть применены к изображению. Источник изображения может быть установлен в файле XML (папке `layout`) или программно в коде Java.

Синтаксис

- Метод `setImageResource(int resId)` устанавливает `setImageResource(int resId)` в качестве содержимого этого `ImageView` .
- **Использование:** `imageView.setImageResource(R.drawable.anyImage)`

параметры

параметр	Описание
`resId`	ваше имя файла изображения в папке `res` (обычно в папке с возможностью `drawable`)

Examples

Установить ресурс изображения

```
<ImageView
 android:id="@+id/imgExample"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 ...
 />
```

установить в качестве содержимого `ImageView` атрибут XML с атрибутом XML:

```
android:src="@drawable/android2"
```

установить программно:

```
 ImageView imgExample = (ImageView) findViewById(R.id.imgExample);
```

```
imgExample.setImageResource(R.drawable.android2);
```

Установить альфа

«alpha» используется для указания непрозрачности для изображения.

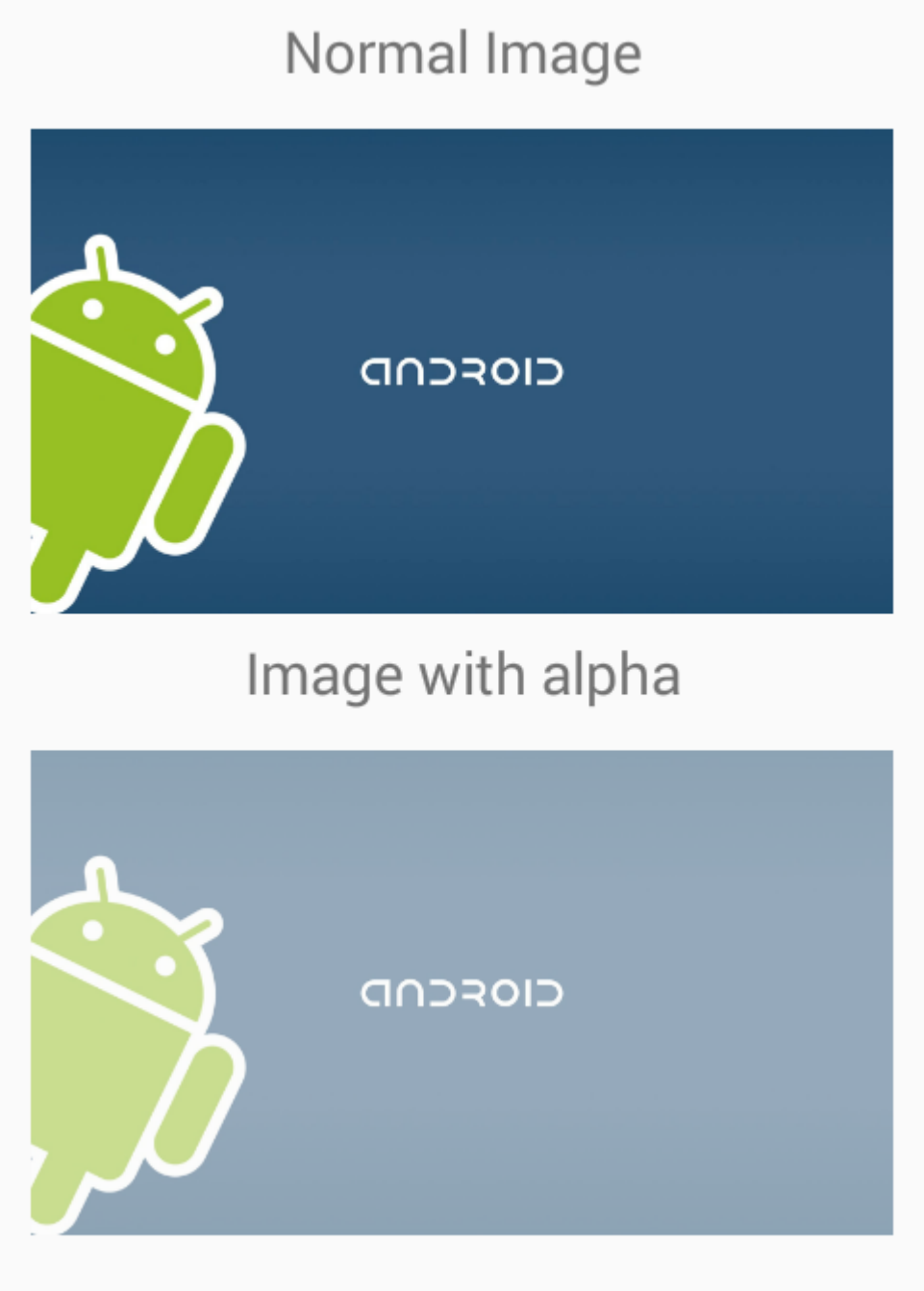
установить альфа, используя атрибут XML:

```
android:alpha="0.5"
```

Примечание: принимает значение float от 0 (прозрачное) до 1 (полностью видимое)

установить букву программно:

```
imgExample.setAlpha(0.5f);
```



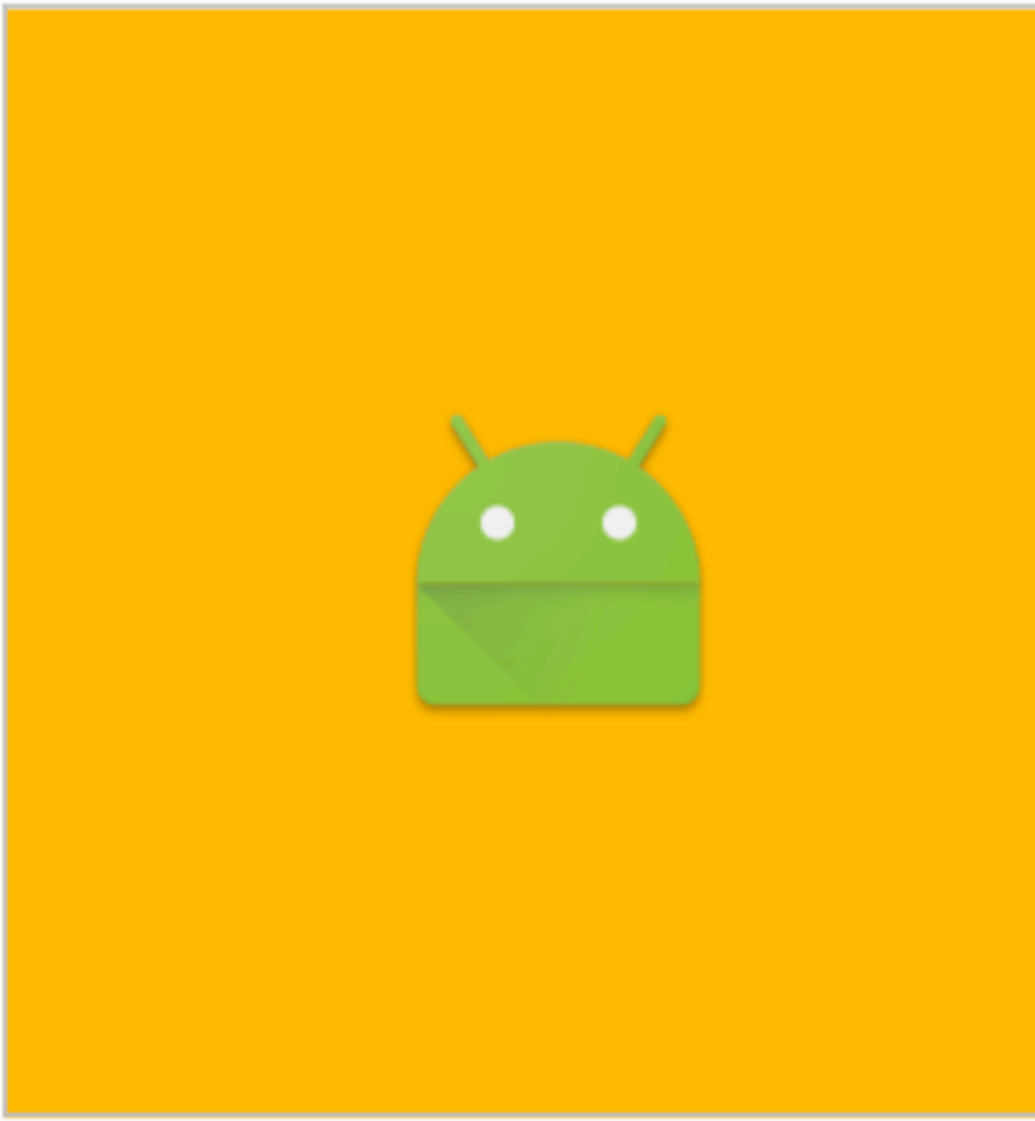
ImageView ScaleType - Центр

Изображение, содержащееся в ImageView, может не соответствовать точному размеру, указанному в контейнере. В этом случае структура позволяет изменять размер изображения несколькими способами.

Центр

```
    <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"/>
```

Это не изменит размер изображения, и оно центрирует его внутри контейнера *(оранжевый = контейнер)*



меньше изображения, изображение не будет изменяться, и вы сможете увидеть его часть

сильный текст

ImageView ScaleType - CenterCrop

Равномерно масштабируйте изображение (поддерживайте соотношение сторон изображения), чтобы обе размеры (ширина и высота) изображения были равны или больше соответствующего размера представления (минус заполнение).

Официальные документы

Когда изображение соответствует пропорциям контейнера:

`ImageView` .

XML-атрибут:

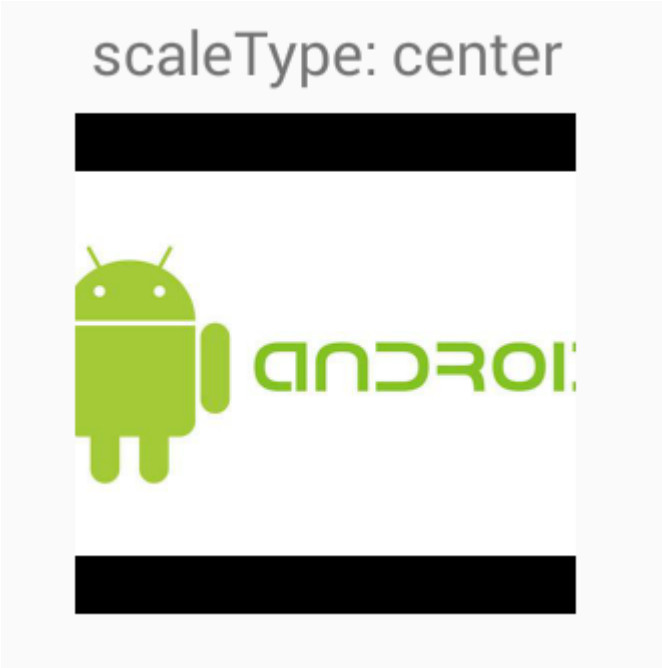
```
android:scaleType="..."
```

я проиллюстрирую различные типы шкал с квадратным `ImageView` который имеет черный фон, и мы хотим отобразить прямоугольный чертеж на белом фоне в `ImageView` .

```
<ImageView
 android:id="@+id/imgExample"
 android:layout_width="200dp"
 android:layout_height="200dp"
 android:background="#000"
 android:src="@drawable/android2"
 android:scaleType="..."/>
```

scaleType должен быть одним из следующих значений:

1. `center` : центрируйте изображение в представлении, но не масштабируйте.



2. `centerCrop` : масштабируйте изображение равномерно (поддерживайте пропорции изображения), чтобы оба размера (ширина и высота) изображения были равны или больше соответствующего размера представления (минус заполнение). Изображение затем центрируется в представлении.

3. `centerInside` : равномерно масштабируйте изображение (поддерживайте соотношение сторон изображения), чтобы обе размеры (ширина и высота) изображения были равны или меньше соответствующего размера представления (минус заполнение). Изображение затем центрируется в представлении.

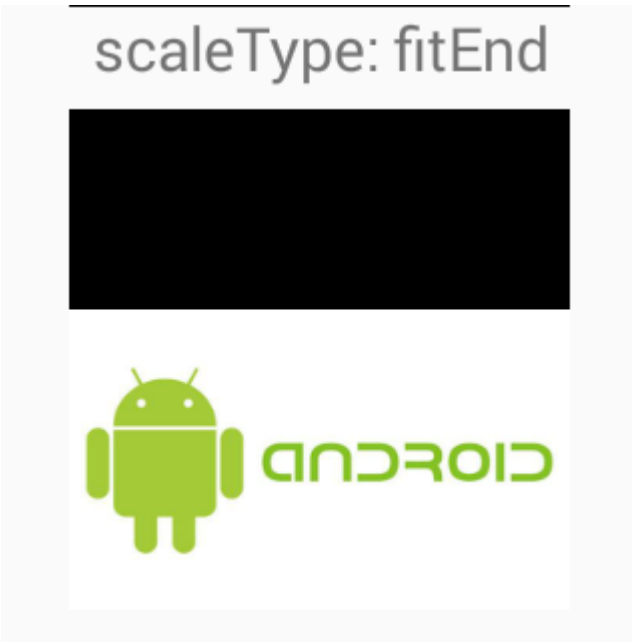
4. `matrix` : Масштабирование с использованием матрицы изображения при рисовании.

5. `fitXY` : масштабируйте изображение, используя FILL .

6. `fitStart` : масштабируйте изображение с помощью START .

7. `fitCenter` : масштабируйте изображение с помощью CENTER .

8. `fitEnd` : масштабируйте изображение с помощью END .



Установить оттенок

Установите цвет тонирования для изображения. По умолчанию оттенок будет смешиваться с использованием режима `SRC_ATOP` .

задать оттенок с использованием атрибута XML:

```
android:tint="#009c38"
```

Примечание. Должно быть значение цвета в виде `"#rgb"` , `"#argb"` , `"#rrggbb"` или `"#aarrggbb"` .

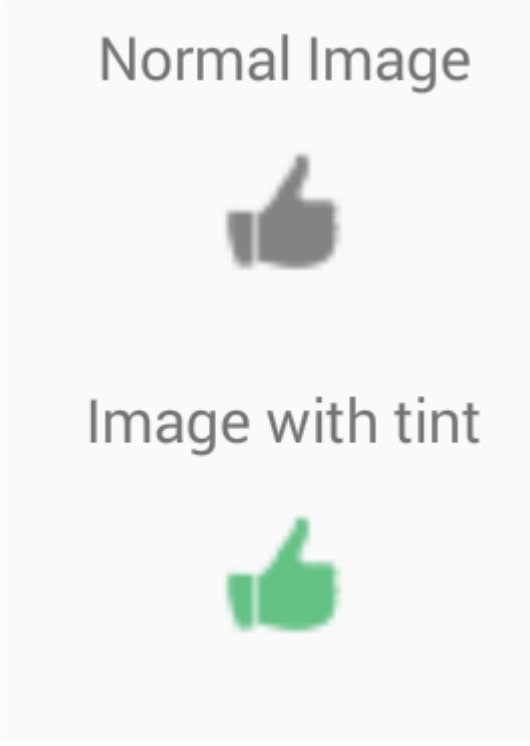
установить оттенок программно:

```
imgExample.setColorFilter(Color.argb(255, 0, 156, 38));
```

и вы можете очистить этот цветной фильтр:

```
imgExample.clearColorFilter();
```

Пример:



MLRoundedImageView.java

Скопируйте и вставьте следующий класс в свой пакет:

```
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;
    }
}
```

Используйте этот класс в XML с именем пакета вместо `ImageView`

```
<com.androidbuts.example.MLRoundedImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />
```

Прочитайте ImageView онлайн: https://riptutorial.com/ru/android/topic/4709/imageview

глава 48: IntentService

Синтаксис

4. <service android: name = ". UploadS3IntentService" android: exported = "false" />

замечания

`IntentService` предоставляет простой способ разгрузить работу в фоновом потоке. Он обрабатывает все о получении запросов, помещает их в очередь, останавливает себя и т. Д. Для вас. Его также легко реализовать, что делает его идеальным для использования, когда у вас есть трудоемкие операции, которые не входят в поток Main (UI).

Examples

Создание IntentService

Чтобы создать IntentService, создайте класс, который расширяет `IntentService` и внутри него, метод, который переопределяет `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.
     }
}
```

Образец Намерения

Вот пример `IntentService` который претендует на загрузку изображений в фоновом режиме. Все, что вам нужно сделать для реализации `IntentService` - это предоставить конструктор, который вызывает конструктор `super(String)` , и вам нужно реализовать метод `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
    }
}
```

Чтобы запустить `IntentService` , вам необходимо отправить `Intent` . Вы можете сделать это из `Activity` , например. Конечно, вы не ограничиваетесь этим. Ниже приведен пример того, как вы вызовите новую `Service` из класса `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` обрабатывает данные из своего `Intent` , так что вы можете отправлять несколько `Intent` s, не беспокоясь о том, столкнутся ли они друг с другом. `Intent` обрабатывается только одно `Intent` , остальные идут в очередь. Когда все задания будут завершены, `IntentService` автоматически отключится.

Основной пример IntentService

Абстрактный класс `IntentService` является базовым классом для сервисов, которые работают в фоновом режиме без какого-либо пользовательского интерфейса. Поэтому, чтобы обновить пользовательский интерфейс, мы должны использовать приемник, который может быть либо `BroadcastReceiver` либо `ResultReceiver` :

- `BroadcastReceiver` следует использовать, если вашему сервису необходимо обмениваться данными с несколькими компонентами, которые хотят прослушивать связь.
- `ResultReceiver` : должен использоваться, если вашему сервису необходимо связываться только с родительским приложением (т. `ResultReceiver` вашим приложением).

В `IntentService` у нас есть один ключевой метод `onHandleIntent()` , в котором мы будем делать все действия, например, подготовку уведомлений, создание аварийных сигналов и

т. Д.

Если вы хотите использовать собственный `IntentService` , вы должны расширить его следующим образом:

```
public class YourIntentService extends IntentService {
    public YourIntentService () {
        super("YourIntentService ");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        // TODO: Write your own code here.
    }
}
```

Вызов / запуск активности можно выполнить следующим образом:

```
Intent i = new Intent(this, YourIntentService.class);
startService(i);  // For the service.
startActivity(i); // For the activity; ignore this for now.
```

Подобно любой деятельности, вы можете передать дополнительную информацию, такую как данные пакета, следующим образом:

```
Intent passDataIntent = new Intent(this, YourIntentService.class);
msgIntent.putExtra("foo","bar");
startService(passDataIntent);
```

Теперь предположим, что мы передали некоторые данные в класс `YourIntentService` . Исходя из этих данных, действие может быть выполнено следующим образом:

```
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.
        }
    }
}
```

В приведенном выше коде также показано, как обрабатывать ограничения в `OnHandleIntent()` .

Прочитайте IntentService онлайн: https://riptutorial.com/ru/android/topic/5319/intentservice

глава 49: Java на Android

Вступление

Android поддерживает все языковые функции Java 7 и набор функций языка Java 8, которые различаются по версии платформы. На этой странице описываются новые возможности языка, которые вы можете использовать, как правильно настроить проект для их использования и любые известные проблемы, с которыми вы можете столкнуться.

Examples

Java 8 имеет подмножество с Retrolambda

Retrolambda позволяет запускать код Java 8 с помощью лямбда-выражений, ссылок на методы и операторов try-with-resources на Java 7, 6 или 5. Это делается путем преобразования вашего скомпилированного байт-кода Java 8, чтобы он мог работать в более старой среде выполнения Java.

Особенности языка бэкпорта:

- Лямбда-выражения передаются обратно, превращая их в анонимные внутренние классы. Это включает в себя оптимизацию использования экземпляра singleton для выражений лямбда без учета состояния, чтобы избежать повторного распределения объектов. Ссылки на методы - это, по сути, только синтаксический сахар для лямбда-выражений, и они обращаются таким же образом.
- Операторы Try-with-resources передаются обратно, удаляя вызовы `Throwable.addSuppressed` если целевая версия байт-кода находится ниже Java 7. Если вы хотите, чтобы исключенные исключения были записаны вместо проглатывания, создайте запрос функции, и мы сделаем это настраивается.
- `Objects.requireNonNull` вызовы заменяются вызовами `Object.getClass` если целевая версия байт-кода находится ниже Java 7. Синтетические проверки на null, сгенерированные JDK 9, используют `Objects.requireNonNull` , тогда как в предыдущих версиях JDK использовался `Object.getClass` .
- Дополнительно также:

    1. Способы по умолчанию возвращаются путем копирования методов по умолчанию в класс сопутствующего класса (имя интерфейса + «$») в качестве статических методов, заменяя методы по умолчанию в интерфейсе абстрактными методами и добавляя необходимые реализации методов ко всем классам, реализующим этот интерфейс ,

2. Статические методы на интерфейсах возвращаются путем перемещения статических методов в класс компаньона (имя интерфейса + «$») и путем изменения всех вызовов методов для вызова нового расположения метода.

Известные ограничения:

- Не поддерживает API-интерфейсы Java 8
- Backporting методы по умолчанию и статические методы на интерфейсах требуют, чтобы все backported интерфейсы и все классы, которые их реализовали, или вызывали их статические методы для резервного копирования вместе с одним выполнением Retrolambda. Другими словами, вы всегда должны делать чистую сборку. Кроме того, методы backporting default не будут работать через границы модулей или зависимостей.
- Может разорваться, если будущая сборка JDK 8 перестанет генерировать новый класс для каждого `invokedynamic` вызова. Retrolambda работает так, что он захватывает байт-код, который `java.lang.invoke.LambdaMetafactory` генерируется динамически, поэтому оптимизация этого механизма может сломать Retrolambda.

Retrolambda gradle plugin автоматически создаст ваш проект Android с Retrolambda. Последнюю версию можно найти на странице выпусков .

Использование:

1. Загрузите и установите jdk8
2. Добавьте в свой `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
    }
}
```

Известные вопросы:

- Lint терпит неудачу в java-файлах, которые имеют lambdas. Волокна Android не понимает синтаксис java 8 и терпит неудачу молча или громко. В настоящее время существует экспериментальная версия, которая устраняет проблему.
- Использование служб Google Play приводит к сбою Retrolambda. Версия 5.0.77 содержит байт-код, который несовместим с Retrolambda. Это должно быть исправлено в новых версиях игровых сервисов, если вы можете их обновить, это должно быть предпочтительным решением. Чтобы обойти эту проблему, вы можете использовать более раннюю версию, например 4.4.52, или добавить `-noverify` в jvm args.

```
retrolambda {
  jvmArgs '-noverify'
}
```

Прочитайте Java на Android онлайн: https://riptutorial.com/ru/android/topic/9223/java-на-android

глава 50: JCodec

Examples

Начиная

Вы можете получить JCodec автоматически с помощью maven. Для этого просто добавьте ниже фрагмент к вашему pom.xml.

```
<dependency>
    <groupId>org.jcodec</groupId>
    <artifactId>jcodec-javase</artifactId>
    <version>0.1.9</version>
</dependency>
```

Получение кадра из фильма

Получение одного кадра из фильма (поддерживает только AVC, H.264 в MP4, ISO BMF, Quicktime-контейнер):

```
int frameNumber = 150;
BufferedImage frame = FrameGrab.getFrame(new File("filename.mp4"), frameNumber);
ImageIO.write(frame, "png", new File("frame_150.png"));
```

Получение последовательности кадров из фильма (поддерживает только AVC, H.264 в MP4, ISO BMF, Quicktime-контейнер):

```
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);
}
```

Прочитайте JCodec онлайн: https://riptutorial.com/ru/android/topic/9948/jcodec

глава 51: JSON в Android с org.json

Синтаксис

- **Объект** : объект представляет собой неупорядоченный набор пар имя / значение. Объект начинается с {(левая скобка) и заканчивается на} (правая фигурная скобка). За каждым именем следует: (двоеточие), а пары имя / значение разделяются запятой.
- **Массив** : массив представляет собой упорядоченный набор значений. Массив начинается с [(левая скобка) и заканчивается на] (правая скобка). Значения разделяются запятой.
- **Значение** . Значение может быть строкой в двойных кавычках, или числом, или истинным, или ложным или нулевым, или объектом или массивом. Эти структуры могут быть вложенными.
- **Строка** : Строка представляет собой последовательность из нуля или более символов Юникода, завернутую в двойные кавычки, с помощью обратных слэшей. Символ представляется как одна символьная строка. Строка очень похожа на строку C или Java.
- **Номер** : число очень похоже на число C или Java, за исключением того, что восьмеричные и шестнадцатеричные форматы не используются.

замечания

Этот `org.json` посвящен использованию пакета `org.json` , который включен в Android SDK.

Examples

Разбирайте простой объект JSON

Рассмотрим следующую строку JSON:

```
{
  "title": "test",
  "content": "Hello World!!!",
  "year": 2016,
  "names" : [
        "Hannah",
        "David",
        "Steve"
   ]
}
```

Этот объект JSON может быть проанализирован с использованием следующего кода:

```
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());
}
```

Вот еще один пример: JSONArray, вложенный внутри JSONObject:

```
{
    "books":[
      {
        "title":"Android JSON Parsing",
        "times_sold":186
      }
    ]
}
```

Это может быть проанализировано следующим кодом:

```
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");
```

Создание простого объекта JSON

Создайте `JSONObject` с помощью пустого конструктора и добавьте поля, используя метод `put()` , который перегружен, чтобы он мог использоваться с различными типами:

```
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);
}
```

Полученная строка `JSON` выглядит так:

```
{
   "name":"test",
   "content":"Hello World!!!1",
   "year":2016,
   "value":3.23,
   "member":true,
   "null_value":null
}
```

Добавить JSONArray в 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();
```

Полученная строка JSON выглядит так:

```
{
   "the_array":[
      "ASDF",
      "QWERTY"
   ]
}
```

Создайте строку JSON с нулевым значением.

Если вам нужно создать строку JSON со значением `null` например:

```
{
   "name":null
}
```

Затем вам нужно использовать специальную константу JSONObject.NULL .

Пример функционирования:

```
jsonObject.put("name", JSONObject.NULL);
```

Работа с нулевой строкой при разборе json

```
{
    "some_string": null,
    "ather_string": "something"
}
```

Если мы будем использовать этот способ:

```
JSONObject json = new JSONObject(jsonStr);
String someString = json.optString("some_string");
```

У нас будет выход:

```
someString = "null";
```

Поэтому мы должны обеспечить это обходное решение:

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

А затем позвоните:

```
JSONObject json = new JSONObject(jsonStr);
String someString = optNullableString(json, "some_string");
String someString2 = optNullableString(json, "some_string", "");
```

И мы будем иметь Output, как мы и ожидали:

```
someString = null; //not "null"
someString2 = "";
```

Использование JsonReader для чтения JSON из потока

`JsonReader` считывает кодированное значение JSON как поток токенов.

```
  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);
  }
```

Создание вложенного объекта JSON

Чтобы создать вложенный объект JSON, вам нужно просто добавить один объект JSON к другому:

```
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";
}
```

Теперь `mainObject` содержит ключ, называемый `claim` с целым `requestObject` в качестве значения.

Обработка динамического ключа для ответа JSON

Это пример того, как обрабатывать динамический ключ для ответа. Здесь `A` и `B` являются динамическими ключами, это может быть что угодно

отклик

```
{
  "response": [
    {
      "A": [
        {
          "name": "Tango"
        },
        {
          "name": "Ping"
        }
      ],
      "B": [
        {
          "name": "Jon"
        },
        {
          "name": "Mark"
        }
      ]
    }
  ]
}
```

Код 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);
    }
}
```

Проверьте наличие полей на JSON

Иногда бывает полезно проверить наличие или отсутствие поля на вашем JSON, чтобы избежать исключения `JSONException` в вашем коде.

Для этого используйте `JSONObject#has(String)` или метод, например, в следующем примере:

Образец JSON

```
{
   "name":"James"
}
```

Код 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".
```

Обновление элементов в JSON

образец json для обновления

```
 {
 "student":{"name":"Rahul", "lastname":"sharma"},
 "marks":{"maths":"88"}
 }
```

Чтобы обновить значение элементов в json, нам нужно назначить значение и обновление.

```
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);
}
```

обновленное значение

```
 {
 "student":{"name":"Kumar", "lastname":"sharma"},
 "marks":{"maths":"88"}
 }
```

Прочитайте JSON в Android с org.json онлайн: https://riptutorial.com/ru/android/topic/106/json-в-android-с-org-json

глава 52: Leakcanary

Вступление

Leak Canary - это Android и Java-библиотека, используемая для обнаружения утечки в приложении

замечания

Вы можете увидеть пример в ссылке ниже

https://github.com/square/leakcanary

Examples

Внедрение Leak Canary в приложении для Android

В вашем *build.gradle* вам нужно добавить следующие зависимости:

```
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'
```

В вашем классе `Application` вам нужно добавить код ниже внутри вашего `onCreate()` :

```
LeakCanary.install(this);
```

Это все, что вам нужно сделать для *LeakCanary* , оно будет автоматически показывать уведомления, когда есть утечка в вашей сборке.

Прочитайте Leakcanary онлайн: https://riptutorial.com/ru/android/topic/10041/leakcanary

глава 53: Lint Warnings

замечания

Инструмент **Lint** проверяет исходные файлы проекта Android на возможные ошибки и улучшает оптимизацию для правильности, безопасности, производительности, удобства использования, доступности и интернационализации. Вы можете запустить Lint из командной строки или из Android Studio.

Официальная документация:

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

Examples

Использование инструментов: игнорировать в xml-файлах

Инструменты атрибута `tools:ignore` можно использовать в файлах xml, чтобы убрать предупреждения lint.

НО отклонение предупреждений lint с этой техникой в большинстве случаев является неправильным способом продолжения.

Предупреждение о завязке должно быть понято и исправлено ... его можно игнорировать, если и только если у вас есть полное понимание его смысла и сильная причина его игнорировать.

Вот прецедент, когда законно игнорировать предупреждение:

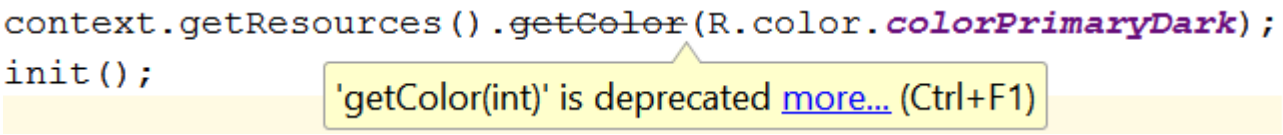
- Вы разрабатываете системное приложение (подписанное с ключом производителя устройства)
- Вашему приложению необходимо изменить дату устройства (или любое другое защищенное действие)

Затем вы можете сделать это в своем манифесте: (т. Е. Запрашивать защищенное разрешение и игнорировать предупреждение о линии, потому что вы знаете, что в вашем случае разрешение будет предоставлено)

```
<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"/>
```

Импорт ресурсов без ошибки «Устаревшая»

Используя Android API 23 или выше, очень часто можно увидеть такую ситуацию:



Эта ситуация вызвана структурными изменениями API Android в отношении получения ресурсов. Теперь функция:

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

должен быть использован. Но в библиотеке android.support.v4 есть другое решение.

Добавьте следующую зависимость в файл build.gradle:

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

Затем доступны все методы из библиотеки поддержки:

```
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));
```

Кроме того, можно использовать больше методов из библиотеки поддержки:

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

Настроить LintOptions с помощью градиента

Вы можете настроить пух, добавив `lintOptions` раздел в `build.gradle` файл:

```
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
    }
}
```

Вы можете запустить lint для определенного варианта (см. Ниже), например `./gradlew lintRelease` , или для всех вариантов (`./gradlew lint`), и в этом случае он создает отчет, в котором описываются конкретные варианты, к которым относится данная проблема.

Проверьте здесь ссылку DSL для всех доступных опций .

Как настроить файл lint.xml

Вы можете указать свои настройки проверки Lint в файле `lint.xml` . Если вы создаете этот файл вручную, поместите его в корневой каталог вашего Android-проекта. Если вы настраиваете настройки Lint в Android Studio, файл lint.xml автоматически создается и добавляется в ваш проект Android для вас.

Пример:

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

Установив значение атрибута серьезности в теге, вы можете отключить проверку Lint для проблемы или изменить уровень серьезности проблемы.

В следующем примере показано содержимое файла `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>
```

Настройка проверки линта в исходных файлах Java и XML

Вы можете отключить проверку Lint из ваших исходных файлов Java и XML.

Настройка проверки lint в Java

Чтобы отключить проверку Lint специально для **Java-класса** или метода в вашем проекте Android, добавьте **аннотацию** `@SuppressLint` к этому Java-коду.

Пример:

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

Чтобы отключить проверку всех проблем Lint:

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

Настройка проверки линта в XML

Вы можете использовать атрибуты `tools:ignore` чтобы отключить проверку Lint для определенных разделов ваших **XML-файлов** .

Например:

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

Чтобы подавить проверку всех проблем Lint в элементе XML, используйте

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

Отказ от комментариев

Хорошая практика отметить некоторые предупреждения в коде. Например, некоторые устаревшие методы необходимы для вашего тестирования или старой версии поддержки. Но проверка Lint отметит этот код предупреждениями. Чтобы избежать этой проблемы,

вам нужно использовать аннотацию @SuppressWarnings.

Например, добавьте игнорирование предупреждений на устаревшие методы. Вы также должны указать описание предупреждений в аннотации:

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

Используя эту аннотацию, вы можете игнорировать все предупреждения, включая Lint, Android и другие. Использование Suppress Warnings помогает правильно понять код!

Прочитайте Lint Warnings онлайн: https://riptutorial.com/ru/android/topic/129/lint-warnings

глава 54: Looper

Вступление

`Looper` - это класс Android, используемый для запуска цикла сообщений для потока, который, как правило, не связан с ними.

Наиболее распространенным `Looper` в Android является основной цикл, также известный как основной поток. Этот экземпляр уникален для приложения, и его можно получить статически с помощью `Looper.getMainLooper()` .

Если `Looper` связан с текущим потоком, его можно получить с помощью `Looper.myLooper()` .

Examples

Создайте простой LooperThread

Типичный пример реализации потока `Looper` предоставленного официальной документацией, использует `Looper.prepare()` и `Looper.loop()` и связывает `Handler` с циклом между этими вызовами.

```
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();
    }
}
```

Запустить цикл с помощью HandlerThread

`HandlerThread` можно использовать для запуска потока с помощью `Looper` . Затем этот петлитель можно использовать для создания `Handler` для связи с ним.

```
HandlerThread thread = new HandlerThread("thread-name");
thread.start();
Handler handler = new Handler(thread.getLooper());
```

Прочитайте Looper онлайн: https://riptutorial.com/ru/android/topic/10593/looper

глава 55: LruCache

замечания

Вы должны использовать Lru Cache в приложениях, где повторяющиеся нагрузки на ресурсы повлияют на поведение плавного приложения. Например, фотогалерея с большими эскизами (128x128).

Всегда будьте осторожны с размером кеша Lru, так как слишком высокая установка может повлиять на приложение.

После того, как Lru Cache больше не полезен, избегайте ссылок на него, чтобы позволить сборщику мусора очистить его от памяти.

Для лучшей производительности не забудьте загрузить ресурсы, такие как растровые изображения, используя лучшие методы, такие как выбор правильного параметра inSampleSize, прежде чем добавлять его в кэш Lru.

Examples

Инициализация кеша

Кэш Lru сохранит все добавленные ресурсы (значения) для быстрого доступа, пока не достигнет предела памяти, и в этом случае он потеряет менее используемый ресурс (значение) для хранения нового.

Для инициализации кеша Lru вам необходимо указать максимальное значение памяти. Это значение зависит от ваших требований к приложениям и от того, насколько важно, чтобы ресурс поддерживал плавное использование приложения. Рекомендуемое значение для галереи изображений, например, будет 1/8 из вашей максимальной доступной памяти.

Также обратите внимание, что Lru Cache работает на основе ключа. В следующем примере ключ - это `String` а значение - `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();
    }
};
```

Добавление растрового изображения (ресурса) в кеш

Чтобы добавить ресурс в кэш, вы должны предоставить ключ и ресурс. Сначала убедитесь, что значение уже не в кеше

```
public void addResourceToMemoryCache(String key, Bitmap resource) {
    if (memoryCache.get(key) == null)
        memoryCache.put(key, resource);
}
```

Получение растрового изображения (Resouce) из кеша

Чтобы получить ресурс из кэша, просто передайте ключ вашего ресурса (String в этом примере)

```
public Bitmap getResourceFromMemoryCache(String key) {
    memoryCache.get(key);
}
```

Прочитайте LruCache онлайн: https://riptutorial.com/ru/android/topic/7709/lrucache

глава 56: MediaSession

Синтаксис

- void mediaSessionCompat.setFlags (int flags)
- void mediaSessionCompat.setMediaButtonReceiver (PendingIntent mbr)
- void mediaSessionCompat.setCallback (обратный вызов MediaSessionCompat.Callback)
- void mediaSessionCompat.setActive (boolean active)
- MediaSessionCompat.Token mediaSessionCompat.getSessionToken ()
- void mediaSessionCompat.release ()
- void mediaSessionCompat.setPlaybackState (состояние PlaybackStateCompat)
- void mediaSessionCompat.setMetadata (метаданные MediaMetadataCompat)

замечания

Для лучшей практики используйте библиотеку **media-compat** . Библиотека поддерживает обратную совместимость, переведя методы сеанса мультимедиа в эквивалентные методы на более ранних версиях платформы, когда они доступны.

Examples

События для приема и обработки кнопок

В этом примере создается объект `MediaSession` при `MediaSession Service` . Объект `MediaSession` освобождается при уничтожении `Service` :

```
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();
    }
}
```

Следующий метод создает и настраивает `MediaSession` вызовы кнопки `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);
}
```

Следующий метод отправляет метаданные (хранящиеся в `HashMap`) на устройство с использованием 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);
}
```

Следующий метод устанавливает `PlaybackState` . Он также устанавливает, какие действия кнопки `MediaSession` будут реагировать на:

```
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);
}
```

Прочитайте MediaSession онлайн: https://riptutorial.com/ru/android/topic/6250/mediasession

глава 57: MediaStore

Examples

Извлечение аудио / MP3-файлов из определенной папки устройства или выборка всех файлов

Во-первых, добавьте следующие разрешения для манифеста вашего проекта, чтобы разрешить доступ к хранилищу устройств:

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

Затем создайте файл *AudioModel.class* и поместите в него следующий класс модели, чтобы разрешить получение и установку элементов списка:

```
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;
    }
}
```

Затем используйте следующий метод для чтения всех файлов MP3 из папки вашего устройства или для чтения всех файлов вашего устройства:

```
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;
}
```

Приведенный выше код вернет список всех файлов MP3 с именем, дорожкой, исполнителем и альбомом. Для получения дополнительной информации см. Документацию Media.Store.Audio .

Чтобы прочитать файлы определенной папки, используйте следующий запрос (вам нужно заменить имя папки):

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    MediaStore.Audio.Media.DATA + " like ? ",
    new String[]{"%yourFolderName%"}, // Put your device folder / file location here.
    null);
```

Если вы хотите получить все файлы с вашего устройства, используйте следующий запрос:

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    null,
    null,
    null);
```

Примечание. Не забудьте включить разрешения доступа к хранилищу.

Теперь все, что вам нужно сделать, это вызвать метод выше, чтобы получить файлы MP3:

```
getAllAudioFromDevice(this);
```

Пример с активностью

```
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;
    }
```

```
}
```

Прочитайте MediaStore онлайн: https://riptutorial.com/ru/android/topic/7136/mediastore

глава 58: Moshi

Вступление

Moshi - это современная библиотека JSON для Android и Java. Это упрощает анализ JSON на Java-объекты и Java обратно в JSON.

замечания

Не забывайте, всегда читайте README !

Examples

JSON в 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);
```

сериализовать объекты Java как 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);
```

Встроенные типовые адаптеры

Moshi имеет встроенную поддержку для чтения и записи основных типов данных Java:

- Примитивы (int, float, char ...) и их бокс-аналоги (Integer, Float, Character ...).
- Массивы
- Коллекции
- Списки
- наборы
- Карты Строки Перечисления

Он поддерживает ваши классы моделей, выписывая их по полям. В приведенном выше примере Моши использует эти классы:

```
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"
    }
  ]
}
```

Прочитайте Moshi онлайн: https://riptutorial.com/ru/android/topic/8744/moshi

глава 59: MVVM (Архитектура)

замечания

Синтаксические причуды с DataBinding

При привязке функции viewModel к свойству в xml некоторые префиксы функций, такие как `get` или `is` , удаляются. Например. `ViewModel::getFormattedText` в ViewModel станет `@{viewModel.formattedText}` при привязке его к свойству в xml. Аналогично `ViewModel::isContentVisible` -> `@{viewModel.contentVisible}` (нотация Java Bean)

Сгенерированные классы привязки, такие как `ActivityMainBinding` , называются после xml, для которого они создаются привязки, а не для класса java.

Пользовательские привязки

В файле activity_main.xml я устанавливаю атрибут textColor в `app` а не пространство имен `android` . Это почему? Поскольку для атрибута `textColor` задан пользовательский сеттер, который разрешает идентификатор ресурса ColorRes, отправляемый ViewModel, фактическому цвету.

```
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);
    }

}
```

Подробнее о том, как это работает, проверьте библиотеку DataBinding: пользовательские сеттеры

Подождите ... есть логика в вашем xml !!!?

Вы можете утверждать, что то, что я делаю в xml для `android:visibility` и `app:textColor` являются неправильными / анти-шаблонами в контексте MVVM, потому что на мой взгляд

существует логика представления. Однако я бы сказал, что для меня важнее сохранить зависимости Android от моего ViewModel для тестирования.

Кроме того, что действительно делает `app:textColor` ? Он разрешает только указатель ressource на фактический цвет, связанный с ним. Таким образом, ViewModel все еще решает, какой цвет отображается на основе какого-либо условия.

Что касается `android:visibility` я чувствую из-за того, как называется метод, на самом деле это нормально использовать здесь тернарный оператор. Из-за имени `isLoadingVisible` и `isContentVisible` действительно нет сомнений в том, что каждый результат должен решить в представлении. Поэтому я чувствую, что это скорее выполнение команды, заданной ViewModel, чем реализация логики представления.

С другой стороны, я согласен с тем, что с помощью `viewModel.isLoading ? View.VISIBLE : View.GONE` было бы плохо, потому что вы делаете предположения в представлении, что это означает для представления.

Полезный материал

Следующие ресурсы помогли мне в попытке понять эту концепцию:

- Джереми Ликнес - Модель-View-ViewModel (MVVM) Разъяснение (C #) (08.2010)
- Шамлия Шуккур - Понимание основ дизайна MVVM (C #) (03.2013)
- Frode Nilsen - Android Databinding: Goodbye Presenter, Hello ViewModel! (07,2015)
- Джо Берч - Подходит к Android с MVVM (09.2015)
- Флорина Мунтенеску - шаблоны архитектуры Android Часть 3: Model-View-ViewModel (10.2016)

Examples

Пример MVVM с использованием библиотеки DataBinding

Вся цель MVVM состоит в том, чтобы отделить слои, содержащие логику, от уровня представления.

На Android мы можем использовать библиотеку DataBinding, чтобы помочь нам в этом и сделать большую часть нашей логической Unit-testable, не беспокоясь о зависимостях Android.

В этом примере я покажу центральные компоненты для глупого простого приложения, которое делает следующее:

- При запуске поддельного сетевого вызова и покажите загрузчик
- Показывать представление с помощью счетчика кликов TextView, текстового текста и кнопки для увеличения счетчика
- На кнопке щелкните счетчик обновления и цвет счетчика обновлений и текст

сообщения, если счетчик достигнет некоторого числа

Начнем с слоя вида:

`activity_main.xml` :

Если вы не знакомы с тем, как работает DataBinding, вам, вероятно, потребуется 10 минут, чтобы ознакомиться с ним. Как вы можете видеть, все поля, которые вы обычно обновляете сеттерами, привязаны к функциям переменной viewModel.

Если у вас возник вопрос о свойствах `android:visibility` или `app:textColor` проверьте раздел «Примечания».

```
<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>
```

Затем слой модели. Здесь у меня есть:

- два поля, представляющие состояние приложения
- getters читать количество кликов и состояние возбуждения
- метод увеличения количества кликов
- метод восстановления некоторого предыдущего состояния (важно для изменений ориентации)

Также я определяю здесь «состояние возбуждения», которое зависит от количества кликов. Впоследствии это будет использовано для обновления цвета и сообщения в представлении.

Важно отметить, что в модели нет предположений о том, как состояние может отображаться пользователю!

`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;
```

```
        }
    }
}
```

Затем ViewModel.

Это приведет к изменениям в модели и форматированию данных из модели, чтобы показать их в представлении. Обратите внимание, что именно здесь мы оцениваем, какое представление GUI подходит для состояния, заданного моделью (`resolveCounterColor` и `resolveLabelText`). Таким образом, мы могли бы, например, легко реализовать `UnderachieverClickerModel` который имеет более низкие пороговые значения для состояния возбуждения, не касаясь какого-либо кода в viewModel или представлении.

Также обратите внимание, что ViewModel не содержит ссылок на объекты просмотра. Все свойства привязаны через аннотации `@Bindable` и обновляются, когда либо `notifyChange()` (сигнализирует, что все свойства необходимо обновить), либо `notifyPropertyChanged(BR.propertyName)` (сигнализирует, что эти свойства необходимо обновить).

`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;
        }
    }

}
```

Связывание всего этого в Деятельности!

Здесь мы видим представление, инициализирующее viewModel со всеми зависимостями, которые могут потребоваться, которые должны быть созданы из контекста android.

После инициализации viewModel он привязан к XML-макету через DataBindingUtil (см. Раздел «Синтаксис» для именования сгенерированных классов).

Примечание. Подписки подписываются на этот уровень, потому что мы должны обрабатывать отмену подписки на них, когда действие приостановлено или уничтожено, чтобы избежать утечек памяти и NPE. Также здесь сохраняется и сохранение и перезагрузка viewState на 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;
    }
}
```

Чтобы увидеть все в действии, посмотрите этот примерный проект .

Прочитайте MVVM (Архитектура) онлайн: https://riptutorial.com/ru/android/topic/7549/mvvm--архитектура-

глава 60: NavigationView

замечания

NavigationView представляет собой стандартное меню навигации для приложения. Содержимое меню может быть заполнено файлом ресурсов меню.

Перед использованием `NavigationView` вы должны добавить зависимость библиотеки поддержки дизайна в файле build.gradle:

```
 dependencies {
    compile 'com.android.support:design:24.2.0'
}
```

Официальная документация:

https://developer.android.com/reference/android/support/design/widget/NavigationView.html

Технические характеристики материалов:

https://material.google.com/patterns/navigation-drawer.html#navigation-drawer-content

Examples

Как добавить NavigationView

Чтобы использовать NavigationView, просто добавьте зависимость в файле `build.gradle` как описано в разделе примечаний

Затем добавьте `NavigationView` в макет

```
<?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 : представление, которое будет отображаться в верхней части ящика

```
<?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 абстракции для панели инструментов, чтобы отделить ее от содержимого:

```
<?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 Реальный контент активности только для демонстрации, здесь вы бы поставили свой обычный макет 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>
```

Определите файл меню как *res/menu/activity_main_drawer.xml* :

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

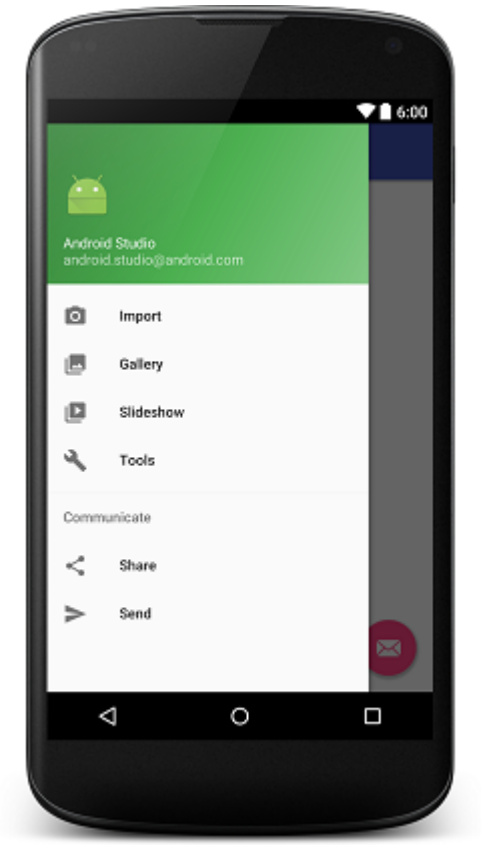
```
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_camera"
            android:icon="@drawable/ic_menu_camera"
            android:title="Import" />
        <item
            android:id="@+id/nav_gallery"
            android:icon="@drawable/ic_menu_gallery"
            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>
```

И, наконец, `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;
    }
}
```

Это будет выглядеть так:

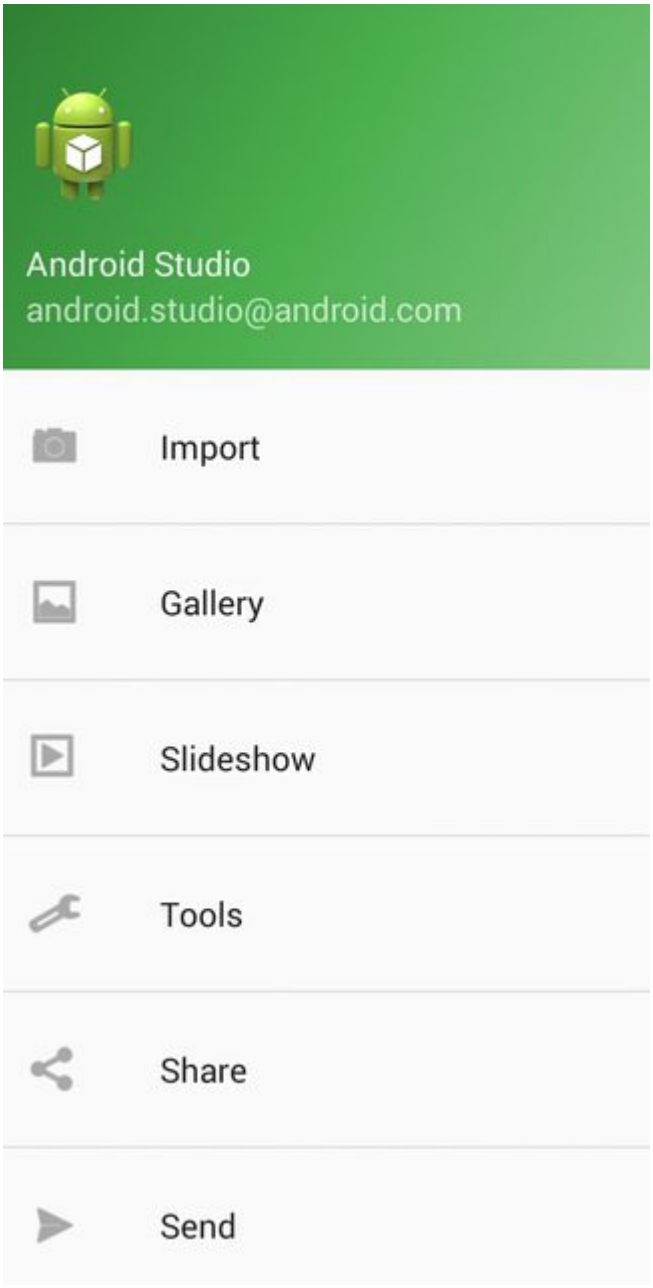


Добавить подчеркивание в элементах меню

Каждая группа заканчивается разделителем строк. Если каждый элемент в вашем меню имеет свою группу, вы достигнете желаемого графического вывода. Он будет работать только в том случае, если ваши разные группы имеют разные `android:id` . Кроме того, в `menu.xml` забудьте указать `android:checkable="true"` для отдельного элемента `android:checkableBehavior="single"` для группы элементов.

```
<?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>
```



Добавить разделителей в меню

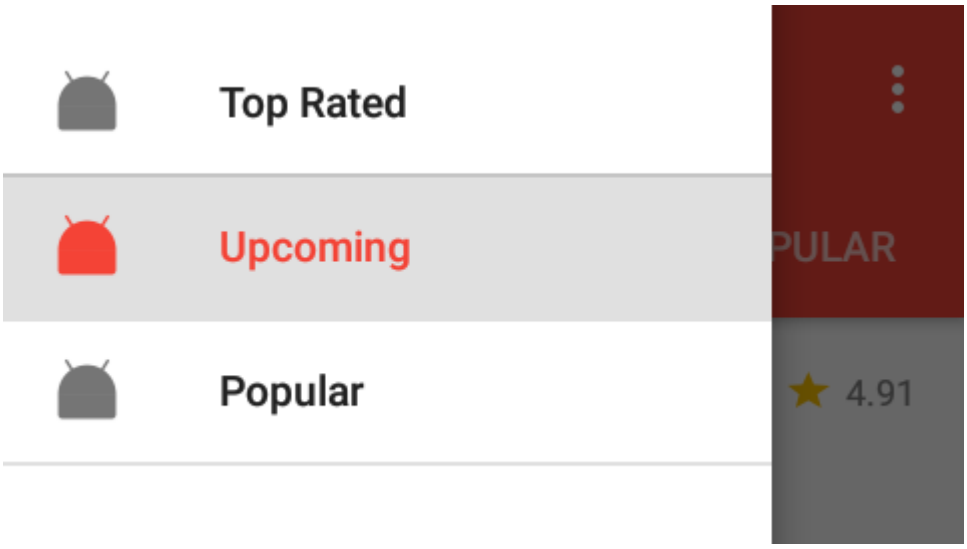
Получите доступ к RecyclerView в NavigationView и добавьте ItemDecoration к нему.

```
NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new DividerItemDecoration(this));
```

Код для 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);
        }
    }
}
```

Предварительный просмотр:

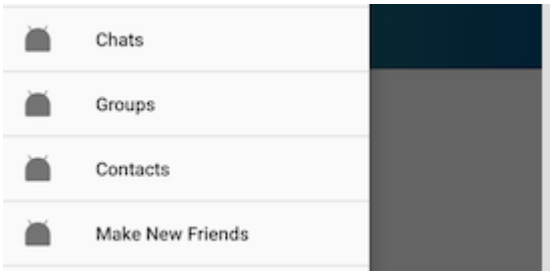


Добавьте разделитель меню, используя DividerItemDecoration по умолчанию.

Просто используйте класс DividerItemDecoration по умолчанию:

```
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new
DividerItemDecoration(context,DividerItemDecoration.VERTICAL));
```

Предварительный просмотр:



Прочитайте NavigationView онлайн: https://riptutorial.com/ru/android/topic/97/navigationview

глава 61: Notification Channel Android O

Вступление

Каналы уведомлений позволяют разработчикам приложений группировать наши уведомления по группам-каналам, причем пользователь имеет возможность изменять настройки уведомлений для всего канала сразу. В Android O эта функция введена. Теперь он доступен для предварительного просмотра разработчиков.

Синтаксис

1. class NotificationUtils {} // Для создания канала уведомлений
2. createChannel () // Общий метод для создания уведомления

параметры

метод	Описание
IMPORTANCE_MAX	неиспользуемый
IMPORTANCE_HIGH	везде показывает, шумит и заглядывает
IMPORTANCE_DEFAULT	показывает везде, шумит, но не визуально вторгается
IMPORTANCE_LOW	показывает везде, но не навязчиво
IMPORTANCE_MIN	только показывает в тени, под складкой
IMPORTANCE_NONE	уведомление не имеет значения; не отображается в тени

Examples

Канал уведомлений

Что такое каналы уведомлений?

Каналы уведомлений позволяют разработчикам приложений группировать наши уведомления в группы-каналы, причем пользователь имеет возможность изменять настройки уведомлений для всего канала одновременно. Например, для каждого канала пользователи могут полностью блокировать все уведомления, переопределять уровень важности или показывать значок уведомлений. Эта новая функция помогает значительно улучшить пользовательский интерфейс приложения.

Создание каналов уведомлений

```
 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;
}}
```

В приведенном выше коде мы создали два экземпляра NotificationChannel, передав uniqueid

имя канала, а также уровень важности в его конструкторе. Для каждого канала уведомлений мы применяли следующие характеристики.

1. звук
2. огни
3. вибрация
4. Уведомление на экране блокировки.

Наконец, мы получили NotificationManager из системы, а затем зарегистрировали канал, вызвав метод createNotificationChannel (), передав канал, который мы создали.

Мы можем создавать сразу несколько каналов уведомлений с помощью createNotificationChannels (), передавая список Java экземпляров NotificationChannel. Вы можете получить все каналы уведомлений для приложения с помощью getNotificationChannels () и получить определенный канал с помощью getNotificationChannel (), передавая только идентификатор канала в качестве аргумента.

Уровень важности в каналах уведомлений

метод	Описание
IMPORTANCE_MAX	неиспользуемый
IMPORTANCE_HIGH	везде показывает, шумит и заглядывает
IMPORTANCE_DEFAULT	показывает везде, шумит, но не визуально вторгается
IMPORTANCE_LOW	показывает везде, но не навязчиво, значение равно 0
IMPORTANCE_MIN	только показывает в тени, под складкой
IMPORTANCE_NONE	уведомление не имеет значения; не отображается в тени

Создать уведомление и отправить на канал

Мы создали два уведомления, используя NotificationUtils другой, используя 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);
}
```

Мы также можем установить NotificationChannel, используя Notification.Builder (). Для этого мы можем использовать ***setChannel (String channelId)*** .

Обновление настроек канала уведомлений

После создания канала уведомлений пользователь отвечает за его настройки и поведение. Вы можете снова вызвать createNotificationChannel (), чтобы переименовать существующий канал уведомлений или обновить его описание. Следующий пример кода описывает, как вы можете перенаправить пользователя к настройкам для канала уведомлений, создав намерение начать действие. В этом случае цель требует расширенных данных, включая идентификатор канала уведомления и имя пакета вашего приложения.

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

XML-файл:

```
<!--...-->
<Button
    android:id="@+id/btn_android_notif_settings"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Notification Settings"/>
<!--...-->
```

Удаление канала уведомлений

Вы можете удалить каналы уведомлений, вызвав 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);
```

Теперь создайте MainActivity и 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

мы собираемся отредактировать нашу MainActivity, чтобы мы могли получить название и автора из компонентов EditText, а затем отправить их на канал Android. Мы получаем Notification.Builder для канала Android, который мы создали в нашем NotificationUtils, а затем уведомляем NotificationManager.

import android.app.Notification; import android.os.Bundle; import android.support.v7.app.AppCompatActivity; import android.text.TextUtils; импортировать android.view.View; import android.widget.Button; import 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());
                }
            }
        });
    }
}
```

Прочитайте Notification Channel Android O онлайн:
https://riptutorial.com/ru/android/topic/10018/notification-channel-android-o

глава 62: OkHttp

Examples

Входной перехватчик

`Interceptors` используются для перехвата вызовов `OkHttp` . Причиной перехвата может быть мониторинг, переписывание и повторные вызовы. Он может использоваться для исходящего запроса или входящего ответа.

```
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;
  }
}
```

Перезагрузка ответов

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

Основной пример использования

Мне нравится обертывать мой `OkHttp` в класс под названием `HttpClient` например, и в этом классе у меня есть методы для каждого из основных HTTP-глаголов, `post` , `get` , `put` и `delete` , чаще всего. (Обычно я использую интерфейс, чтобы его можно было реализовать, чтобы иметь возможность легко перейти на другую реализацию, если потребуется):

```
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();
}
```

Синтаксис одинаковый для `put` , `get` и `delete` за исключением 1 слова (`.put(body)`), поэтому было бы неприятно публиковать этот код. Использование довольно просто, просто вызовите соответствующий метод на каком-то `url` с некоторой полезной нагрузкой `json` и метод вернет строку, в результате вы сможете впоследствии использовать и анализировать. Предположим, что ответ будет `json` , мы можем легко создать `JSONObject` :

```
String response = httpClient.post(MY_URL, JSON_PAYLOAD);
JSONObject json = new JSONObject(response);
// continue to parse the response according to it's structure
```

Синхронный вызов

```
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());
  }
```

Асинхронный вызов

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

Параметры формы проводки

```
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());
  }
```

Проводка многостраничного запроса

```
 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());
  }
```

Настройка OkHttp

Захватите через Maven:

```
<dependency>
  <groupId>com.squareup.okhttp3</groupId>
  <artifactId>okhttp</artifactId>
  <version>3.6.0</version>
</dependency>
```

или Gradle:

```
compile 'com.squareup.okhttp3:okhttp:3.6.0'
```

Прочитайте OkHttp онлайн: https://riptutorial.com/ru/android/topic/3625/okhttp

глава 63: ORMLite в android

Examples

Android OrmLite по примеру SQLite

ORMLite - это пакет реляционного сопоставления объектов, который обеспечивает простую и легкую функциональность для сохранения объектов Java в SQL-базах данных, избегая при этом сложностей и накладных расходов более стандартных пакетов ORM.

Говоря о Android, OrmLite реализуется по доступной базе данных SQLite. Он делает прямые вызовы API для доступа к SQLite.

Настройка гребенки

Чтобы начать работу, вы должны включить пакет в градиент сборки.

```
 // https://mvnrepository.com/artifact/com.j256.ormlite/ormlite-android
compile group: 'com.j256.ormlite', name: 'ormlite-android', version: '5.0'
POJO configuration
```

Затем вы должны настроить POJO для сохранения в базе данных. Здесь следует обратить внимание на аннотации:

- Добавьте аннотацию @DatabaseTable в начало каждого класса. Вы также можете использовать @Entity.
- Добавьте аннотацию @DatabaseField прямо перед каждым полем, которое необходимо сохранить. Вы также можете использовать @Column и другие.
- Добавьте конструктор no-argument для каждого класса с видимостью по крайней мере пакета.

```
@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;
    }
}
```

В приведенном выше примере есть одна таблица (form_model) с 4 полями.

Поле id - это автоматически созданный индекс.

username - это индекс базы данных.

Более подробную информацию об аннотации можно найти в официальной документации .

Помощник базы данных

Для продолжения вам необходимо создать вспомогательный класс базы данных, который должен расширить класс OrmLiteSqliteOpenHelper.

Этот класс создает и обновляет базу данных, когда ваше приложение установлено, а также может предоставлять классы DAO, используемые другими вашими классами.

DAO означает объект доступа к данным и обеспечивает все функциональные возможности scrum и специализируется на обработке одного сохраненного класса.

Класс-помощник должен реализовывать следующие два метода:

- onCreate (SQLiteDatabase sqliteDatabase, ConnectionSource connectionSource);

  onCreate создает базу данных, когда ваше приложение впервые установлено

- onUpgrade (база данных SQLiteDatabase, ConnectionSource connectionSource, int oldVersion, int newVersion);

  onUpgrade обрабатывает обновление таблиц базы данных при обновлении приложения до новой версии

Пример класса помощника базы данных:

```
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;
        }
    }
```

Постоянный объект для SQLite

Наконец, класс, который сохраняет объект в базе данных.

```
     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;
    }
}
```

Аксессор DOA у конструктора вышеуказанного класса определяется как:

```
 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;
}
```

Прочитайте ORMLite в android онлайн: https://riptutorial.com/ru/android/topic/7571/ormlite-в-android

глава 64: PackageManager

Examples

Получить версию приложения

```
public String getAppVersion() throws PackageManager.NameNotFoundException {
    PackageManager manager = getApplicationContext().getPackageManager();
    PackageInfo info = manager.getPackageInfo(
                            getApplicationContext().getPackageName(),
                            0);

    return info.versionName;
}
```

Название версии и код версии

Чтобы получить `versionName` и `versionCode` текущей сборки вашего приложения, вы должны запросить менеджер пакетов 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
}
```

Время установки и время обновления

Чтобы получить время, в которое ваше приложение было установлено или обновлено, вы должны запросить менеджер пакетов 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
}
```

Утилитный метод с помощью PackageManager

Здесь мы можем найти полезный метод, используя PackageManager,

Ниже метод поможет получить имя приложения, используя имя пакета

```
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;
}
```

Ниже метод поможет получить значок приложения, используя имя пакета,

```
private Drawable getAppIcon(String packageName, Context context) {
    Drawable appIcon = null;
    try {
        appIcon = context.getPackageManager().getApplicationIcon(packageName);
    } catch (PackageManager.NameNotFoundException e) {
    }

    return appIcon;
}
```

Ниже метод поможет получить список установленных приложений.

```
public static List<ApplicationInfo> getLaunchIntent(PackageManager packageManager) {

    List<ApplicationInfo> list =
packageManager.getInstalledApplications(PackageManager.GET_META_DATA);

    return list;
}
```

Примечание: выше метод также даст приложение запуска.

Ниже метод поможет скрыть значок приложения из панели запуска.

```
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);
    }
}
```

Примечание. После выключения устройства и включения этого значка появится в панели запуска.

Прочитайте PackageManager онлайн:
https://riptutorial.com/ru/android/topic/4670/packagemanager

глава 65: Parcelable

Вступление

Parcelable - это специфический для Android интерфейс, где вы сами реализуете сериализацию. Он был создан гораздо эффективнее, чем Serializable, и чтобы обойти некоторые проблемы со схемой сериализации Java по умолчанию.

замечания

Важно помнить, что порядок, в котором вы пишете поля в парцеллу, ДОЛЖЕН БЫТЬ ОЖИДАЕМОЙ ЗАКАЗ, что вы читаете их из посылки при создании своего настраиваемого объекта.

Порочный интерфейс имеет строгий предел размера 1 МБ. Это означает, что любой объект или комбинации объектов, которые вы помещаете в посылку, которая занимает более 1 МБ пространства, будет повреждена с другой стороны. Это может быть трудно обнаружить, поэтому имейте в виду, какие объекты вы планируете сделать обоснованными. Если они имеют большие деревья зависимостей, рассмотрите другой способ передачи данных.

Examples

Создание настраиваемого объекта.

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

Объект Parcelable, содержащий другой объект Parcelable

Пример класса, содержащего класс:

```
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;
    }
}
```

Владелец - просто нормальный класс.

Использование Enums with 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
}
```

Это предпочтительнее (например), используя порядковый номер, потому что вставка новых значений в ваш перечисление не повлияет на ранее сохраненные значения

Прочитайте Parcelable онлайн: https://riptutorial.com/ru/android/topic/1849/parcelable

глава 66: Ping ICMP

Вступление

Запрос ICMP Ping можно выполнить в Android, создав новый процесс для запуска запроса ping. Результат запроса может быть оценен после завершения запроса ping из его процесса.

Examples

Выполняет одиночный Ping

В этом примере выполняется попытка одного запроса Ping. Команда ping внутри `runtime.exec` метода `runtime.exec` может быть изменена на любую действительную команду ping, которую вы могли бы выполнить самостоятельно в командной строке.

```
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();
}
```

Прочитайте Ping ICMP онлайн: https://riptutorial.com/ru/android/topic/9434/ping-icmp

глава 67: ProGuard - Обфускация и сокращение вашего кода

Examples

Правила для некоторых из широко используемых библиотек

В настоящее время он содержит правила для следующих библиотек: -

1. Нож для масла
2. RxJava
3. Библиотека поддержки Android
4. Библиотека поддержки Android Design
5. модифицировать
6. Гссон и Джексон
7. эфирное масло
8. Crashlitycs
9. Пикассо
10. залп
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;
}
```

Включите ProGuard для вашей сборки

Для включения конфигураций `ProGuard` для вашего приложения вам необходимо включить его в файл уровня градиента уровня модуля. вам нужно установить значение `minifyEnabled true` .

Вы также можете включить `shrinkResources true` который удалит ресурсы, которые флаг `ProGuard` не используется.

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

Вышеприведенный код применит ваши конфигурации ProGuard, содержащиеся в `proguard-rules.pro` («proguard-project.txt» в Eclipse), к вашему выпущенному apk.

Чтобы позднее определить строку, в которой произошла ошибка в трассировке стека, « proguard-rules.pro» должен содержать следующие строки:

```
-renamesourcefileattribute SourceFile
-keepattributes SourceFile,LineNumberTable
```

Чтобы включить Proguard в Eclipse, добавьте `proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt` в " project.properties"

Удаление протоколов трассировки (и других) во время сборки

Если вы хотите удалить вызовы на определенные методы, считая, что они возвращают void и не имеют побочных эффектов (так как в них их не изменяются никакие системные значения, ссылочные аргументы, статика и т. Д.), Тогда вы можете заставить ProGuard удалить их из после завершения сборки.

Например, я нахожу это полезным при удалении отладочных / подробных протоколирующих операторов, полезных при отладке, но генерация строк для них не требуется в производстве.

```
# 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(...);
}
```

Примечание. Если `-dontoptimize` используется в любой конфигурации ProGuard, чтобы он не уменьшал / удалял неиспользуемый код, тогда это не будет вытеснять утверждения. (Но кто не хочет удалять неиспользуемый код, не так ли?)

Примечание2: этот вызов удалит вызов в журнал, но не защитит ваш код. Строки фактически останутся в сгенерированном apk. Читайте больше в этом сообщении .

Защита вашего кода от хакеров

Обфускация часто рассматривается как волшебное решение для защиты кода, делая ваш код более сложным для понимания, если он когда-либо декомпилируется хакерами.

Но если вы думаете, что удаление `Log.x(..)` фактически удаляет информацию, необходимую хакерам, у вас будет неприятный сюрприз.

Удаление всех вызовов журнала с помощью:

```
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    ...etc
}
```

действительно удалит сам вызов журнала, но обычно это *не* строки, которые вы вставляете в них.

Если, например, внутри вашего `Log.d(MyTag,"Score="+score);` вызова вы вводите общее сообщение журнала, например: `Log.d(MyTag,"Score="+score);` , компилятор преобразует символ `+` в новый StringBuilder () за пределами вызова журнала. ProGuard не изменяет этот новый объект.

У вашего скомпилированного кода по-прежнему будет висячий `StringBuilder` для `"Score="` , добавленный с запутанной версией для переменной `score` (предположим, что она была преобразована в `b`).
Теперь хакер знает, что такое `b` , и понимает ваш код.

Хорошая практика, чтобы фактически удалить эти остатки из вашего кода, либо не помещает их туда в первую очередь (вместо этого используйте вместо этого форматирование String, с правилами proguard для их удаления) или для переноса ваших вызовов `Log` с помощью:

```
    if (BuildConfig.DEBUG) {
        Log.d(TAG,".."+var);
    }
```

Совет:

Проверьте, насколько хорошо защищен ваш запутанный код, де-компилируя его самостоятельно!

1. dex2jar - преобразует apk в jar
2. jd - декомпилирует банку и открывает ее в редакторе gui

Включение ProGuard с помощью настраиваемого файла конфигурации обфускации

ProGuard позволяет разработчику запутывать, сжимать и оптимизировать свой код.

 # 1 Первым шагом процедуры является включение proguard в сборку .

Это можно сделать, **установив команду «minifyEnabled» в true** на нужную сборку

 # 2 Второй шаг - указать, какие файлы proguard мы используем для данной сборки

Это можно сделать, **установив строку 'proguardFiles' с правильными именами файлов**

```
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 Затем разработчик может редактировать свой файл proguard с правилами, которые он желает.

Это можно сделать, отредактировав файл (например, «proguard-rules-tests.pro») и добавив нужные ограничения. *Следующий файл служит примером файла 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
```

Наконец, всякий раз, когда разработчик запускает и / или генерирует свой новый .APK-файл, будут применены настраиваемые конфигурации proguard, выполняя таким образом требования.

Прочитайте ProGuard - Обфускация и сокращение вашего кода онлайн:
https://riptutorial.com/ru/android/topic/4500/proguard---обфускация-и-сокращение-вашего-кода

глава 68: RecyclerView

Вступление

RecyclerView - это более продвинутая версия List View с улучшенной производительностью и дополнительными функциями.

параметры

параметр	подробность
адаптер	Подкласс RecyclerView.Adapter, ответственный за предоставление представлений, которые представляют элементы в наборе данных
Позиция	Позиция элемента данных в адаптере
Индекс	Индекс прикрепленного дочернего представления, используемого при вызове getChildAt (int). Контраст с позицией
переплет	Процесс подготовки дочернего представления для отображения данных, соответствующих позиции в адаптере
Переработать (просмотреть)	Представление, ранее использовавшееся для отображения данных для конкретной позиции адаптера, может быть помещено в кеш для последующего повторного использования для повторного отображения одного и того же типа данных позже. Это может значительно повысить производительность, пропуская первоначальную инфляцию или строительство
Лом (просмотреть)	Детский вид, который во время макета вошел во временное состояние. Представления лома могут быть повторно использованы, не становясь полностью отделенными от родительского RecyclerView, либо немодифицированы, если переадресация не требуется или не изменена адаптером, если представление считается грязным
Грязный (просмотреть)	Детский вид, который должен быть отсканирован адаптером перед отображением

замечания

`RecyclerView` - это гибкое представление для предоставления ограниченного окна в большой

набор данных.

Перед использованием `RecyclerView` вам необходимо добавить зависимость библиотеки поддержки в файле `build.gradle` :

```
dependencies {
    // Match the version of your support library dependency
    compile 'com.android.support:recyclerview-v7:25.3.1'
}
```

Вы можете найти последнюю версию номера recyclerview с официального сайта .

Другие связанные темы:

Существуют и другие темы, описывающие компоненты `RecyclerView` :

- RecycleManager LayoutManagers
- RecyclerView ItemDecorations
- RecyclerView onClickListeners

Официальная документация

http://developer.android.com/reference/android/support/v7/widget/RecyclerView.html

Старые версии:

```
  //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

Добавление RecyclerView

Добавьте зависимость, как описано в разделе «Замечание», затем добавьте `RecyclerView` в ваш макет:

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

После добавления виджета `RecyclerView` к вашему макету, получите дескриптор объекта, подключите его к диспетчеру макета и прикрепите адаптер для отображаемых данных:

```
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);
```

Или просто настройте диспетчер компоновки из xml, добавив следующие строки:

```
xmlns:app="http://schemas.android.com/apk/res-auto"
app:layoutManager="android.support.v7.widget.LinearLayoutManager"
```

Если вы знаете, что изменения в содержимом `RecyclerView` не изменят размер макета `RecyclerView` , используйте следующий код для повышения производительности компонента. Если RecyclerView имеет фиксированный размер, он знает, что сам RecyclerView не будет изменять размер из-за его дочерних элементов, поэтому он не будет вызывать структуру запроса вообще. Он просто обрабатывает само изменение. Если это недействительно независимо от родителя, координатор, макет или что-то еще. (вы можете использовать этот метод еще до установки `LayoutManager` и `Adapter`):

```
mRecyclerView.setHasFixedSize(true);
```

`RecyclerView` предоставляет эти встроенные менеджеры макетов для использования. Таким

образом, вы можете создать список, сетку и шахматную сетку с помощью `RecyclerView` :

1. LinearLayoutManager показывает элементы в списке вертикальной или горизонтальной прокрутки.
2. GridLayoutManager показывает элементы в сетке.
3. StaggeredGridLayoutManager показывает элементы в шахматном порядке.

Более плавная загрузка предметов

Если элементы вашего `RecyclerView` загружают данные из сети (обычно изображения) или выполняют другую обработку, это может занять значительное количество времени, и вы можете получить элементы на экране, но не полностью загруженные. Чтобы этого избежать, вы можете расширить существующий `LinearLayoutManager` чтобы предварительно загрузить несколько элементов, прежде чем они станут видимыми на экране:

```
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;
  }
}
```

Перетаскивание и прокрутка с помощью RecyclerView

Вы можете реализовать функции перетаскивания и удаления с помощью функции RecyclerView без использования сторонних библиотек.
Просто используйте класс `ItemTouchHelper` включенный в библиотеку поддержки RecyclerView.

`ItemTouchHelper` экземпляр `ItemTouchHelper` с `SimpleCallback` обратного вызова `SimpleCallback` и в зависимости от того, какую функциональность вы поддерживаете, вы должны переопределить `onMove(RecyclerView, ViewHolder, ViewHolder)` и / или `onSwiped(ViewHolder, int)` и, наконец, подключиться к вашему `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);
```

Стоит отметить, что конструктор `SimpleCallback` применяет одну и ту же стратегию `SimpleCallback` всех элементов в `RecyclerView` . В любом случае возможно обновить направление `getSwipeDirs(RecyclerView, ViewHolder)` по умолчанию для определенных элементов, просто переопределив метод `getSwipeDirs(RecyclerView, ViewHolder)` .

Предположим, например, что наш `RecyclerView` включает в себя `HeaderViewHolder` и что мы, очевидно, не хотим применять его для прокрутки. Достаточно переопределить `getSwipeDirs` следующим образом:

```
@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);
}
```

Добавить заголовок / нижний колонтитул в RecyclerView

Это пример кода адаптера.

```
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
    }
}
}
```

Вот хорошее представление о реализации `RecyclerView` с верхним и нижним колонтитулом.

Альтернативный метод:

Хотя приведенный выше ответ будет работать, вы также можете использовать этот подход, используя представление recycler с помощью `NestedScrollView` Вы можете добавить макет для заголовка, используя следующий подход:

```
<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>
```

Или вы также можете использовать `LinearLayout` с вертикальным выравниванием в вашем `NestedScrollView` .

Примечание. Это будет работать только с `RecyclerView` выше **23.2.0**

```
compile 'com.android.support:recyclerview-v7:23.2.0'
```

Использование нескольких ViewHolders с ItemViewType

Иногда RecyclerView должен будет использовать несколько типов представлений, которые будут отображаться в списке, показанном в пользовательском интерфейсе, и каждому представлению нужен другой макет xml для раздувания.

Для этой проблемы вы можете использовать разные ViewHolders в одном адаптере, используя специальный метод в RecyclerView - `getItemViewType(int position)` .

Ниже приведен пример использования двух ViewHolders:

1. ViewHolder для отображения записей в списках
2. ViewHolder для отображения нескольких видов заголовков

```
@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.
    }
}
```

Фильтрация элементов внутри RecyclerView с помощью SearchView

добавить `filter` в `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` инициализируется в конструкторе адаптера, например `itemsCopy.addAll(items)` .

Если вы это сделаете, просто вызовите `filter` из `OnQueryTextListener` из `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;
```

```
    }
});
```

Всплывающее меню с recyclerView

поместите этот код в свой ViewHolder

note: В этом коде я использую `btnExpand` click-event, для всего события click `recyclerview` вы можете установить прослушиватель в объект itemView.

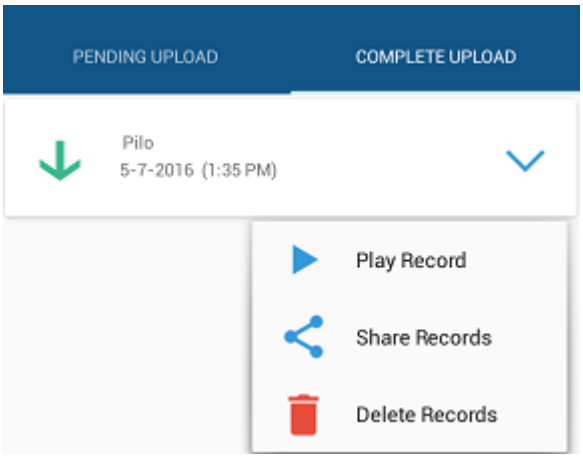
```
public class MyViewHolder extends RecyclerView.ViewHolder{
        CardView cv;
        TextView recordName, visibleFile, date, time;
        Button btnIn, btnExpand;

        public MyViewHolder(final View itemView) {
            super(itemView);

            cv = (CardView)itemView.findViewById(R.id.cardview);
            recordName = (TextView)itemView.findViewById(R.id.tv_record);
            visibleFile = (TextView)itemView.findViewById(R.id.visible_file);
            date = (TextView)itemView.findViewById(R.id.date);
            time = (TextView)itemView.findViewById(R.id.time);
            btnIn = (Button)itemView.findViewById(R.id.btn_in_out);

            btnExpand = (Button) itemView.findViewById(R.id.btn_expand);

            btnExpand.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    PopupMenu popup = new PopupMenu(btnExpand.getContext(), itemView);

                    popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
                        @Override
                        public boolean onMenuItemClick(MenuItem item) {
                            switch (item.getItemId()) {
                                case R.id.action_delete:
                                    moveFile(recordName.getText().toString(),
getAdapterPosition());
                                    return true;
                                case R.id.action_play:
                                    String valueOfPath = recordName.getText().toString();
                                    Intent intent = new Intent();
                                    intent.setAction(android.content.Intent.ACTION_VIEW);
                                    File file = new File(valueOfPath);
                                    intent.setDataAndType(Uri.fromFile(file), "audio/*");
                                    context.startActivity(intent);
                                    return true;
                                case R.id.action_share:
                                    String valueOfPath = recordName.getText().toString();
                                    File filee = new File(valueOfPath);
                                    try {
                                        Intent sendIntent = new Intent();
                                        sendIntent.setAction(Intent.ACTION_SEND);
                                        sendIntent.setType("audio/*");
                                        sendIntent.putExtra(Intent.EXTRA_STREAM,
Uri.fromFile(filee));
                                        context.startActivity(sendIntent);
```

```
                                    } catch (NoSuchMethodError | IllegalArgumentException |
NullPointerException e) {
                                        e.printStackTrace();
                                    } catch (Exception e) {
                                        e.printStackTrace();
                                    }
                                    return true;
                                default:
                                    return false;
                            }
                        }
                    });
                    // here you can inflate your menu
                    popup.inflate(R.menu.my_menu_item);
                    popup.setGravity(Gravity.RIGHT);

                    // if you want icon with menu items then write this try-catch block.
                    try {
                        Field mFieldPopup=popup.getClass().getDeclaredField("mPopup");
                        mFieldPopup.setAccessible(true);
                        MenuPopupHelper mPopup = (MenuPopupHelper) mFieldPopup.get(popup);
                        mPopup.setForceShowIcon(true);
                    } catch (Exception e) {

                    }
                    popup.show();
                }
            });

        }
    }
```

альтернативный способ отображения значков в меню

```
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) {

}
```

Вот результат:



Анимация изменения данных

`RecyclerView` будет выполнять соответствующую анимацию, если используется какой-либо из методов «уведомлять», за исключением `notifyDataSetChanged` ; это включает `notifyItemChanged` , `notifyItemInserted` , `notifyItemMoved` , `notifyItemRemoved` и т. д.

Адаптер должен расширить этот класс вместо `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);
            }
        }
    }
}
```

Вы **не** должны использовать один и тот же `List` для `setModels` и `List` в адаптере.

Вы объявляете `models` глобальными переменными. `DataModel` - только фиктивный класс.

```
private List<DataModel> models;
private YourAdapter adapter;
```

Инициализируйте `models` перед тем, как передать их адаптеру. `YourAdapter` - это реализация `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));
```

Вызовите это после обновления ваших глобальных `models` .

```
adapter.setModels(new ArrayList(models));
```

Если вы не переопределяете `equals` , сравнение сравнивается по ссылке.

Пример использования SortedList

Android представил класс `SortedList` вскоре после появления `RecyclerView` . Этот класс обрабатывает все вызовы метода «уведомлять» в `RecyclerView.Adapter` для обеспечения правильной анимации и даже позволяет выполнять многократные изменения, поэтому анимации не дрожат.

```
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 с DataBinding

Вот общий класс ViewHolder, который можно использовать с любым макетом DataBinding. Здесь экземпляр определенного класса `ViewDataBinding` создается с использованием `ViewDataBinding` класса `View` и `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;
    }
}
```

После создания этого класса вы можете использовать `<layout>` в вашем файле макета, чтобы включить привязку данных для этого макета следующим образом:

`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>
```

и вот ваш образец dataModel:

```
public class ItemModel {
    public String itemLabel;
}
```

По умолчанию библиотека `ViewDataBinding` данных Android генерирует класс `ViewDataBinding` на основе имени файла макета, преобразуя его в регистр Pascal и суффикс «привязки» к нему. В этом примере это будет `MyItemBinding` для файла макета `my_item.xml` . Этот класс Binding также будет иметь метод setter, чтобы установить объект, определенный как данные в файле макета (`ItemModel` для этого примера).

Теперь, когда у нас есть все части, мы можем реализовать наш адаптер следующим образом:

```
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();
    }
}
```

Бесконечная прокрутка в Recycleview.

Здесь я поделился фрагментом кода для реализации бесконечной прокрутки в режиме просмотра.

Шаг 1. Сначала сделайте один абстрактный метод в адаптере Recycleview, как показано

ниже.

```
public abstract class ViewAllCategoryAdapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    public abstract void load();
}
```

Шаг 2: Теперь переопределите метод onBindViewHolder и `getItemCount()` класса ViewAllCategoryAdapter и вызовите метод `Load()` как показано ниже.

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, final int position) {
    if ((position >= getItemCount() - 1)) {
        load();
    }
}

@Override
public int getItemCount() {
    return YOURLIST.size();
}
```

Шаг 3. Теперь каждая логика бэкэнд завершена, теперь пришло время выполнить эту логику. Просто вы можете переопределить метод загрузки, когда вы создаете объект вашего адаптера. Этот метод автоматически вызывается, пока пользователь достигает в конце списка.

```
  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);
```

Теперь метод `load()` автоматически вызывается во время прокрутки пользователя в конце списка.

Лучшая удача

Показывать представление по умолчанию до загрузки элементов или когда данные недоступны

Скриншот

Класс адаптера

```
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>
```

Я использую FontAwesome с библиотекой Iconics для изображений. Добавьте это в файл build.gradle на уровне приложения.

```
compile 'com.mikepenz:fontawesome-typeface:4.6.0.3@aar'
compile 'com.mikepenz:iconics-core:2.8.1@aar'
```

Добавление разделительных линий в элементы RecyclerView

Просто добавьте эти строки в инициализацию

```
RecyclerView mRecyclerView = (RecyclerView) view.findViewById(recyclerView);
mRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
mRecyclerView.addItemDecoration(new DividerItemDecoration(getActivity(),
DividerItemDecoration.VERTICAL));
```

Добавьте `adapter` и вызовите `.notifyDataSetChanged();` по-прежнему !
Это не встроенная функция Recyclerview, но добавлена в библиотеки поддержки. Поэтому не забудьте включить это в свой файл build.gradle на уровне приложения

```
compile "com.android.support:appcompat-v7:25.3.1"
compile "com.android.support:recyclerview-v7:25.3.1"
```

> Несколько элементов ItemDecorations могут быть добавлены в один RecyclerView.

Изменение цвета разделителя :

Очень легко установить цвет для украшения предметов.

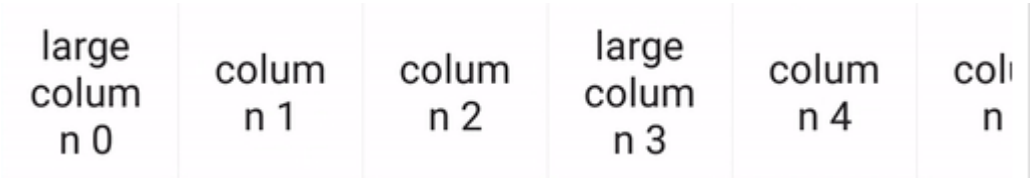
1. шаг: создание `divider.xml` файл , который находится на `drawable` папке

```
<?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>
```

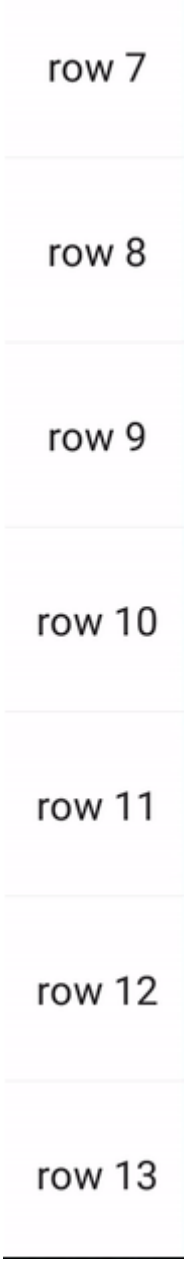
2. шаг: установка возможности рисования

```
    // 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);
```



Прочитайте RecyclerView онлайн: https://riptutorial.com/ru/android/topic/169/recyclerview

глава 69: RecyclerView onClickListeners

Examples

Новый пример

```
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);
    }
}
```

Примеры Kotlin и RxJava

Первый пример переопределяется в Котлине и использует RxJava для более чистого взаимодействия.

```
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
        }
    }
}
```

Использование довольно просто. Можно подписаться на отдельную тему, используя средства RxJava.

```
val adapter = SampleAdapter(arrayOf("Hello", "World"))
adapter.itemClickStream.subscribe { v ->
    if (v.id == R.id.textView) {
        // do something
    }
}
```

Easy OnLongClick и пример OnClick

Прежде всего, реализуйте свое мнение:

```
implements View.OnClickListener, View.OnLongClickListener
```

Затем зарегистрируйте слушателей следующим образом:

```
itemView.setOnClickListener(this);
itemView.setOnLongClickListener(this);
```

Затем переопределите слушателей следующим образом:

```
@Override
public void onClick(View v) {
    onclicklistner.onItemClick(getAdapterPosition(), v);
}

@Override
public boolean onLongClick(View v) {
    onclicklistner.onItemLongClick(getAdapterPosition(), v);
    return true;
}
```

И, наконец, добавьте следующий код:

```
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);
}
```

Демо-версия адаптера

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

Вышеприведенный код может быть вызван следующим кодом:

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

Элемент Click Listeners

Чтобы внедрить прослушиватель элементов и / или прослушиватель длинного клика элемента, вы можете создать интерфейс в своем адаптере:

```
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;
    }

}
```

Затем вы можете настроить прослушиватели кликов после создания экземпляра адаптера:

```
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;
    }
});
```

Еще один способ реализации элемента прослушивания элемента

Еще один способ внедрения элемента прослушивателя элементов - использовать интерфейс с несколькими методами, число которых равно числу доступных для просмотра представлений, и использовать переопределенные прослушиватели кликов, как вы можете видеть ниже. Этот метод более гибкий, потому что вы можете настроить прослушиватели кликов на разные виды и достаточно легко управлять логикой щелчка отдельно для каждого.

```
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);
        }
    }
}
```

И когда у вас есть экземпляр адаптера, вы можете настроить прослушиватель кликов, который прослушивает щелчок по каждому из видов:

```
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 прослушиватель

```
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);
    }
}
```

В Главной Активности

```
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) {


     }
    }));
```

Прочитайте RecyclerView onClickListeners онлайн:
https://riptutorial.com/ru/android/topic/96/recyclerview-onclicklisteners

глава 70: RecyclerView и LayoutManagers

Examples

GridLayoutManager с динамическим числом периодов

При создании recyclerview с менеджером компоновки gridlayout вам нужно указать счетчик span в конструкторе. Span count относится к числу столбцов. Это довольно неуклюже и не учитывает больший размер экрана или ориентацию экрана. Один из подходов - создать несколько макетов для различных размеров экрана. Еще один более динамичный подход можно увидеть ниже.

Сначала мы создаем собственный класс RecyclerView следующим образом:

```
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);
        }
```

```
    }
}
```

Этот класс определяет, сколько столбцов может поместиться в recyclerview. Чтобы использовать его, вам нужно поместить его в свой макет. Xml следующим образом:

```
<?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"
    />
```

Обратите внимание, что мы используем атрибут columnWidth. Рециркулятор будет нуждаться в нем, чтобы определить, сколько столбцов поместится в доступное пространство.

В вашей деятельности / фрагменте вы просто получаете ссылку на recylerview и устанавливаете для него адаптер (и любые украшения или анимации предметов, которые вы хотите добавить). **НЕ УСТАНАВЛИВАЙТЕ МЕНЕДЖЕРА**

```
RecyclerView recyclerView = (RecyclerView) findViewById(R.id.auto_fit_recycler_view);
recyclerView.setAdapter(new MyAdapter());
```

(где MyAdapter - ваш класс адаптера)

Теперь у вас есть recyclerview, который будет настраивать spancount (т.е. столбцы), чтобы соответствовать размеру экрана. В качестве окончательного добавления вы можете захотеть центрировать столбцы в recyclerview (по умолчанию они выровнены с layout_start). Вы можете это сделать, немного изменив класс AutofitRecyclerView. Начните с создания внутреннего класса в recyclerview. Это будет класс, который простирается от GridLayoutManager. Он добавит достаточно отступов влево и вправо, чтобы центрировать строки:

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

Затем, когда вы устанавливаете LayoutManager в AutofitRecyclerView, используйте CenteredGridLayoutManager следующим образом:

```
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);
}
```

И это все! У вас есть динамический spancount, выравниваемый по центру gridlayoutmanager, основанный на recyclerview.

Источники:

- Блог пользователя Chiu-Ki Chan Square Island
- Переполнение стека

Добавление заголовка в recyclerview с помощью менеджера gridlayout

Чтобы добавить заголовок к recyclerview с помощью gridlayout, сначала нужно сказать, что для заголовка заголовка является первая позиция, а не стандартная ячейка, используемая для содержимого. Затем менеджеру компоновки должно быть сказано, что первая позиция

должна иметь диапазон, равный счету * span всего списка. *

Возьмите обычный класс RecyclerView.Adapter и настройте его следующим образом:

```
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
    }
}
```

Затем в активности / фрагменте:

```
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);
```

Тот же подход может быть использован для добавления нижнего колонтитула в дополнение к заголовку или вместо него.

Источник: блог на острове Чиу-Ки Чан

Простой список с LinearLayoutManager

В этом примере добавляется список мест с изображением и именем с использованием `ArrayList` пользовательских объектов `Place` качестве набора данных.

Схема действий

Макет действия / фрагмента или где используется RecyclerView, должен содержать RecyclerView. Нет ScrollView или определенного макета.

```
<?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>
```

Определить модель данных

Вы можете использовать любой класс или примитивный тип данных в качестве модели, например, `int` , `String` , `float[]` или `CustomObject` . RecyclerView будет ссылаться на `List` этих объектов / примитивов.

Когда элемент списка относится к различным типам данных, таким как текст, цифры, изображения (как в этом примере с местами), часто рекомендуется использовать пользовательский объект.

```
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;
    }
}
```

Элемент списка элементов

Вы должны указать файл макета xml, который будет использоваться для каждого элемента списка. В этом примере для изображения используется `ImageView` и `TextView` для имени. `LinearLayout` позиционирует `ImageView` слева, а `TextView` прямо к изображению.

```
<?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>
```

Создайте адаптер RecyclerView и ViewHolder

Затем вам нужно наследовать `RecyclerView.Adapter` и `RecyclerView.ViewHolder` . Обычная

структура классов:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    public class ViewHolder extends RecyclerView.ViewHolder {
        // ...
    }
}
```

Во-первых, мы реализуем `ViewHolder` . Он только наследует конструктор по умолчанию и сохраняет необходимые представления в некоторых полях:

```
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);
    }
}
```

Конструктор адаптера устанавливает используемый набор данных:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    private List<Place> mPlaces;

    public PlaceListAdapter(List<Place> contacts) {
        mPlaces = contacts;
    }

    // ...
}
```

Чтобы использовать наш макет элемента настраиваемого списка, мы переопределяем метод `onCreateViewHolder(...)` . В этом примере файл макета называется `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);
    }

    // ...
}
```

В `onBindViewHolder(...)` мы фактически устанавливаем содержимое представлений. Мы получаем использованную модель, находя ее в `List` в данной позиции, а затем `ViewHolder` изображение и имя на представлениях `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());
    }

    // ...
}
```

Нам также необходимо реализовать `getItemCount()` , которые просто возвращают размер `List` .

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public int getItemCount() {
        return mPlaces.size();
    }

    // ...
}
```

(Генерировать случайные данные)

В этом примере мы создадим некоторые случайные места.

```
@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;
}
```

Подключите RecyclerView с помощью PlaceListAdapter и набора данных

Подключение `RecyclerView` с адаптером очень просто. Вы должны установить `LinearLayoutManager` качестве менеджера макета для достижения макета списка.

```
@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));
}
```

Готово!

StaggeredGridLayoutManager

1. Создайте свой RecyclerView в вашем XML-файле макета:

```
<android.support.v7.widget.RecyclerView
            android:id="@+id/recycleView"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
```

2. Создайте класс модели для хранения ваших данных:

```
     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. Создайте файл макета для хранения элементов 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. Создайте класс адаптера для 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. Выполните активацию RecyclerView в вашей деятельности или фрагменте:

```
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() );
```

Не забудьте добавить зависимость Picasso в файле build.gradle:

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

Прочитайте RecyclerView и LayoutManagers онлайн:
https://riptutorial.com/ru/android/topic/6772/recyclerview-и-layoutmanagers

глава 71: Renderscript

Вступление

RenderScript - это язык сценариев, который позволяет писать высокопроизводительный графический рендеринг и исходный вычислительный код. Он предоставляет средства для написания критического кода производительности, который система затем компилирует в собственный код для процессора, на котором он может работать. Это может быть процессор, многоядерный процессор или даже графический процессор. В конечном счете это зависит от многих факторов, которые не всегда доступны разработчику, но также зависит от архитектуры, которую поддерживает внутренний компилятор платформы.

Examples

Начиная

RenderScript - это среда для высокопроизводительных параллельных вычислений на Android. Сценарии, которые вы пишете, будут выполняться на всех доступных процессорах (например, CPU, GPU и т. Д.) Параллельно, позволяя вам сосредоточиться на задаче, которую вы хотите достичь, а не на том, как она запланирована и выполнена.

Сценарии написаны на языке C99 (C99 - старая версия стандарта языка программирования C). Для каждого скрипта создается Java-класс, который позволяет вам легко взаимодействовать с RenderScript в вашем Java-коде.

Настройка вашего проекта

Существуют два разных способа доступа к RenderScript в приложении: библиотеки Android Framework или Библиотека поддержки. Даже если вы не хотите настраивать таргетинг на устройства до уровня API 11, вы всегда должны использовать реализацию библиотеки поддержки, поскольку он обеспечивает совместимость устройств на разных устройствах. Чтобы использовать реализацию библиотеки поддержки, вам нужно использовать хотя бы инструменты для сборки версии `18.1.0` !

Теперь давайте настроим файл build.gradle вашего приложения:

```
android {
    compileSdkVersion 24
    buildToolsVersion '24.0.1'

    defaultConfig {
        minSdkVersion 8
        targetSdkVersion 24
```

```
        renderscriptTargetApi 18
        renderscriptSupportModeEnabled true
    }
}
```

- `renderscriptTargetApi` : этот параметр должен быть установлен на самый ранний уровень API версии, который предоставляет все необходимые функции RenderScript.
- `renderscriptSupportModeEnabled` : Это позволяет использовать реализацию RenderScript библиотеки поддержки.

Как работает RenderScript

Типичный RenderScript состоит из двух вещей: ядра и функции. Функция - это то, на что это похоже - она принимает вход, делает что-то с этим входом и возвращает результат. Ядро - это реальная сила RenderScript.

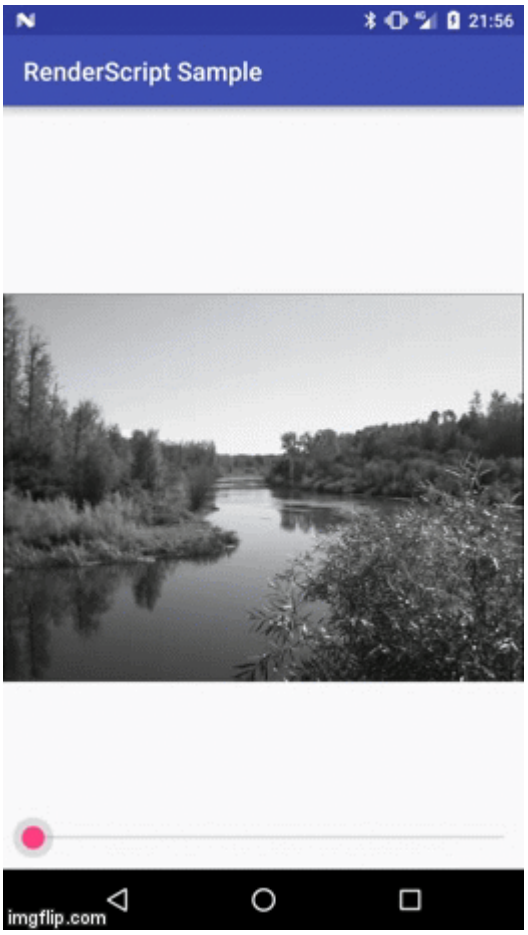
Ядро - это функция, выполняемая против каждого элемента внутри `Allocation` . `Allocation` может использоваться для передачи данных, таких как `Bitmap` или `byte` массив, в `RenderScript` и они также используются для получения результата из ядра. Ядра могут либо принимать одно `Allocation` качестве входных данных, либо другое как вывод, или они могут изменять данные внутри всего одного `Allocation` .

Вы можете написать одно ядро, но есть также множество предопределенных ядер, которые вы можете использовать для выполнения общих операций, таких как Gaussian Image Blur.

Как уже упоминалось для каждого файла RenderScript, для него взаимодействует класс. Эти классы всегда начинаются с префикса `ScriptC_` за которым следует имя файла RenderScript. Например, если ваш RenderScript-файл называется `example` тогда сгенерированный класс Java будет называться `ScriptC_example` . Все предопределенные скрипты начинаются с префикса `Script` - например, Gaussian Image Blur Script называется `ScriptIntrinsicBlur` .

Написание первого RenderScript

Следующий пример основан на примере GitHub. Он выполняет базовые манипуляции с изображениями, изменяя насыщенность изображения. Вы можете найти исходный код **здесь** и проверить его, если вы хотите поиграть с ним самостоятельно. Вот краткое описание того, как должен выглядеть результат:



RilerScript Boilerplate

Файлы RenderScript находятся в папке `src/main/rs` в вашем проекте. Каждый файл имеет расширение файла `.rs` и должен содержать два оператора `#pragma` в верхней части:

```
#pragma version(1)
#pragma rs java_package_name(your.package.name)
```

- `#pragma version(1)` : Это можно использовать для установки версии RenderScript, которую вы используете. В настоящее время существует только версия 1.
- `#pragma rs java_package_name(your.package.name)` : это может использоваться для установки имени пакета класса Java, сгенерированного для взаимодействия с этим конкретным RenderScript.

Существует еще одна # `#pragma` вы обычно должны устанавливать в каждом из ваших файлов RenderScript, и используется для установки точности с плавающей запятой. Вы можете установить точность с плавающей запятой на три разных уровня:

- `#pragma rs_fp_full` : это самый строгий параметр с максимальной точностью, а также значение по умолчанию, если ничего не указывать. Вы должны использовать это, если вам нужна высокая точность с плавающей точкой.
- `#pragma rs_fp_relaxed` : Это обеспечивает не совсем высокую точность с плавающей

запятой, но на некоторых архитектурах она позволяет кучу оптимизаций, которые могут привести к тому, что ваши скрипты будут работать быстрее.
- `#pragma rs_fp_imprecise` : Это обеспечивает еще меньшую точность и должно использоваться, если точность с плавающей запятой не имеет особого значения для вашего скрипта.

Большинство скриптов могут просто использовать `#pragma rs_fp_relaxed` если вам действительно не нужна высокая точность с плавающей точкой.

Глобальные переменные

Теперь, как и в коде C, вы можете определить глобальные переменные или константы:

```
const static float3 gMonoMult = {0.299f, 0.587f, 0.114f};

float saturationLevel = 0.0f;
```

Переменная `gMonoMult` имеет тип `float3` . Это означает, что это вектор, состоящий из 3 чисел с плавающей точкой. Другая переменная `float` называемая `saturationValue` , не является постоянной, поэтому вы можете установить ее во время выполнения на нужное вам значение. Вы можете использовать такие переменные в своих ядрах или функциях, и поэтому они являются другим способом ввода или получения результатов из ваших RenderScripts. Для каждой не постоянной переменной метод getter и setter будет сгенерирован в соответствующем Java-классе.

Ядра

Но теперь давайте начнем внедрять Ядро. Для целей этого примера я не буду объяснять математику, используемую в ядре, чтобы изменить насыщенность изображения, но вместо этого сосредоточимся на том, как реализовать Ядро и как его использовать. В конце этой главы я быстро объясню, что на самом деле делает код в этом ядре.

Ядра в целом

Давайте сначала посмотрим на исходный код:

```
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);
}
```

Как вы видите, это выглядит как обычная функция C с одним исключением: `__attribute__((kernel))` между типом возвращаемого значения и именем метода. Это то, что

говорит RenderScript, что этот метод является ядром. Другое дело, что вы можете заметить, что этот метод принимает параметр `uchar4` и возвращает другое значение `uchar4` . `uchar4` - как переменная `float3` мы обсуждали в главе раньше - вектор. Он содержит 4 значения `uchar` которые являются только байтовыми значениями в диапазоне от 0 до 255.

Вы можете получить доступ к этим отдельным значениям различными способами, например `in.r` вернет байт, соответствующий красному каналу пикселя. Мы используем `uchar4` , поскольку каждый пиксель состоит из 4 значений - `r` для красного, `g` для зеленых, `b` для синего и для альфа - и вы можете получить доступ к ним с этим стенографией. `a` RenderScript также позволяет вам принимать любое количество значений из вектора и создавать с ними другой вектор. Например, `in.rgb` вернет значение `uchar3` которое просто содержит красную, зеленую и синюю части пикселя без альфа-значения.

Во время выполнения RenderScript вызовет этот метод Kernel для каждого пикселя изображения, поэтому возвращаемое значение и параметр являются только одним значением `uchar4` . RenderScript будет запускать многие из этих вызовов параллельно на всех доступных процессорах, поэтому RenderScript настолько мощный. Это также означает, что вам не нужно беспокоиться о потоковой или потоковой безопасности, вы можете просто реализовать все, что хотите сделать для каждого пикселя, и RenderScript позаботится об остальном.

При вызове ядра на Java вы предоставляете две переменные `Allocation` , которые содержат входные данные, и другую, которая будет получать результат. Ваш метод Kernel будет вызываться для каждого значения во входном `Allocation` и будет записывать результат в Output `Allocation` .

Методы API RenderScript Runtime

В ядре выше используется несколько методов, которые предоставляются из коробки. RenderScript предоставляет множество таких методов, и они жизненно важны для всего, что вы собираетесь делать с RenderScript. Среди них методы для математических операций, таких как `sin()` и вспомогательные методы, такие как `mix()` который смешивает два значения в соответствии с другими значениями. Но существуют также методы более сложных операций при работе с векторами, кватернионами и матрицами.

Официальная **ссылка Runtime Runtime API Reference** является лучшим ресурсом, если вы хотите узнать больше о конкретном методе или ищете конкретный метод, который выполняет общую операцию, например вычисление точечного произведения матрицы. Вы можете найти эту документацию **здесь** .

Реализация ядра

Теперь давайте рассмотрим особенности того, что делает это Ядро. Вот первая строка в ядре:

```
float4 f4 = rsUnpackColor8888(in);
```

Первая строка вызывает встроенный метод **`rsUnpackColor8888()`** , который преобразует `uchar4` значение до `float4` значений. Каждый цветной канал также преобразуется в диапазон `0.0f - 1.0f` где `0.0f` соответствует `0.0f` значению `0` и от `1.0f` до `255` . Основная цель этого - сделать математику в этом ядре намного проще.

```
float3 dotVector = dot(f4.rgb, gMonoMult);
```

В следующей строке используется встроенный метод **`dot()`** для вычисления точечного произведения двух векторов. `gMonoMult` - это постоянное значение, которое мы определили в нескольких главах выше. Поскольку оба вектора должны иметь одинаковую длину для вычисления точечного произведения, а также, поскольку мы просто хотим влиять на цветовые каналы, а не на альфа-канал пикселя, мы используем сокращенное `.rgb` для получения нового вектора `float3` который просто содержит красный, зеленый и синий. Те из нас, кто все еще помнит из школы, как работает точечный продукт, быстро заметят, что точечный продукт должен возвращать только одно значение, а не вектор. Однако в приведенном выше коде мы присваиваем результат вектору `float3` . Это опять-таки особенность RenderScript. Когда вы назначаете одномерное число вектору, все элементы в векторе будут установлены в это значение. Например, следующий фрагмент присваивает `2.0f` каждому из трех значений в векторе `float3` :

```
float3 example = 2.0f;
```

Таким образом, результат указанного выше точечного продукта присваивается каждому элементу в `float3` элементе `float3` .

Теперь наступает часть, в которой мы фактически используем глобальную переменную `saturationLevel` для изменения насыщенности изображения:

```
float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
```

Это использует встроенный метод **`mix()`** для смешивания исходного цвета с вектором продукта dot, который мы создали выше. Как они смешиваются вместе, определяется глобальной переменной `saturationLevel` . Таким образом, `saturationLevel` `0.0f` приведет к тому, что результирующий цвет не будет иметь `0.0f` значений цвета и будет состоять только из значений в `dotVector` что приведет к получению черно-белого или серого изображения. Значение `1.0f` приведет к тому, что полученный цвет будет полностью составлен из исходных значений цвета, а значения выше `1.0f` будут умножать исходные цвета, чтобы сделать их более яркими и интенсивными.

```
return rsPackColorTo8888(newColor);
```

Это последняя часть в ядре. **`rsPackColorTo8888()`** преобразует вектор `float3` обратно в

значение `uchar4` которое затем возвращается. Результирующие байтовые значения зажимаются в диапазоне от 0 до 255, поэтому значения float выше `1.0f` приведут к `1.0f` значению 255, а значения ниже `0.0` приведут к байтовому значению `0` .

И это целая реализация ядра. Теперь остается только одна часть: Как вызвать ядро на Java.

Вызов RenderScript в Java

основы

Как уже упоминалось выше для каждого файла RenderScript генерируется класс Java, который позволяет вам взаимодействовать с вашими сценариями. Эти файлы имеют префикс `ScriptC_` за которым следует имя файла RenderScript. Чтобы создать экземпляр этих классов, вам сначала понадобится экземпляр класса `RenderScript` :

```
final RenderScript renderScript = RenderScript.create(context);
```

Статический метод `create()` может использоваться для создания экземпляра `RenderScript` из `Context` . Затем вы можете создать экземпляр класса Java, который был сгенерирован для вашего скрипта. Если вы вызвали файл `saturation.rs` файла RenderScript, класс будет называться `ScriptC_saturation` :

```
final ScriptC_saturation script = new ScriptC_saturation(renderScript);
```

В этом классе вы можете установить уровень насыщенности и вызвать ядро. Установщик, который был сгенерирован для переменной `saturationLevel` будет иметь префикс `set_` за которым следует имя переменной:

```
script.set_saturationLevel(1.0f);
```

Также есть getter с префиксом `get_` который позволяет вам получить уровень насыщенности, установленный в настоящий момент:

```
float saturationLevel = script.get_saturationLevel();
```

Ядра, которые вы определяете в своем RenderScript, имеют префикс `forEach_` за которым следует имя метода Kernel. Ядро, которое мы написали, ожидает, что в качестве параметров ввода и `Allocation` выходов будет `Allocation` :

```
script.forEach_saturation(inputAllocation, outputAllocation);
```

Входное `Allocation` должно содержать входное изображение, и после `forEach_saturation`

метода `forEach_saturation` распределение вывода будет содержать измененные данные изображения.

После того, как у вас есть экземпляр `Allocation` вы можете скопировать данные из этих `Allocations` с помощью этих методов с помощью методов `copyFrom()` и `copyTo()` . Например, вы можете скопировать новое изображение в свой ввод «Выделение путем вызова:

```
inputAllocation.copyFrom(inputBitmap);
```

Точно так же вы можете получить результат изображения, вызвав `copyTo()` на выходе `Allocation` :

```
outputAllocation.copyTo(outputBitmap);
```

Создание экземпляров распределения

Существует много способов создания `Allocation` . После того, как у вас есть экземпляр `Allocation` вы можете скопировать новые данные и с этими `Allocations` с помощью `copyTo()` и `copyFrom()` как описано выше, но для их создания изначально вы должны знать, с какими данными вы работаете. Начнем с ввода `Allocation` :

Мы можем использовать статический метод `createFromBitmap()` чтобы быстро создать наш вход. `Allocation` из `Bitmap` :

```
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, image);
```

В этом примере входное изображение никогда не изменяется, поэтому нам больше не нужно снова изменять `Allocation` ввода. Мы можем повторно использовать его каждый раз, когда `saturationLevel` изменяется для создания нового выходного `Bitmap` .

Создание вывода `Allocation` немного сложнее. Сначала нам нужно создать так называемый `Type` . `Type` используется, чтобы рассказать о `Allocation` с какими данными он имеет дело. Обычно используется класс `Type.Builder` для быстрого создания соответствующего `Type` . Давайте сначала посмотрим на код:

```
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
        .setX(inputBitmap.getWidth())
        .setY(inputBitmap.getHeight())
        .create();
```

Мы работаем с обычным 32-битным (или, другими словами, 4 байтами) на пиксель `Bitmap` с 4-мя цветными каналами. Вот почему мы выбираем `Element.RGBA_8888` для создания `Type` . Затем мы используем методы `setX()` и `setY()` чтобы установить ширину и высоту выходного изображения того же размера, что и входное изображение. Затем метод `create()` создает `Type` с указанными параметрами.

Как только у нас будет правильный `Type` мы можем создать выход `Allocation` со статическим методом `createTyped()` :

```
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);
```

Теперь мы почти закончили. Нам также нужен выходной `Bitmap` в котором мы можем скопировать данные из вывода `Allocation` . Для этого мы используем статический метод `createBitmap()` для создания нового пустого `Bitmap` с тем же размером и конфигурацией, что и входной `Bitmap` .

```
final Bitmap outputBitmap = Bitmap.createBitmap(
        inputBitmap.getWidth(),
        inputBitmap.getHeight(),
        inputBitmap.getConfig()
);
```

И вместе с этим у нас есть все части головоломки для выполнения нашего RenderScript.

Полный пример

Теперь давайте соединим все это в одном примере:

```
// 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);
```

Заключение

С этим введением вы должны быть готовы написать свои собственные ядра RenderScript для простой обработки изображений. Однако есть несколько вещей, которые вы должны иметь в виду:

- **RenderScript работает только в проектах приложений** : в настоящее время файлы RenderScript не могут быть частью проекта библиотеки.
- **Следите за памятью** : RenderScript работает очень быстро, но он также может быть интенсивным в памяти. В любой момент никогда не должно быть более одного экземпляра `RenderScript` . Вы должны также использовать его как можно больше. Обычно вам просто нужно создать экземпляры « `Allocation` один раз и использовать их в будущем. То же самое касается выходных `Bitmaps` или экземпляров скриптов. Повторно используйте как можно больше.
- **Сделайте свою работу в фоновом режиме** : снова RenderScript работает очень быстро, но не мгновенно. Любое Ядро, особенно сложные, должно быть выполнено из потока пользовательского интерфейса в `AsyncTask` или что-то подобное. Однако по большей части вам не нужно беспокоиться о утечке памяти. Все классы, связанные с RenderScript, используют только `Context` приложения и, следовательно, не вызывают утечки памяти. Но вам все равно придется беспокоиться о обычных вещах, таких как утечка `View` , `Activity` или любого экземпляра `Context` который вы используете сами!
- **Используйте встроенный материал** : существует множество предопределенных скриптов, которые выполняют такие задачи, как размытие изображения, смешивание, преобразование, изменение размера. И есть еще много встроенных методов, которые помогут вам реализовать свои ядра. Скорее всего, если вы хотите что-то сделать, есть либо скрипт, либо метод, который уже делает то, что вы пытаетесь сделать. Не изобретайте велосипед.

Если вы хотите быстро начать работу и поиграть с фактическим кодом, я рекомендую вам взглянуть на пример проекта GitHub, который реализует точный пример, о котором говорится в этом уроке. Вы можете найти проект **здесь** . Получайте удовольствие от RenderScript!

Размытие изображения

В этом примере показано, как использовать Renderscript API для размытия изображения (с использованием растрового изображения). В этом примере используется ScriptInstrinsicBlur, предоставляемый API 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;
    }
}
```

Каждый скрипт имеет ядро, которое обрабатывает данные и обычно вызывается методом `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
    }
}
```

Здесь был приведен пример. Рекомендуется выполнять обработку в фоновом потоке.

Размытие

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;
    }
}
```

Использование:

```
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);
```

Прочитайте Renderscript онлайн: https://riptutorial.com/ru/android/topic/5214/renderscript

глава 72: Retrofit2

Вступление

Официальная страница Retrofit описывает себя как

Типичный клиент REST для Android и Java.

Retrofit превращает ваш REST API в интерфейс Java. Он использует аннотации для описания HTTP-запросов, по умолчанию заменяет параметр URL и поддержку параметров запроса. Кроме того, он обеспечивает функциональные возможности для купирования многостраничного запроса и загрузки файлов.

замечания

Зависимости для модифицированной библиотеки:

Из официальной документации :

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

Простой запрос GET

Мы собираемся показать, как сделать запрос `GET` API, который отвечает с помощью объекта `JSON` или массива `JSON` . Первое, что нам нужно сделать, это добавить зависимости Retrofit и `GSON` Converter к файлу градиента нашего модуля.

Добавьте зависимости для модифицированной библиотеки, как описано в разделе

«Примечания».

Пример ожидаемого объекта JSON:

```
{
    "deviceId": "56V56C14SF5B4SF",
    "name": "Steven",
    "eventCount": 0
}
```

Пример массива JSON:

```
[
    {
        "deviceId": "56V56C14SF5B4SF",
        "name": "Steven",
        "eventCount": 0
    },
    {
        "deviceId": "35A80SF3QDV7M9F",
        "name": "John",
        "eventCount": 2
    }
]
```

Пример соответствующего класса модели:

```
public class Device
{
    @SerializedName("deviceId")
    public String id;

    @SerializedName("name")
    public String name;

    @SerializedName("eventCount")
    public int eventCount;
}
```

Аннотации `@SerializedName` представлены в библиотеке `GSON` и позволяют `serialize` и `deserialize` этот класс в `JSON` используя сериализованное имя в качестве ключей. Теперь мы можем построить интерфейс для API, который будет фактически извлекать данные с сервера.

```
public interface DeviceAPI
{
    @GET("device/{deviceId}")
    Call<Device> getDevice (@Path("deviceId") String deviceID);

    @GET("devices")
    Call<List<Device>> getDevices();
}
```

Здесь очень много места в довольно компактном пространстве, поэтому давайте

сломаем его:

- Аннотации `@GET` поступают из Retrofit и `@GET` библиотеке, что мы определяем запрос GET.
- Путь в круглых скобках - это конечная точка, на которую должен наступить наш запрос GET (мы установим базовый url чуть позже).
- Скопированные скобки позволяют нам заменять часть пути во время выполнения, чтобы мы могли передавать аргументы.
- Функция, которую мы определяем, называется `getDevice` и принимает идентификатор устройства, который мы хотим в качестве аргумента.
- `@PATH` сообщает Retrofit, что этот аргумент должен заменить местозаполнитель « deviceId» в пути.
- Функция возвращает объект `Call` типа `Device` .

Создание класса-оболочки:

Теперь мы создадим небольшой класс-оболочку для нашего API, чтобы сохранить код инициализации Retrofit, завершенный красиво.

```
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);
    }
}
```

Этот класс создает экземпляр GSON, чтобы иметь возможность анализировать ответ JSON, создает экземпляр Retrofit с нашим базовым url и GSONConverter, а затем создает экземпляр нашего API.

Вызов 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());
    }
});
```

Это использует наш интерфейс API для создания объекта `Call<Device>` и для создания `Call<List<Device>>` соответственно. Вызов `enqueue` сообщает Retrofit, чтобы сделать этот вызов в фоновом потоке и вернуть результат в обратный вызов, который мы здесь создаем.

Примечание. Анализ массива примитивных объектов JSON (например, *String, Integer, Boolean* и *Double*) аналогичен анализу массива JSON. Однако вам не нужен ваш собственный класс модели. Вы можете получить массив строк, например, с типом возвращаемого вызова как `Call<List<String>>` .

Добавить запись в Retrofit2

Запросы дооснащения могут регистрироваться с использованием интерсептера. Доступны несколько уровней детализации: NONE, BASIC, HEADERS, BODY. См. Проект Github здесь .

1. Добавить зависимость для build.gradle:

```
compile 'com.squareup.okhttp3:logging-interceptor:3.8.1'
```

2. Добавить регистрационный перехватчик при создании 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();
```

Экспозиция журналов в терминале (Android Monitor) - это то, чего следует избегать в версии выпуска, поскольку это может привести к нежелательному раскрытию важной информации, такой как Auth Tokens и т. Д.

Чтобы избежать появления журналов во время выполнения, проверьте следующее условие

```
if(BuildConfig.DEBUG){
     //your interfector code here
    }
```

Например:

```
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();
```

Загрузка файла через Multipart

Объявите свой интерфейс с помощью аннотаций Retrofit2:

```
public interface BackendApiClient {
    @Multipart
    @POST("/uploadFile")
    Call<RestApiDefaultResponse> uploadPhoto(@Part("file\"; filename=\"photo.jpg\" ")
RequestBody photo);
}
```

Где `RestApiDefaultResponse` - это настраиваемый класс, содержащий ответ.

Построение реализации вашего API и вызов этого вызова:

```
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>);
```

Дооснащение перехватчиком OkHttp

В этом примере показано, как использовать перехватчик запросов с помощью OkHttp. Это имеет множество вариантов использования, таких как:

- Добавление универсального `header` в запрос. Например, аутентификация запроса
- Отладка сетевых приложений
- Получение исходного `response`
- Ведение журнала транзакций и т. Д.
- Установка пользовательского агента пользователя

```
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);
```

Дополнительную информацию см. В разделе OkHttp .

Заголовок и тело: пример аутентификации

`@Header` и `@Body` можно поместить в сигнатуры методов, а Retrofit автоматически создаст их на основе ваших моделей.

```
public interface MyService {
     @POST("authentication/user")
```

```
     Call<AuthenticationResponse> authenticateUser(@Body AuthenticationRequest request,
@Header("Authorization") String basicToken);
}
```

AuthenticaionRequest - это наша модель, POJO, содержащая информацию, требуемую сервером. Для этого примера наш сервер хочет ключ клиента и секрет.

```
public class AuthenticationRequest {
     String clientKey;
     String clientSecret;
}
```

Обратите внимание, что в `@Header("Authorization")` мы указываем, что мы `@Header("Authorization")` заголовок авторизации. Другие заголовки будут заполнены автоматически, так как Retrofit может сделать вывод о том, что они основаны на типе объектов, которые мы отправляем и ожидаем взамен.

Мы где-то создаем нашу службу для переоснащения. Мы обязательно используем HTTPS.

```
Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https:// some example site")
            .client(client)
            .build();
MyService myService = retrofit.create(MyService.class)
```

Затем мы можем воспользоваться нашим сервисом.

```
AuthenticationRequest request = new AuthenticationRequest();
request.setClientKey(getClientKey());
request.setClientSecret(getClientSecret());
String basicToken = "Basic " + token;
myService.authenticateUser(request, basicToken);
```

Загрузите несколько файлов с помощью Retrofit как multipart

После того, как вы настроите среду Retrofit в своем проекте, вы можете использовать следующий пример, демонстрирующий, как загрузить несколько файлов с помощью 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;
}
```

Ниже приведен интерфейс

```
public interface WebAPIService {
    @Multipart
    @POST("main.php")
    Call<String> postFile(@PartMap Map<String,RequestBody> Files, @Part("json") String
description);
}
```

Загрузите файл с сервера с помощью Retrofit2

Объявление интерфейса для загрузки файла

```
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);
}
```

Опция 1 используется для загрузки файла с сервера с фиксированным URL. а опция 2 используется для передачи динамического значения в виде полного URL-адреса для запроса вызова. Это может быть полезно при загрузке файлов, которые зависят от параметров, таких как пользователь или время.

Уточнение настроек для совершения вызовов 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);
    }

}
```

Теперь сделаем реализацию api для загрузки файла с сервера

```
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) {
```

```
            }
        });
    }
```

И после получения ответа в обратном вызове введите код стандартного ввода-вывода для сохранения файла на диск. Вот код:

```
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;
        }
    }
```

Обратите внимание, что мы указали **ResponseBody** как возвращаемый тип, иначе Retrofit

попытается разобрать и преобразовать его, что не имеет смысла, когда вы загружаете файл.

Если вы хотите больше на переделанные вещи, подойдите к этой ссылке, поскольку это очень полезно. [1]: https://futurestud.io/blog/retrofit-getting-started-and-android-client

Отладка со Stetho

Добавьте в приложение следующие зависимости.

```
compile 'com.facebook.stetho:stetho:1.5.0'
compile 'com.facebook.stetho:stetho-okhttp3:1.5.0'
```

В методе `onCreate` вашего класса `onCreate` вызовите следующее.

```
Stetho.initializeWithDefaults(this);
```

При создании экземпляра `Retrofit` создайте собственный экземпляр OkHttp.

```
OkHttpClient.Builder clientBuilder = new OkHttpClient.Builder();
clientBuilder.addNetworkInterceptor(new StethoInterceptor());
```

Затем установите этот экземпляр OkHttp в экземпляре Retrofit.

```
Retrofit retrofit = new Retrofit.Builder()
    // ...
    .client(clientBuilder.build())
    .build();
```

Теперь подключите свой телефон к компьютеру, запустите приложение и введите `chrome://inspect` свой браузер Chrome. Теперь вам необходимо обновить сетевые вызовы для проверки.

Retrofit 2 Custom Xml Converter

Добавление зависимостей в файл build.gradle.

```
dependencies {
    ....
    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile ('com.thoughtworks.xstream:xstream:1.4.7') {
        exclude group: 'xmlpull', module: 'xmlpull'
    }
    ....
}
```

Затем создайте Factory Converter

```
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);
    }
}
```

создайте класс для обработки запроса тела.

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

создайте класс для обработки ответа тела.

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

Итак, этот момент мы можем отправлять и получать любой XML, нам просто нужно создать аннотации XStream для сущностей.

Затем создайте экземпляр 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();
```

Простой запрос POST с GSON

Пример JSON:

```
{
    "id": "12345",
    "type": "android"
}
```

Определите свой запрос:

```
public class GetDeviceRequest {

    @SerializedName("deviceId")
    private String mDeviceId;

    public GetDeviceRequest(String deviceId) {
        this.mDeviceId = deviceId;
    }

    public String getDeviceId() {
        return mDeviceId;
    }

}
```

Определите свой сервис (конечные точки для ударов):

```
public interface Service {

    @POST("device")
    Call<Device> getDevice(@Body GetDeviceRequest getDeviceRequest);

}
```

Определите экземпляр singleton сетевого клиента:

```
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;
    }

}
```

Определите простой объект модели для устройства:

```
public class Device {

    @SerializedName("id")
    private String mId;

    @SerializedName("type")
    private String mType;

    public String getId() {
        return mId;
    }

    public String getType() {
        return mType;
    }

}
```

Определить контроллер для обработки запросов на устройство

```
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
            }
```

```
        });
```

Чтение URL-адреса XML-формы с помощью дооснащения 2

Мы будем использовать retrofit 2 и SimpleXmlConverter для получения данных xml с url и анализа на Java-класс.

Добавить зависимость от сценария Gradle:

```
compile 'com.squareup.retrofit2:retrofit:2.1.0'
compile 'com.squareup.retrofit2:converter-simplexml:2.1.0'
```

Создать интерфейс

Также создайте оболочку класса xml в нашем случае Rss class

```
    public interface ApiDataInterface{

        // path to xml link on web site

        @GET (data/read.xml)

        Call<Rss> getData();

}
```

Функция чтения 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());
        }


    }
```

Это пример Java-класса с аннотациями SimpleXML

Подробнее об аннотации 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;
```

Прочитайте Retrofit2 онлайн: https://riptutorial.com/ru/android/topic/1132/retrofit2

глава 73: Retrofit2 с RxJava

Examples

Retrofit2 с RxJava

Во-первых, добавьте соответствующие зависимости в файл 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'
    ....
}
```

Затем создайте модель, которую хотите получить:

```
public class Server {
    public String name;
    public String url;
    public String apikey;
    public List<Site> siteList;
}
```

Создайте интерфейс, содержащий методы, используемые для обмена данными с удаленным сервером:

```
public interface ApiServerRequests {

    @GET("api/get-servers")
    public Observable<List<Server>> getServers();
}
```

Затем создайте экземпляр `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;
}
```

Затем, в любом месте от кода, вызовите метод:

```
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
       }
   });
```

Модернизация с помощью RxJava для получения данных асинхронно

Из репозитория GitHub RxJava *RxJava представляет собой реализацию виртуальных расширений Java VM: библиотека для составления асинхронных и основанных на событиях программ с использованием наблюдаемых последовательностей. Он расширяет шаблон наблюдателя для поддержки последовательностей данных / событий и добавляет операторы, которые позволяют скомпоновать последовательности вместе декларативно, в то же время абстрагируя проблемы, связанные с такими проблемами, как низкоуровневая потоковая передача, синхронизация, потоковая безопасность и параллельные структуры данных.*

Retrofit - это безопасный для HTTP-клиент HTTP для Android и Java, при этом разработчики могут сделать все сетевые вещи намного проще. В качестве примера мы собираемся загрузить JSON и показать его в RecyclerView в виде списка.

Начиная:

Добавьте зависимости RxJava, RxAndroid и Retrofit на уровне вашего приложения.
Build.gradle file: `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"
```

Определить ApiClient и ApiInterface для обмена данными с сервера

```
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;
}}
```

открытый интерфейс ApiInterface {

```
@GET("repos/{org}/{repo}/issues")
Observable<List<Issue>> getIssues(@Path("org") String organisation,
                                  @Path("repo") String repositoryName,
                                  @Query("page") int pageNumber);}
```

Обратите внимание, что getRepos () возвращает Observable, а не только список проблем.

Определить модели

Пример для этого показан. Вы можете использовать бесплатные услуги, такие как JsonSchema2Pojo или это.

```
public class Comment {

@SerializedName("url")
@Expose
private String url;
@SerializedName("html_url")
@Expose
private String htmlUrl;

//Getters and Setters
}
```

Создать экземпляр «Дополнения»

```
ApiInterface apiService = ApiClient.getInstance().create(ApiInterface.class);
```

Затем используйте этот экземпляр для извлечения данных с сервера

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

Теперь вы успешно извлекли данные с сервера с помощью Retrofit и RxJava.

Пример вложенных запросов: несколько запросов, объединение результатов

Предположим, у нас есть API, который позволяет нам получать метаданные объекта в одном запросе (`getAllPets`) и другой запрос, который имеет полные данные одного ресурса (`getSinglePet`). Как мы можем запросить их все в одной цепочке?

```
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;
    }
}
```

}

Прочитайте Retrofit2 с RxJava онлайн: https://riptutorial.com/ru/android/topic/7632/retrofit2-c-rxjava

глава 74: RoboGuice

Examples

Простой пример

RoboGuice - это основа, которая обеспечивает простоту и удобство Injection Dependency для Android, используя собственную библиотеку Guice от 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 );
    }
}
```

Установка для проектных решений

Добавьте следующий pom в раздел зависимостей вашего файла построения градиента:

```
project.dependencies {
    compile 'org.roboguice:roboguice:3.+'
    provided 'org.roboguice:roboblender:3.+'
}
```

Аннотация @ContentView

Аннотацию @ContentView можно использовать для дальнейшего облегчения разработки действий и замены оператора setContentView:

```
@ContentView(R.layout.myactivity_layout)
public class MyActivity extends RoboActivity {
    @InjectView(R.id.text1) TextView textView;

    @Override
    protected void onCreate( Bundle savedState ) {
        textView.setText("Hello!");
    }
}
```

@InjectResource аннотация

Вы можете вводить любые типы ресурсов, строки, анимации, чертежи и т. Д.

Чтобы ввести свой первый ресурс в действие, вам необходимо:

- Наследовать от RoboActivity
- Аннотируйте свои ресурсы с помощью @InjectResource

пример

```
@InjectResource(R.string.app_name) String name;

@InjectResource(R.drawable.ic_launcher) Drawable icLauncher;

@InjectResource(R.anim.my_animation) Animation myAnimation;
```

Аннотация @InjectView

Вы можете вводить любое представление с помощью аннотации @InjectView:

Вам необходимо:

- Наследовать от RoboActivity
- Настройка просмотра содержимого
- Аннотируйте свои представления с помощью @InjectView

пример

```
@InjectView(R.id.textView1) TextView textView1;

@InjectView(R.id.textView2) TextView textView2;

@InjectView(R.id.imageView1) ImageView imageView1;
```

Введение в RoboGuice

`RoboGuice` - это основа, которая обеспечивает простоту и удобство Injection Dependency для Android, используя собственную библиотеку Guice от Google.

RoboGuice 3 сокращает ваш код приложения. Меньше кода означает меньше возможностей для ошибок. Это также упрощает работу с вашим кодом - теперь ваш код не мешает механике платформы Android, но теперь он может сосредоточиться на фактической логике, уникальной для вашего приложения.

Чтобы дать вам представление, взгляните на этот простой пример типичной `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 );
        }
    }
```

Этот пример - 19 строк кода. Если вы пытаетесь прочитать `onCreate()` , вам нужно пропустить более 5 строк инициализации шаблона, чтобы найти единственное, что действительно имеет значение: `name.setText()` . И сложные действия могут в конечном итоге получить гораздо больше такого типа кода инициализации.

Сравните это с тем же приложением, написанным с помощью `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 );
        }
    }
```

Цель RoboGuice заключается в том, чтобы сделать ваш код более удобным для вашего приложения, а не для всего кода инициализации и жизненного цикла, который, как правило, требуется для поддержки в Android.

Аннотации:

Аннотации @ContentView:

Аннотацию @ContentView можно использовать для дальнейшего облегчения разработки действий и замены оператора setContentView:

```
@ContentView(R.layout.myactivity_layout)
    public class MyActivity extends RoboActivity {
        @InjectView(R.id.text1) TextView textView;

        @Override
        protected void onCreate( Bundle savedState ) {
```

```
            textView.setText("Hello!");
        }
    }
```

@InjectResource аннотация:

Сначала вам нужна операция, которая наследуется от RoboActivity. Затем, предположив, что у вас есть анимация my_animation.xml в папке res / anim, вы можете теперь ссылаться на нее с аннотацией:

```
public class MyActivity extends RoboActivity {
    @InjectResource(R.anim.my_animation) Animation myAnimation;
    // the rest of your code
}
```

@ Вводная аннотация:

Вы убедитесь, что ваша деятельность простирается от RoboActivity и аннотирует члена службы System с помощью @Inject. Roboguice сделает все остальное.

```
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();
```

В дополнение к представлениям, ресурсам, службам и другим специфичным для Android задачам RoboGuice может вводить обычные объекты Java. По умолчанию Roboguice вызовет конструктор аргументов no вашего POJO

```
class MyActivity extends RoboActivity {
    @Inject Foo foo; // this will basically call new Foo();
}
```

Прочитайте RoboGuice онлайн: https://riptutorial.com/ru/android/topic/2563/roboguice

глава 75: Robolectric

Вступление

Единичное тестирование выполняет часть кода и тестирует его независимо от каких-либо других зависимостей или частей системы (например, базы данных).

Robolectric - это единая тестовая платформа, которая де-фангирует Android SDK, чтобы вы могли тестировать разработку своего приложения для Android. Тесты запускаются внутри JVM на рабочей станции за считанные секунды.

С их помощью вы можете быстро запускать тесты на JVN, все еще используя API Android.

Examples

Robolectric test

```
@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!");
  }
}
```

конфигурация

Чтобы настроить robolectric add `@Config` для проверки класса или метода.

Запуск с пользовательским классом приложений

```
@RunWith(RobolectricTestRunner.class)
@Config(application = MyApplication.class)
public final class MyTest {
}
```

Установить целевой SDK

```
@RunWith(RobolectricTestRunner.class)
@Config(sdk = Build.VERSION_CODES.LOLLIPOP)
public final class MyTest {
}
```

Выполнить с помощью обычного манифеста

Когда указано, robolectric будет выглядеть относительно текущего каталога. Значение по умолчанию - `AndroidManifest.xml`

Ресурсы и активы будут загружены относительно манифеста.

```
@RunWith(RobolectricTestRunner.class)
@Config(manifest = "path/AndroidManifest.xml")
public final class MyTest {
}
```

Использовать квалификаторы

Возможные квалификаторы можно найти в документах android .

```
@RunWith(RobolectricTestRunner.class)
public final class MyTest {

    @Config(qualifiers = "sw600dp")
    public void testForTablet() {
    }
}
```

Прочитайте Robolectric онлайн: https://riptutorial.com/ru/android/topic/8743/robolectric

глава 76: SearchView

Examples

AppView для поиска приложений с помощью RxBindings

build.gradle :

```
dependencies {
    compile 'com.android.support:appcompat-v7:23.3.0'
    compile 'com.jakewharton.rxbinding:rxbinding-appcompat-v7:0.4.0'
}
```

menu / menu.xml :

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/action_search" android:title="Search"
        android:icon="@android:drawable/ic_menu_search"
        app:actionViewClass="android.support.v7.widget.SearchView"
        app:showAsAction="always"/>
</menu>
```

MainActivity.java :

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);

    MenuItem searchMenuItem = menu.findItem(R.id.action_search);
    setupSearchView(searchMenuItem );

    return true;
}

private void setupSearchView(MenuItem searchMenuItem) {
    SearchView searchView = (SearchView) searchMenuItem.getActionView();
    searchView.setQueryHint(getString(R.string.search_hint)); // your hint here

    SearchAdapter searchAdapter = new SearchAdapter(this);
    searchView.setSuggestionsAdapter(searchAdapter);

    // optional: set the letters count after which the search will begin to 1
    // the default is 2
    try {
        int autoCompleteTextViewID =
getResources().getIdentifier("android:id/search_src_text", null, null);
        AutoCompleteTextView searchAutoCompleteTextView = (AutoCompleteTextView)
searchView.findViewById(autoCompleteTextViewID);
        searchAutoCompleteTextView.setThreshold(1);
    } catch (Exception e) {
```

```
        Logs.e(TAG, "failed to set search view letters threshold");
    }

    searchView.setOnSearchClickListener(v -> {
        // optional actions to search view expand
    });
    searchView.setOnCloseListener(() -> {
        // optional actions to search view close
        return false;
    });

    RxSearchView.queryTextChanges(searchView)
            .doOnEach(notification -> {
                CharSequence query = (CharSequence) notification.getValue();
                searchAdapter.filter(query);
            })
            .debounce(300, TimeUnit.MILLISECONDS) // to skip intermediate letters
            .flatMap(query -> MyWebService.search(query)) // make a search request
            .retry(3)
            .subscribe(results -> {
                searchAdapter.populateAdapter(results);
            });

    //optional: collapse the searchView on close
    searchView.setOnQueryTextFocusChangeListener((view, queryTextFocused) -> {
        if (!queryTextFocused) {
            collapseSearchView();
        }
    });
}
```

SearchAdapter.java

```
public class SearchAdapter extends CursorAdapter {
    private List<SearchResult> items = Collections.emptyList();

    public SearchAdapter(Activity activity) {
        super(activity, null, CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
    }

    public void populateAdapter(List<SearchResult> items) {
        this.items = items;
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            c.addRow(new Object[]{i});
        }
        changeCursor(c);
        notifyDataSetChanged();
    }

    public void filter(CharSequence query) {
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            SearchResult result = items.get(i);
            if (result.getText().startsWith(query.toString())) {
                c.addRow(new Object[]{i});
            }
        }
        changeCursor(c);
        notifyDataSetChanged();
```

```
    }

    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        ViewHolder holder = (ViewHolder) view.getTag();
        int position = cursor.getPosition();
        if (position < items.size()) {
            SearchResult result = items.get(position);
            // bind your view here
        }
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        LayoutInflater inflater = (LayoutInflater) context
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        View v = inflater.inflate(R.layout.search_list_item, parent, false);
        ViewHolder holder = new ViewHolder(v);

        v.setTag(holder);
        return v;
    }

    private static class ViewHolder {
        public final TextView text;

        public ViewHolder(View v) {
            this.text= (TextView) v.findViewById(R.id.text);
        }
    }
}
```

Поиск в панели инструментов с фрагментом

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);
    }
}
```

Справочный снимок экрана:

Hello Android

`menu.xml` , нам нужно понять, что она полностью зависит от стиля, применяемого к базовой панели инструментов. Для создания темы на панели инструментов выполните следующие действия.

Создайте стиль в `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>
```

Примените стиль к панели инструментов.

```
<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" />
```

Это дает желаемый цвет для всех видов, соответствующих панели инструментов (кнопка возврата, значки меню и SearchView).

Прочитайте SearchView онлайн: https://riptutorial.com/ru/android/topic/4786/searchview

глава 77: SensorManager

Examples

Получение событий датчиков

Извлечение информации о датчике с бортовых датчиков:

```
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
    }
}
```

Преобразование датчиков в мировую систему координат

Значения датчиков, возвращаемые Android, соответствуют системе координат телефона (например, + Y указывает на верхнюю часть телефона). Мы можем преобразовать эти значения датчиков в мировую систему координат (например, + Y указывает на магнитный север, касательный к земле) с использованием матрицы вращения управляющих датчиков

Во-первых, вам нужно будет объявить и инициализировать матрицы / массивы, в которых будут храниться данные (например, вы можете сделать это в методе `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];
```

Затем нам нужно обнаружить изменения в значениях датчиков, сохранить их в соответствующие массивы (если мы хотим использовать их позже / в другом месте), затем вычислить матрицу вращения и преобразование в мировые координаты:

```
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];

}
```

Решите, если ваше устройство статично или нет, используя акселерометр

Добавьте следующий код в метод `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);
```

Возможно, вам придется настроить чувствительность, адаптировав `movementThreshold` к пробной и ошибке. Затем переопределите метод `onSensorChanged()` следующим образом:

```
@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);
    }
}
```

Если вы хотите, чтобы ваше приложение не было установлено на устройствах, у которых нет акселерометра, вы должны добавить следующую строку в свой манифест:

```
<uses-feature android:name="android.hardware.sensor.accelerometer" />
```

Прочитайте SensorManager онлайн: https://riptutorial.com/ru/android/topic/3344/sensormanager

глава 78: SharedPreferences

Вступление

SharedPreferences обеспечивают способ сохранения данных на диск в виде пар **ключ-значение** .

Синтаксис

- **Метод контекста**
  - public SharedPreferences getSharedPreferences (String name, int mode)
- **Способ действия**
  - public SharedPreferences getPreferences ()
- **Методы совместного доступа**
  - public SharedPreferences.Editor edit ()
  - public boolean содержит ()
  - public Map <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 (клавиша String, long defValue)
  - public String getString (String key, String defValue)
  - public Set getStringSet (клавиша String, Set defValues)
  - public void registerOnSharedPreferenceChangeListener (SharedPreferences.OnSharedPreferenceChangeListener)
  - public void unregisterOnSharedPreferenceChangeListener (SharedPreferences.OnSharedPreferenceChangeListener listener)
- **Методы SharedPreferences.Editor**
  - public void apply ()
  - public boolean commit ()
  - public SharedPreferences.Editor clear ()
  - public SharedPreferences.Editor putBoolean (Строковый ключ, логическое значение)
  - public SharedPreferences.Editor putFloat (String key, float value)
  - public SharedPreferences.Editor putInt (String key, int value)
  - public SharedPreferences.Editor putLong (Строковый ключ, длинное значение)
  - public SharedPreferences.Editor putString (String key, String value)

- public SharedPreferences.Editor putStringSet (String key, Set values)
- public SharedPreferences.Editor remove (String key)

параметры

параметр	подробности
ключ	Непустая `String` идентифицирующая параметр. Он может содержать пробелы или непечатные. Это используется только в вашем приложении (и в XML-файле), поэтому ему не требуется пространство имен, но рекомендуется иметь его как константу в исходном коде. Не локализуйте его.
Значение_по_умолчанию	Все функции get принимают значение по умолчанию, которое возвращается, если данный ключ отсутствует в `SharedPreferences` . Он не возвращается, если ключ присутствует, но значение имеет неправильный тип: в этом случае вы получаете `ClassCastException` .

замечания

- `SharedPreferences` не следует использовать для хранения большого количества данных. Для таких целей гораздо лучше использовать `SQLiteDatabase` .
- `SharedPreferences` - это только один процесс, если вы не используете устаревший режим `MODE_MULTI_PROCESS` . Поэтому, если ваше приложение имеет несколько процессов, вы не сможете читать `SharedPreferences` основного процесса в другом процессе. В таких случаях вы должны использовать другой механизм для обмена данными между процессами, но не используйте `MODE_MULTI_PROCESS` поскольку он не является надежным и не рекомендуется.
- Лучше использовать экземпляр `SharedPreferences` в классе `Singleton` для доступа по всему `context` приложения. Если вы хотите использовать его только для определенного действия, перейдите для `getPreferences()` .
- Избегайте хранения конфиденциальной информации в открытом виде при использовании `SharedPreferences` так как ее можно легко прочитать.

Официальная документация

https://developer.android.com/reference/android/content/SharedPreferences.html

Examples

Чтение и запись значений в 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()` представляет собой метод из `Context` класса - который `Activity` продолжается. Если вам нужен доступ к `getSharedPreferences()` из других классов, вы можете использовать `context.getSharedPreferences()` с ссылкой на объект `Context` из `Activity` , `View` или `Application` .

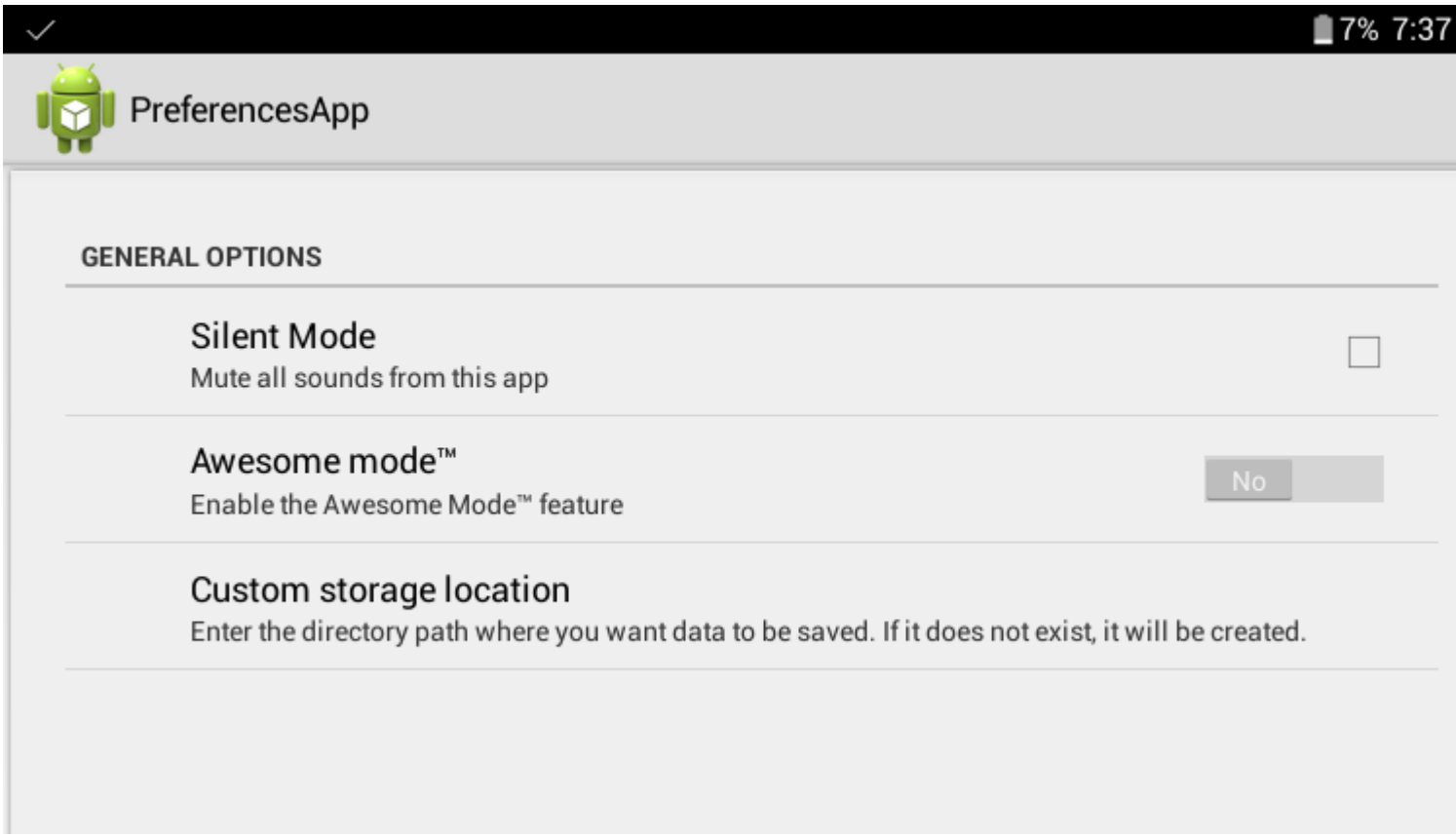
Удаление ключей

```
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();
```

После `apply()` `prefs` содержит «ключ» -> «значение», в дополнение к тому, что он уже содержит. Хотя похоже, что я добавил «ключ», а затем удалил его, сначала происходит удаление. Изменения в `Editor` все применяются за один раз, а не в том порядке, в котором вы их добавили. Все удары происходят до всех пометок.

Реализация экрана настроек с использованием SharedPreferences

Одно использование `SharedPreferences` заключается в реализации экрана «Настройки» в вашем приложении, где пользователь может установить свои настройки / параметры. Как это:



A PreferenceScreen сохраняет пользовательские настройки в `SharedPreferences` . Чтобы

создать PreferenceScreen, вам нужно несколько вещей:

XML-файл для определения доступных параметров:

Это происходит в `/res/xml/preferences.xml` , и для экрана с приведенными выше настройками он выглядит следующим образом:

```
<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>
```

Это определяет доступные параметры на экране настроек. Есть много других типов `Preference` перечисленных в документации разработчиков Android в классе предпочтений .

Затем нам понадобится **Activity для размещения** пользовательского интерфейса **Preferences** . В этом случае он довольно короткий и выглядит следующим образом:

```
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);
    }
}
```

Он расширяет `PreferenceActivity` и предоставляет пользовательский интерфейс для экрана

настроек. Его можно запустить как обычную активность, в данном случае, с чем-то вроде:

```
Intent i = new Intent(this, PreferencesActivity.class);
startActivity(i);
```

Не забудьте добавить `PreferencesActivity` в ваш `AndroidManifest.xml` .

Получение значений параметров внутри вашего приложения довольно просто, просто `setDefaultValues()` вызовите `setDefaultValues()` , чтобы установить значения по умолчанию, определенные в вашем XML, а затем получить стандартные `SharedPreferences` . Пример:

```
//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", "");
```

Извлеките все сохраненные записи из определенного файла SharedPreferences

Метод `getAll()` извлекает все значения из настроек. Мы можем использовать его, например, для регистрации текущего содержимого `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);
    }
}
```

Документация предупреждает вас об изменении `Collection` возвращенной `getAll` :

> Обратите внимание, что вы не должны изменять коллекцию, возвращаемую этим методом, или изменять любое ее содержимое. Согласованность сохраненных вами данных не гарантируется, если вы это сделаете.

Прослушивание изменений SharedPreferences

```
SharedPreferences sharedPreferences = ...;
sharedPreferences.registerOnSharedPreferenceChangeListener(mOnSharedPreferenceChangeListener);
```

```
private final SharedPreferences.OnSharedPreferenceChangeListener
mOnSharedPreferenceChangeListener = new SharedPreferences.OnSharedPreferenceChangeListener() {
    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
        //TODO
    }
}
```

Пожалуйста, обратите внимание:

- Слушатель будет стрелять, только если значение было добавлено или изменено, установка того же значения не вызовет его;
- Слушатель должен быть сохранен в переменной-члене, а **НЕ** с анонимным классом, потому что `registerOnSharedPreferenceChangeListener` сохраняет его со слабой ссылкой, поэтому это будет сбор мусора;
- Вместо использования переменной-члена она также может быть непосредственно реализована классом, а затем вызывает `registerOnSharedPreferenceChangeListener(this);`
- Не забудьте отменить регистрацию слушателя, когда это не требуется, используя `unregisterOnSharedPreferenceChangeListener` .

Чтение и запись данных в SharedPreferences с Singleton

SharedPreferences Manager (Singleton) для чтения и записи всех типов данных.

```
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` который реализуется классами, идущими в `Gson` чтобы избежать обфускации proguard.

```
public interface Model {

}
```

Правила Proguard для интерфейса `Model` :

```
-keep interface com.example.app.Model
-keep class * implements com.example.app.Model { *;}
```

Различные способы создания объекта SharedPreferences

Вы можете получить доступ к SharedPreferences несколькими способами:

Получить файл SharedPreferences по умолчанию:

```
import android.preference.PreferenceManager;
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
```

Получить определенный файл SharedPreferences:

```
public static final String PREF_FILE_NAME = "PrefFile";
SharedPreferences prefs = getSharedPreferences(PREF_FILE_NAME, MODE_PRIVATE);
```

Получить SharedPreferences из другого приложения:

```
// 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)`

возвращает предпочтения, сохраненные `Activity's class name` как описано в документах :

> Извлеките объект SharedPreferences для доступа к предпочтениям, которые являются приватными для этого действия. Это просто вызывает базовый метод getSharedPreferences (String, int), передавая имя класса этой активности в качестве имени предпочтений.

При использовании метода getSharedPreferences (String name, int mode) возвращает префы, сохраненные под заданным `name` . Как и в документах:

> Извлеките и сохраните содержимое файла настроек «имя», возвращая SharedPreferences, через который вы можете получить и изменить свои значения.

Поэтому, если значение, сохраненное в `SharedPreferences` , должно использоваться в приложении, следует использовать `getSharedPreferences (String name, int mode)` с фиксированным именем. Поскольку, используя `getPreferences(int)` возвращает / сохраняет предпочтения, принадлежащие вызывающей его `Activity` .

Commit vs. Apply

Метод `editor.apply()` является **асинхронным** , а `editor.commit()` является **синхронным** .

Очевидно, вы должны вызвать либо `apply()` либо `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()` был добавлен в 2.3 (API 9), он совершает ошибку, не возвращая логическое значение, указывающее на успех или неудачу.

`commit()` возвращает true, если сохранение работает, в противном случае false.

`apply()` была добавлена, поскольку команда разработчиков Android заметила, что почти никто не обратил внимание на возвращаемое значение, поэтому применение выполняется быстрее, так как оно асинхронно.

В отличие от `commit()` , который синхронно записывает свои предпочтения в постоянное хранилище, `apply()` немедленно выполняет изменения в `SharedPreferences` в памяти, но начинает асинхронную фиксацию на диск, и вы не будете уведомлены о каких-либо сбоях. Если другой редактор этой `SharedPreferences` выполняет регулярную `commit()` а `apply()` по-прежнему `SharedPreferences` , `commit()` будет блокироваться до тех пор, пока все асинхронные транзакции (применение) не будут завершены, а также любые другие транзакции синхронизации, которые могут быть отложены.

Поддерживаемые типы данных в SharedPreferences

`SharedPreferences` позволяет вам хранить только примитивные типы данных (`boolean` , `float` , `long` , `int` , `String` и `string set`). Вы не можете хранить более сложные объекты в `SharedPreferences` , и как таковое действительно предназначено для хранения пользовательских настроек или аналогичных, это не означает, что база данных хранит пользовательские данные (например, сохранение списка задач, например, пользователя).

Чтобы сохранить что-то в `SharedPreferences` вы используете ключ и значение. Ключ - это то, как вы можете ссылаться на то, что вы сохранили позже, и данные Value, которые вы хотите сохранить.

```
    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();
```

Хранить, извлекать, удалять и очищать данные из общих ресурсов

Создать SharedPreferences BuyyaPref

```
SharedPreferences pref = getApplicationContext().getSharedPreferences("BuyyaPref",
MODE_PRIVATE);
Editor editor = pref.edit();
```

Хранение данных в виде пары 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
```

Получить данные SharedPreferences

Если значение для ключа не существует, верните второе значение параметра (в этом случае значение null, это как значение по умолчанию)

```
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
```

Удаление значения ключа из 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
```

Очистить все данные из SharedPreferences

```
 editor.clear();
 editor.commit(); // commit changes
```

Поддержка pre-Honeycomb с помощью StringSet

Вот класс утилиты:

```
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() {}
}
```

Пример сохранения предпочтений в виде типа данных StringSet:

```
Set<String> sets = new HashSet<>();
sets.add("John");
sets.add("Nicko");
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
SharedPreferencesCompat.putStringSet(preferences.edit(), "pref_people", sets);
```

Чтобы вернуть их обратно:

```
Set<String> people = SharedPreferencesCompat.getStringSet(preferences, "pref_people", new
HashSet<String>());
```

Ссылка: поддержка Android

Добавить фильтр для EditTextPreference

Создайте этот класс:

```
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;
    }
}
```

Использование:

```
EditText compressPic = ((EditTextPreference)
findPreference(getString("pref_key_compress_pic"))).getEditText();
compressPic.setFilters(new InputFilter[]{ new InputFilterMinMax(1, 100) });
```

Прочитайте SharedPreferences онлайн:
https://riptutorial.com/ru/android/topic/119/sharedpreferences

глава 79: ShortcutManager

Examples

Динамические клавиши быстрого запуска

```
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));
```

Мы можем легко удалить все динамические ярлыки, позвонив: -

```
 shortcutManager.removeAllDynamicShortcuts();
```

Мы можем обновить существующие динамические shorcuts, используя

```
shortcutManager.updateShortcuts(Arrays.asList(shortcut);
```

Обратите внимание, что `setDynamicShortcuts(List)` используется для переопределения всего списка динамических ярлыков, `addDynamicShortcuts(List)` используется для добавления динамических ярлыков в существующий список динамических ярлыков

Прочитайте ShortcutManager онлайн:
https://riptutorial.com/ru/android/topic/7661/shortcutmanager

глава 80: SpannableString

Синтаксис

- `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

Добавить стили в TextView

В следующем примере мы создаем Activity для отображения одного TextView.

TextView будет использовать `SpannableString` качестве своего содержимого, что иллюстрирует некоторые из доступных стилей.

Вот, что мы будем делать с текстом:

- Сделайте это больше
- Смелый
- подчеркивание
- выделять подчеркиванием
- Зачеркнутые
- цветной
- Выделенные
- Показать как верхний индекс
- Показать как индекс
- Показать как ссылку
- Сделайте его интерактивным.

```
@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);

}
```

И результат будет выглядеть так:

Многострочный, с несколькими цветами

Метод: **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;
}
```

Использование:

```
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`
```

Прочитайте SpannableString онлайн:
https://riptutorial.com/ru/android/topic/10553/spannablestring

глава 81: SQLite

Вступление

SQLite - это система управления реляционными базами данных, написанная на языке C. Чтобы начать работу с базами данных SQLite в рамках Android, определите класс, который расширяет SQLiteOpenHelper , и настройте по мере необходимости.

замечания

Класс `SQLiteOpenHelper` определяет статические `onCreate()` и `onUpgrade()` . Эти методы вызывают в соответствующих методах подкласса `SQLiteOpenHelper` которые вы настраиваете с помощью собственных таблиц.

Examples

Использование класса 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;
            }
        }
    }
}
```

Вставка данных в базу данных

```
// 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
);
```

метод onUpgrade ()

`SQLiteOpenHelper` - это вспомогательный класс для управления созданием базы данных и управлением версиями.

В этом классе метод `onUpgrade()` отвечает за обновление базы данных при внесении изменений в схему. Он вызывается, когда файл базы данных уже существует, но его версия ниже, чем версия, указанная в текущей версии приложения. Для каждой версии базы данных необходимо применить определенные изменения.

```
@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;
        }
    }
}
```

Чтение данных из курсора

Ниже приведен пример метода, который будет жить внутри подкласса `SQLiteOpenHelper` . Он использует строку `searchTerm` для фильтрации результатов, итерации через содержимое курсора и возвращает это содержимое в `List` объектов `Product` .

Сначала определите класс POJO `Product` который будет контейнером для каждой строки, полученной из базы данных:

```
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;
  }
}
```

Затем определите метод, который будет запрашивать базу данных, и верните `List` объектов `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();
    }
}
```

Создание контракта, помощника и поставщика для SQLite в 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;
    }

}
```

Как пользоваться:

```
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;
}
```

Обновление строки в таблице

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

Выполнение транзакции

Транзакции могут использоваться для многократного внесения изменений в базу данных атомарно. Любая нормальная транзакция следует этой схеме:

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

Вызов `beginTransaction()` внутри активных транзакций не влияет.

Удалить строки (строки) из таблицы

Чтобы удалить все строки из таблицы

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

db.delete(TABLE_NAME, null, null);
db.close();
```

Чтобы удалить все строки из таблицы и получить количество удаленной строки в возвращаемом значении

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

int numRowsDeleted = db.delete(TABLE_NAME, String.valueOf(1), null);
db.close();
```

Чтобы удалить строки (строки) с условием `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();
```

Сохранить изображение в SQLite

Настройка базы данных

```
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);
    }
}
```

Вставить в базу данных:

```
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 );
}
```

Получение данных :

```
 byte[] image = cursor.getBlob(1);
```

Замечания:

1. Перед вставкой в базу данных сначала необходимо преобразовать образ Bitmap в массив байтов, а затем применить его с помощью запроса базы данных.
2. При извлечении из базы данных вы, безусловно, имеете массив байтов изображения, вам нужно преобразовать массив байтов в исходное изображение. Таким образом, вы должны использовать BitmapFactory для декодирования.

Ниже приведен класс Utility, который, я надеюсь, поможет вам:

```
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);
    }
}
```

Создать базу данных из папки с данными

Поместите файл dbname.sqlite или dbname.db в папку с ресурсами вашего проекта.

```
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) {

        }
    }
```

Вот как вы можете получить доступ к объекту базы данных к своей деятельности.

```
// Create Databasehelper class object in your activity.
   private Databasehelper db;
```

Затем в onCreate Method его инициализируйте и вызовите метод createDatabase (), как показано ниже.

```
db = new Databasehelper(MainActivity.this);
        try {
            db.createDataBase();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

Выполните все операции вставки, обновления, удаления и выбора, как показано ниже.

```
String query = "select Max(Id) as Id from " + TABLE_NAME;
        db.openDataBase();
        int count = db.getId(query);
        db.close();
```

Экспорт и импорт базы данных

Например, вы можете импортировать и экспортировать свою базу данных для bacukup. Не забывайте о разрешениях.

```
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();
    }
}
```

Массовая вставка

Вот пример вставки больших фрагментов данных сразу. Все данные, которые вы хотите вставить, собираются внутри массива 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;
}
```

И вот пример того, как его использовать:

```
ContentResolver resolver = mContext.getContentResolver();
ContentValues[] valueList = new ContentValues[object.size()];
//add whatever you like to the valueList
resolver.bulkInsert(IChatContract.MessageColumns.CONTENT_URI, valueList);
```

Прочитайте SQLite онлайн: https://riptutorial.com/ru/android/topic/871/sqlite

глава 82: SyncAdapter с периодической синхронизацией данных

Вступление

Компонент адаптера синхронизации в вашем приложении инкапсулирует код для задач, которые передают данные между устройством и сервером. Основываясь на планировании и триггерах, которые вы предоставляете в своем приложении, инфраструктура адаптера синхронизации использует код в компоненте адаптера синхронизации.

Недавно я работал над SyncAdapter, я хочу поделиться своими знаниями с другими, это может помочь другим.

Examples

Синхронизирующий адаптер с каждым минимальным запрашивающим значением с сервера.

```
<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>
```

Этот код необходимо добавить в файл манифеста

В приведенном выше коде у нас есть служба syncservice и contetrovider и authenticatorservice.

В приложении нам нужно создать пакет xml для добавления файлов syncadpter и authenticator xml. **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"/>
```

Authenticator

```
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;
```

```
    }
}
```

Основная деятельность

```
     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();
}
```

}

Прочитайте SyncAdapter с периодической синхронизацией данных онлайн: https://riptutorial.com/ru/android/topic/10774/syncadapter-с-периодической-синхронизацией-данных

глава 83: TabLayout

Examples

Использование TabLayout без ViewPager

В большинстве случаев `TabLayout` используется вместе с ViewPager , чтобы получить функциональность салфетки, которая поставляется вместе с ним.

Можно использовать `TabLayout` без `ViewPager` , используя `TabLayout.OnTabSelectedListener` .

Во-первых, добавьте `TabLayout` в XML-файл вашей деятельности:

```
<android.support.design.widget.TabLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:id="@+id/tabLayout" />
```

Для навигации в рамках `Activity` вручную укажите пользовательский интерфейс на основе выбранной вкладки.

```
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) {

    }
});
```

Прочитайте TabLayout онлайн: https://riptutorial.com/ru/android/topic/7601/tablayout

глава 84: TensorFlow

Вступление

TensorFlow был разработан с учетом мобильных и встроенных платформ. У нас есть пример кода и поддержка сборки, которые вы можете попробовать сейчас для этих платформ:

Android iOS Raspberry Pi

замечания

Оценка работы MindRocks

Examples

Как пользоваться

Установите Bazel отсюда . Bazel - это основная система сборки для TensorFlow. Теперь отредактируйте WORKSPACE, мы найдем файл WORKSPACE в корневом каталоге TensorFlow, который мы клонировали ранее.

```
# 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)
```

Как ниже с нашими sdk и 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)
```

Прочитайте TensorFlow онлайн: https://riptutorial.com/ru/android/topic/9991/tensorflow

глава 85: TextInputLayout

Вступление

TextInputLayout был введен для отображения плавающей метки в EditText. EditText должен быть обернут TextInputLayout для отображения плавающей метки.

замечания

`TextInputLayout` - это макет, который обертывает `EditText` (или потомок), чтобы показать плавающий метку, когда подсказка скрыта из-за ввода пользователем текста. Кроме того, `TextInputLayout` позволяет отображать сообщение об ошибке ниже `EditText` .

Убедитесь, что в файл `build.gradle` вашего приложения добавлена `build.gradle` зависимость:

```
compile 'com.android.support:design:25.3.1'
```

Examples

Основное использование

Это основное использование `TextInputLayout` .
Обязательно добавьте зависимость в файле `build.gradle` как описано в разделе примечаний.

Пример:

```
<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>
```

Обработка ошибок

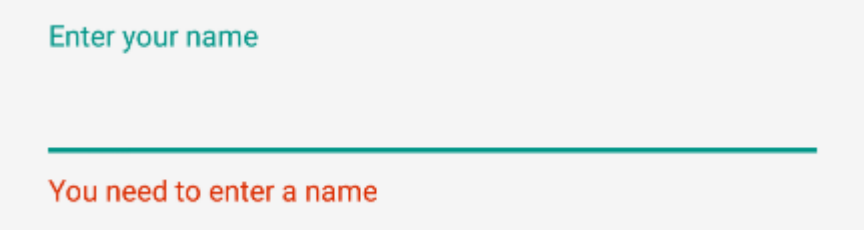
Вы можете использовать `TextInputLayout` для отображения сообщений об ошибках в соответствии с рекомендациями по разработке материалов с использованием методов `setError` и `setErrorEnabled` .

Чтобы показать ошибку ниже использования EditText:

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setErrorEnabled(true);
til.setError("You need to enter a name");
```

Чтобы включить ошибку в `TextInputLayout` вы можете использовать `app:errorEnabled="true"` в xml или `til.setErrorEnabled(true);` как показано выше.

Вы получите:



Добавление подсчета символов

TextInputLayout имеет счетчик символов для EditText, определенных внутри него. Счетчик будет отображаться ниже EditText.

Просто используйте `setCounterEnabled()` и `setCounterMaxLength` :

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setCounterEnabled(true);
til.setCounterMaxLength(15);
```

или `app:counterEnabled` и `app:counterMaxLength` в xml.

```
<android.support.design.widget.TextInputLayout
    app:counterEnabled="true"
    app:counterMaxLength="15">

    <EditText/>

</android.support.design.widget.TextInputLayout>
```

Переключатели просмотров

С типом ввода пароля вы также можете включить значок, который может отображать или скрывать весь текст, используя атрибут `passwordToggleEnabled` .

Вы также можете настроить тот же параметр по умолчанию, используя следующие атрибуты:

- `passwordToggleDrawable` : изменить значок глаз по умолчанию
- `passwordToggleTint` : применить оттенок к видимости пароля для переключения.

- `passwordToggleTintMode` : указать режим наложения, используемый для применения фонового оттенка.

Пример:

```
<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` - это `EditText` с дополнительным исправлением, чтобы отображать подсказку в IME в режиме «extract» .

Режим **Extract** - это режим, который переключает редактор клавиатуры, когда вы нажимаете EditText, когда пространство слишком мало (например, пейзаж на смартфоне). В этом случае, используя `EditText` во время редактирования текста, вы можете видеть, что IME не дает вам намека на то, что вы редактируете

`TextInputEditText` исправляет эту проблему, предоставляя текст подсказки, в то время как IME устройства пользователя находится в режиме Extract.

Пример:

```
<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>
```

Настройка внешнего вида TextInputLayout

Вы можете настроить внешний вид `TextInputLayout` и встроенного `EditText` , указав пользовательские стили в ваших `styles.xml` . Определенные стили могут быть добавлены как стили или темы в `TextInputLayout` .

Пример настройки отображения подсказки:

`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>
```

Чтобы применить стиль, обновите TextInputLayout и EditText следующим образом.

```
<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>
```

Пример настройки цвета акцента `TextInputLayout` . Цвет акцента влияет на цвет базовой линии `EditText` и цвет текста для плавающего текста подсказки:

`styles.xml` :

```
<style name="TextInputLayoutWithPrimaryColor" parent="Widget.Design.TextInputLayout">
    <item name="colorAccent">@color/primary</item>
</style>
```

файл макета:

```
<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>
```

Прочитайте TextInputLayout онлайн: https://riptutorial.com/ru/android/topic/5652/textinputlayout

глава 86: TextView

Вступление

Все, что связано с настройкой TextView в Android SDK

Синтаксис

- TextView (контекст контекста)
- (TextView) findViewById (int id)
- void setText (int resid)
- void setText (текст CharSequence) // Вы можете использовать String в качестве аргумента

замечания

Попробуйте использовать его в xml-дизайне или программно.

Examples

Текстовое изображение с различным текстом

Вы можете архивировать различные Textsizes внутри Textview со 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)
```

Настройка 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);
        }
```

```
    }
}
```

Использование:

```
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);
    }
}
```

Планировка:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@mipmap/background">


    <pk.sohail.gallerytest.activity.CustomTextView
        android:id="@+id/pager_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:gravity="center"
        android:text="@string/txt_title_photo_gallery"
        android:textColor="@color/white"
        android:textSize="30dp"
        android:textStyle="bold"
        app:outerShadowRadius="10dp"
        app:strokeColor="@color/title_text_color"
        app:strokeJoinStyle="miter"
        app:strokeWidth="2dp" />

</RelativeLayout>
```

Attars:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <declare-styleable name="CustomTextView">

        <attr name="outerShadowRadius" format="dimension" />
        <attr name="strokeWidth" format="dimension" />
        <attr name="strokeMiter" format="dimension" />
        <attr name="strokeColor" format="color" />
        <attr name="strokeJoinStyle">
            <enum name="miter" value="0" />
            <enum name="bevel" value="1" />
```

```
            <enum name="round" value="2" />
        </attr>
    </declare-styleable>

</resources>
```

Программное использование:

```
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");
```

Spannable TextView

Spannable `TextView` может использоваться в Android для выделения определенной части текста с другим цветом, стилем, размером и / или событием клика в одном виджете `TextView` .

Подумайте, что вы определили `TextView` следующим образом:

```
TextView textview=findViewById(R.id.textview);
```

Затем вы можете применить к нему различную подсветку, как показано ниже:

- **Spannable цвет:** Для того , чтобы установить другой цвет на какой - то части текста, А `ForegroundColorSpan` может быть использован, как показано в следующем примере:

  ```
  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 );
  ```

  Результат, созданный приведенным выше кодом:

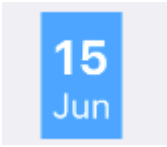
  

- **Spannable font:** для установки другого размера шрифта на некоторую часть текста можно использовать `RelativeSizeSpan` , как показано в следующем примере:

  ```
  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 );
```

Результат, созданный приведенным выше кодом:



- **Spannable typeface: для** того, чтобы установить шрифт другого шрифта на некоторую часть текста, может использоваться пользовательский `TypefaceSpan` , как показано в следующем примере:

```
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 );
```

Однако для того, чтобы сделать вышеуказанный код, класс `CustomTypefaceSpan` должен быть получен из класса `TypefaceSpan` . Это можно сделать следующим образом:

```
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 с изображением

Android позволяет программистам размещать изображения во всех четырех углах `TextView` . Например, если вы создаете поле с `TextView` и в то же время вы хотите показать, что это поле редактируется, тогда разработчики обычно помещают значок редактирования рядом с этим полем. Android предоставляет нам интересный вариант, называемый **составным** для `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" />
```

Вы можете установить переносимое на любую сторону `TextView` следующим образом:

```
android:drawableLeft="@drawable/edit"
android:drawableRight="@drawable/edit"
android:drawableTop="@drawable/edit"
android:drawableBottom="@drawable/edit"
```

Настройка выталкиваемого продукта также может быть достигнута программно следующим образом:

```
yourTextView.setCompoundDrawables(leftDrawable, rightDrawable, topDrawable, bottomDrawable);
```

Установка любого из параметров, переданных `setCompoundDrawables()` в `null` , удалит значок с соответствующей стороны `TextView` .

Зачеркнутый TextView

Зачеркните весь текст

```
String sampleText = "This is a test strike";
textView.setPaintFlags(tv.getPaintFlags()| Paint.STRIKE_THRU_TEXT_FLAG);
textView.setText(sampleText);
```

Результат: ~~это тестовый удар~~

Зачеркивать только части текста

```
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);
```

Вывод: ~~Это~~ испытание удара

Настройка темы и стиля

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>
```

Сделать RelativeSizeSpan выровненным вверх

Чтобы сделать выравнивание `RelativeSizeSpan` сверху, пользовательский класс может быть получен из класса `SuperscriptSpan` . В следующем примере производный класс называется `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" />
```

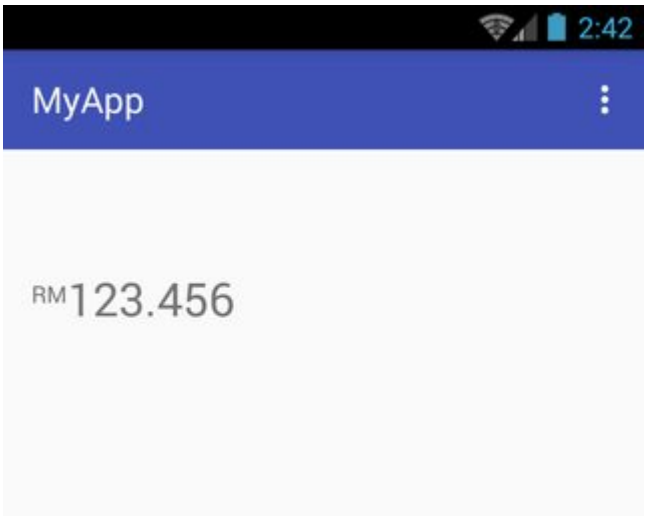
MainActivity.java:

```
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 );
    }
}
```

Справочный снимок экрана:



Pinchzoom на 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 с двумя разными цветами

Цветной текст можно создать, передав текст и имя шрифта в следующую функцию:

```
private String getColoredSpanned(String text, String color) {
    String input = "<font color=" + color + ">" + text + "</font>";
    return input;
}
```

Затем цветной текст можно настроить на `TextView` (или даже на `Button` , `EditText` и т. Д.), Используя приведенный ниже пример кода.

Сначала определите `TextView` следующим образом:

```
TextView txtView = (TextView)findViewById(R.id.txtView);
```

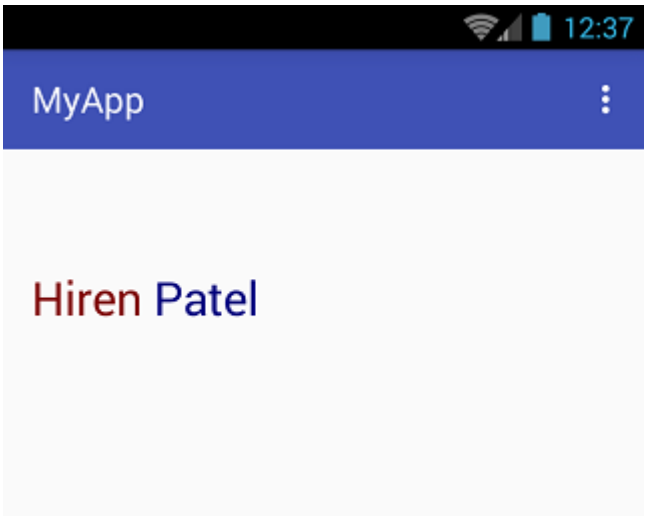
Затем создайте по-разному цветной текст и назначьте его строкам:

```
String name = getColoredSpanned("Hiren", "#800000");
String surName = getColoredSpanned("Patel","#000080");
```

Наконец, установите две разноцветные строки в `TextView` :

```
txtView.setText(Html.fromHtml(name+" "+surName));
```

Справочный снимок экрана:



Прочитайте TextView онлайн: https://riptutorial.com/ru/android/topic/4212/textview

глава 87: TransitionDrawable

Examples

Добавьте переход или Перекрестное затухание между двумя изображениями.

Шаг 1. Создание перехода в XML

Сохраните этот файл `transition.xml` в папке `res/drawable` вашего проекта.

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/image1"/>
    <item android:drawable="@drawable/image2"/>
</transition>
```

Image1 и image2 - это два изображения, которые мы хотим переместить, и их также следует помещать в папку `res/drawable` .

Шаг 2. Добавьте код для ImageView в свой XML-макет, чтобы отобразить вышеприведенное.

```
<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>
```

Шаг 3: Получите доступ к XML-переходу, выведенному в методе onCreate () вашей активности и начните переход в событие 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);
        }
    });
}
```

Анимировать цвет фона представления (цвет переключателя) с помощью 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);
}
```

Прочитайте TransitionDrawable онлайн:
https://riptutorial.com/ru/android/topic/6088/transitiondrawable

глава 88: Typedef Аннотации: @IntDef, @StringDef

замечания

Пакет аннотаций включает в себя ряд полезных аннотаций метаданных, которые вы можете украсить своим собственным кодом, чтобы помочь поймать ошибки.

Просто добавьте зависимость в файле `build.gradle` .

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

Examples

Аннотации IntDef

Эта аннотация гарантирует, что используются только действительные целочисленные константы.
Следующий пример иллюстрирует шаги для создания аннотации:

```
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;
    }
}
```

Они также позволяют завершить код для автоматического предоставления разрешенных констант.
Когда вы создаете этот код, генерируется предупреждение, если параметр типа не ссылается на одну из определенных констант.

Объединение констант с флагами

Используя `IntDef#flag()` установленный в `true` , можно комбинировать несколько констант.

Используя тот же пример в этом разделе:

```
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)

    .....

}
```

Пользователи могут комбинировать разрешенные константы с флагом (например, `|` , `&` , `^`).

Прочитайте Typedef Аннотации: @IntDef, @StringDef онлайн:
https://riptutorial.com/ru/android/topic/4505/typedef-аннотации---intdef---stringdef

глава 89: VectorDrawable и AnimatedVectorDrawable

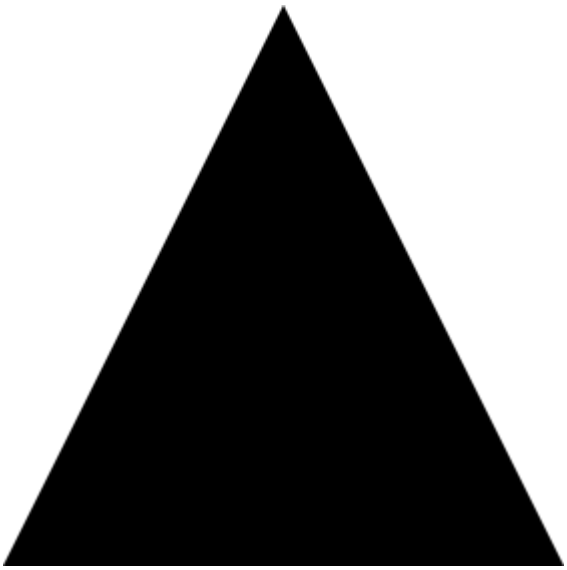
Examples

Basic VectorDrawable

`VectorDrawable` должен состоять как минимум из одного `<path>` определяющего форму

```
<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>
```

Это создаст черный треугольник:



С помощью

А `<clip-path>` определяет форму, которая действует как окно, позволяя только частям `<path>` показать, находятся ли они в форме `<clip-path>` и отключить остальные.

```
<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>
```

В этом случае `<path>` создает черный треугольник, но `<clip-path>` определяет меньшую квадратную форму, позволяя показывать только часть треугольника:

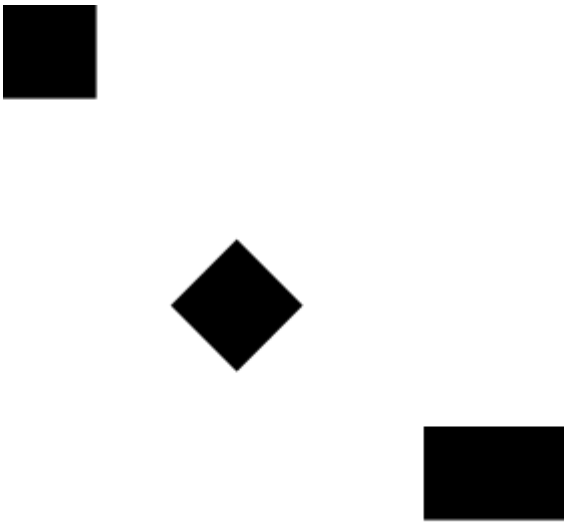
теги

Тег `<group>` позволяет масштабировать, вращать и позицию одного или нескольких элементов `VectorDrawable` для настройки:

```
<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>
```

В приведенном выше примере кода содержится три одинаковых `<path>` , описывающих черные квадраты. Первый квадрат не отрегулирован. Второй квадрат обернут тегом `<group>` который перемещает его и поворачивает на 45 °. Третий квадрат завернут в `<group>` который перемещает его и растягивает его горизонтально на 50%. Результат следующий:



Тег `<group>` может содержать несколько тегов `<path>` и `<clip-path>` . Он может даже содержать другую `<group>` .

Основной анимированный вектор

Для `AnimatedVectorDrawable` требуется как минимум 3 компонента:

- `VectorDrawable` который будет управляться
- `objectAnimator` который определяет, какое свойство изменить и как
- Сам `AnimatedVectorDrawable` , который соединяет `objectAnimator` с `VectorDrawable` для создания анимации

Следующее создает треугольник, который переводит свой цвет с черного на красный.

`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>
```

`objectAnimator` , filename: `color_change_animator.xml`

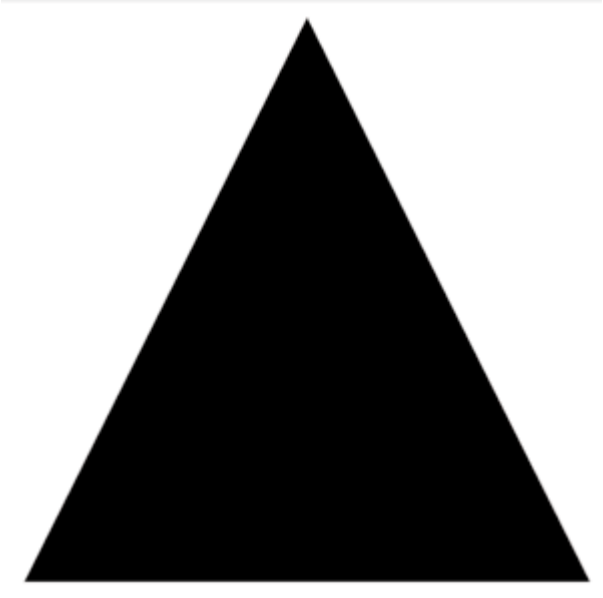
```
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:propertyName="fillColor"
    android:duration="2000"
    android:repeatCount="infinite"
    android:valueFrom="@android:color/black"
    android:valueTo="@android:color/holo_red_light"/>
```

`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>
```

Обратите внимание, что `<target>` указывает `android:name="triangle"` который соответствует `<path>` в `VectorDrawable` . `VectorDrawable` может содержать несколько элементов, а свойство `android:name` используется для определения того, какой элемент был нацелен.

Результат:



Использование штрихов

Использование SVG-штриха облегчает создание векторного рисунка с унифицированной длиной хода в соответствии с принципами Material Design :

> Согласованные мазки тяги являются ключом к объединению общего семейства значков системы. Поддерживайте ширину 2dp для всех инсультов, включая кривые, углы и внутренние и внешние штрихи.

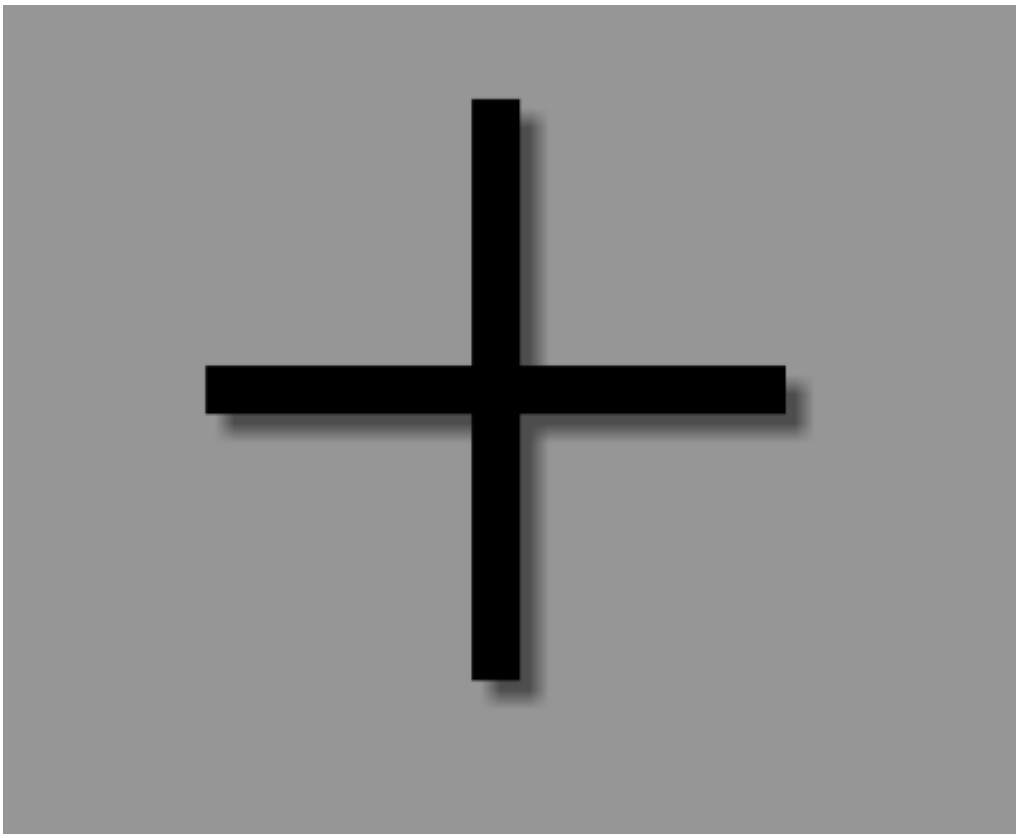
Так, например, так вы можете создать знак «плюс», используя штрихи:

```
<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` определяет цвет штриха.
- `strokeWidth` определяет ширину (в dp) штриха (2dp в этом случае, как указано в рекомендациях).
- `pathData` где мы описываем наше изображение SVG:
- `M12,0` перемещает «курсор» в положение 12,0
- `V24` создает вертикальную линию в положение 12, 24

и т. д., см. документацию по SVG и этот полезный учебник «SVG Path» из w3schools, чтобы узнать больше о конкретных командах пути.

В результате мы получили это без излишеств плюс знак:



Это **особенно полезно** для создания `AnimatedVectorDrawable` , так как теперь вы работаете с одним ударом с единой длиной, а не иначе сложным путем.

Векторная совместимость через AppCompat

Несколько предварительных `build.gradle` в `build.gradle` для векторов для работы вплоть до API 7 для VectorDrawables и API 13 для AnimatedVectorDrawables (с некоторыми предостережениями в настоящее время):

```
//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'
}
```

В вашем `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" />
```

Прочитайте VectorDrawable и AnimatedVectorDrawable онлайн:
https://riptutorial.com/ru/android/topic/1627/vectordrawable-и-animatedvectordrawable

глава 90: VideoView

Examples

Создание видеообъявления

Найдите VideoView в действии и добавьте в него видео.

```
VideoView videoView = (VideoView) .findViewById(R.id.videoView);
videoView.setVideoPath(pathToVideo);
```

Начните воспроизведение видео.

```
videoView.start();
```

Определить VideoView в файле макета XML.

```
<VideoView
    android:id="@+id/videoView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center" />
```

Воспроизвести видео с URL с помощью 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;
      }
});
```

Прочитайте VideoView онлайн: https://riptutorial.com/ru/android/topic/8962/videoview

глава 91: ViewFlipper

Вступление

`ViewFlipper` - это `ViewAnimator` который переключается между двумя или несколькими видами, которые были добавлены к нему. Одновременно отображается только один ребенок. Если требуется, `ViewFlipper` может автоматически переключаться между каждым ребенком с регулярным интервалом.

Examples

ViewFlipper с перемещением изображения

XML-файл:

```
<ViewFlipper
    android:id="@+id/viewflip"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:layout_weight="1"
    />
```

Код 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);
```

```
        }
}
```

Прочитайте ViewFlipper онлайн: https://riptutorial.com/ru/android/topic/9032/viewflipper

глава 92: ViewPager

Вступление

ViewPager - это менеджер макетов, который позволяет пользователю листать страницы слева и справа. Он чаще всего используется совместно с Fragment, что является удобным способом обеспечения и управления жизненным циклом каждой страницы.

замечания

Важно отметить, что использование ViewPager состоит в том, что существуют две разные версии `FragmentPagerAdapter` и `FragmentStatePagerAdapter` .

Если вы используете `android.app.Fragment` native Fragments с помощью FragmentPagerAdapter или FragmentStatePagerAdapter, вам нужно использовать версии поддержки библиотеки v13 для адаптера, то есть `android.support.v13.app.FragmentStatePagerAdapter` .

Если вы используете `android.support.v4.app.Fragment` поддержки `android.support.v4.app.Fragment` с помощью FragmentPagerAdapter или FragmentStatePagerAdapter, вам нужно использовать версии библиотеки поддержки v4 для адаптера, то есть `android.support.v4.app.FragmentStatePagerAdapter` .

Examples

Использование Basic ViewPager с фрагментами

`ViewPager` позволяет отображать несколько фрагментов в активности, которые можно перемещать либо влево, либо вправо. `ViewPager` должен быть фидом либо Представлений, либо фрагментов с помощью `PagerAdapter` .

Есть, однако, еще две конкретные реализации, которые вы найдете наиболее полезными в случае использования фрагментов, которые являются `FragmentPagerAdapter` и `FragmentStatePagerAdapter` . Когда в первый раз необходимо создать экземпляр фрагмента, `getItem(position)` будет вызываться для каждой позиции, для которой требуется создание экземпляра. Метод `getCount()` вернет общее количество страниц, чтобы `ViewPager` знал, сколько фрагментов необходимо `ViewPager` .

И `FragmentPagerAdapter` и `FragmentStatePagerAdapter` хранят кеш фрагментов, которые должен показать `ViewPager` . По умолчанию `ViewPager` попытается сохранить максимум 3 фрагмента, соответствующих `ViewPager` видимому фрагменту, а также рядом с правой и левой. Также `FragmentStatePagerAdapter` сохранит состояние каждого из ваших фрагментов.

Имейте в виду, что обе реализации предполагают, что ваши фрагменты будут сохранять свои позиции, поэтому, если вы сохраните список фрагментов вместо статического числа из них, как вы можете видеть в методе `getItem()` , вам нужно будет создать подкласс `PagerAdapter` и переопределить, по крайней мере, `destroyItem() instantiateItem()` , `destroyItem()` и `getItemPosition()` .

Просто добавьте ViewPager в свой макет, как описано в основном примере :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>
    <android.support.v4.view.ViewPager
        android:id="@+id/vpPager">
    </android.support.v4.view.ViewPager>
</LinearLayout>
```

Затем определите адаптер, который определит, сколько страниц существует и какой фрагмент будет отображаться для каждой страницы адаптера.

```
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

Если вы используете `android.app.Fragment` вы должны добавить эту зависимость:

```
compile 'com.android.support:support-v13:25.3.1'
```

Если вы используете `android.support.v4.app.Fragment` вы должны добавить эту зависимость:

```
compile 'com.android.support:support-fragment:25.3.1'
```

ViewPager с TabLayout

`TabLayout` можно использовать для упрощения навигации.
Вы можете установить вкладки для каждого фрагмента в вашем адаптере, используя `TabLayout.newTab()` но есть еще один удобный и удобный метод для этой задачи, который является `TabLayout.setupWithViewPager()` .

Этот метод будет синхронизироваться, создавая и удаляя вкладки в соответствии с содержимым адаптера, связанного с вашим `ViewPager` каждый раз, когда вы его вызываете. Кроме того, он будет устанавливать обратный вызов, чтобы каждый раз, когда пользователь переворачивает страницу, будет выбрана соответствующая вкладка.

Просто определите макет

```
<?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>
```

Затем выполните `FragmentPagerAdapter` и примените его к `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 с PreferenceFragment

До недавнего времени использование `android.support.v4.app.FragmentPagerAdapter` предотвратило бы использование `PreferenceFragment` как одного из фрагментов, используемых в FragmentPagerAdapter.

В последних версиях библиотеки поддержки v7 теперь есть класс `PreferenceFragmentCompat` , который будет работать с ViewPager и версией версии FragmentPagerAdapter версии v4.

Пример фрагмента, который расширяет `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) {

    }
}
```

Теперь вы можете использовать этот фрагмент в подклассе `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;
        }
    }

    // .......

}
```

Добавление ViewPager

Убедитесь, что в файл `build.gradle` вашего приложения добавлена `build.gradle` зависимость:

```
compile 'com.android.support:support-core-ui:25.3.0'
```

Затем добавьте `ViewPager` в свой макет действий:

```
<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
```

Затем определите свой `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;
    }
}
```

Наконец, настройте `ViewPager` в своей деятельности:

```
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 с индикатором точек

Все, что нам нужно: ViewPager , TabLayout и 2 чертежа для выбранных и стандартных точек.

Во-первых, мы должны добавить `TabLayout` в наш макет экрана и подключить его к `ViewPager` . Мы можем сделать это двумя способами:

Вложенная TabLayout в 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>
```

> В этом случае `TabLayout` будет автоматически подключаться к `ViewPager` , но `TabLayout` будет рядом с `ViewPager` , а не над ним.

Отдельный TabLayout

```
<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"/>
```

> В этом случае мы можем поместить `TabLayout` любом месте, но мы должны подключать `TabLayout` с помощью `ViewPager` программно

```
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);
```

Как только мы создали наш макет, мы должны подготовить наши точки. Поэтому мы создаем три файла: `selected_dot.xml` , `default_dot.xml` и `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>
```

Теперь нам нужно добавить только 3 строки кода в `TabLayout` в наш макет xml, и все готово.

```
app:tabBackground="@drawable/tab_selector"
app:tabGravity="center"
app:tabIndicatorHeight="0dp"
```

Настройка OnPageChangeListener

Если вам необходимо отслеживать изменения на странице выбранного вы можете реализовать `ViewPager.OnPageChangeListener` слушателя на 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
    }
});
```

Прочитайте ViewPager онлайн: https://riptutorial.com/ru/android/topic/692/viewpager

глава 93: WebView

Вступление

WebView - это представление, отображающее веб-страницы внутри вашего приложения. При этом вы можете добавить свой собственный URL.

замечания

Не забудьте добавить разрешение в файл манифеста Android

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

Examples

Диалоги оповещений JavaScript в WebView - как заставить их работать

По умолчанию WebView не реализует диалоговые окна предупреждения JavaScript, то есть. `alert()` ничего не сделает. Чтобы вам было нужно сначала включить JavaScript (очевидно ..), а затем установить `WebChromeClient` для обработки запросов на диалоговые окна предупреждений со страницы:

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

Здесь мы переопределяем `onJsAlert` , а затем `onJsAlert` к супер-реализации, которая дает нам стандартный диалог Android. Вы также можете использовать сообщение и URL-адрес самостоятельно, например, если вы хотите создать пользовательское диалоговое окно стиля или если вы хотите их зарегистрировать.

Общение с Javascript на Java (Android)

Активность 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");
    }
}
```

Обработчик Java Javascript

```
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
    }
}
```

Веб-страница, вызов Javascript

```
<script>
...
Android.jsCallback();
...
Android.jsCallback('hello test');
...
</script>
```

Дополнительный совет

При переходе в сложную структуру данных возможным решением является использование JSON.

```
Android.jsCallback('{ "fake-var" : "fake-value", "fake-array" : [0,1,2] }');
```

На стороне Android используйте свой любимый парсер JSON, то есть: JSONObject

Общение с Java на Javascript

Основной пример

```
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!')");
   }
}
```

Пример открытого набора

Если веб-страница a содержит номер телефона, вы можете позвонить, используя дозвон вашего телефона. Этот код проверяет URL-адрес, который начинается с tel: затем сделайте намерение открыть дозвонщик, и вы можете позвонить по нажатому номеру телефона:

```
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;
}
```

Устранение неполадок WebView путем печати сообщений консоли или удаленной отладки

Печать сообщений консоли webview для logcat

Чтобы обрабатывать `console` сообщения с веб-страницы, вы можете переопределить `onConsoleMessage` в `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;
    }
}
```

И установите его в своей деятельности или фрагменте:

```
webView.setWebChromeClient(new ChromeClient());
```

Итак, эта примерная страница:

```
<html>
<head>
    <script type="text/javascript">
        console.log('test message');
    </script>
</head>
<body>
</body>
</html>
```

будет записывать журнал «тестовое сообщение» на logcat:

> WebView: тестовое сообщение sample.html: 4

`console.info()` , `console.warn()` и `console.error()` также поддерживаются хром-клиентом.

Удаленное отладочное устройство Android с Chrome

Ваш удаленный отлаживающий веб-приложение на основе вашего браузера Chrome.

Включить отладку USB на устройстве Android.

На устройстве Android откройте «Настройки», найдите раздел «Параметры разработчика» и включите USB-отладку.

Подключайтесь и обнаруживайте Android-устройство

Открыть страницу в хроме на следующей странице: chrome: // проверка / # устройств

В диалоговом окне «Inspect Devices» выберите ваше устройство и нажмите « **проверить**» . На вашем компьютере разработки открывается новый экземпляр DevTools от Chrome.

Более подробное руководство и описание DevTools можно найти на developers.google.com.

Открыть локальный файл / создать динамический контент в 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" />
```

Загрузка данных в 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");
```

Прочитайте WebView онлайн: https://riptutorial.com/ru/android/topic/153/webview

глава 94: XMPP зарегистрировать логин и чат простой пример

Examples

XMPP зарегистрироваться и основной пример чата

Установите openfire или любой сервер чата в вашей системе или на сервере. Для получения дополнительной информации нажмите здесь.

Создайте проект android и добавьте эти библиотеки в 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'
```

Затем создайте один класс xmpp из назначения соединения 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();
    }
}

}
```

Наконец, добавьте эту активность:

```
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();
    }

}
```

Убедитесь, что в файле манифеста добавлено разрешение на доступ в Интернет.

Прочитайте XMPP зарегистрировать логин и чат простой пример онлайн: https://riptutorial.com/ru/android/topic/6747/xmpp-зарегистрировать-логин-и-чат-простой-пример

глава 95: Xposed

Examples

Создание модуля Xposed

Xposed - это структура, которая позволяет вам перехватывать вызовы методов других приложений. Когда вы делаете модификацию путем декомпиляции APK, вы можете вставлять / изменять команды напрямую, где хотите. Однако после этого вам нужно будет перекомпилировать / подписать APK, и вы можете распространять только весь пакет. С помощью Xposed вы можете ввести свой собственный код до или после методов или полностью заменить целые методы. К сожалению, вы можете установить Xposed на корневые устройства. Вы должны использовать Xposed всякий раз, когда вы хотите манипулировать поведением других приложений или базовой системы Android и не хотите проходить через декомпиляцию, перекомпиляцию и подписку APK.

Во-первых, вы создаете стандартное приложение без Activity в Android Studio.

Затем вы должны включить следующий код в свой *build.gradle* :

```
repositories {
    jcenter();
}
```

После этого вы добавляете следующие зависимости:

```
provided 'de.robv.android.xposed:api:82'
provided 'de.robv.android.xposed:api:82:sources'
```

Теперь вы должны поместить эти теги в тег *приложения,* найденный в *AndroidManifest.xml,* чтобы Xposed распознал ваш модуль:

```
<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" />
```

ПРИМЕЧАНИЕ. Всегда заменяйте **82** последней версией Xposed .

Подключение метода

Создайте новый класс, реализующий `IXposedHookLoadPackage` и реализующий метод `handleLoadPackage` :

```
public class MultiPatcher implements IXposedHookLoadPackage
{
    @Override
    public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
    {

    }
}
```

Внутри метода вы проверяете `loadPackageParam.packageName` для имени пакета приложения, которое хотите подключить:

```
@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }
}
```

Теперь вы можете подключить свой метод и либо манипулировать им до того, как он будет запущен, либо после:

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

Прочитайте Xposed онлайн: https://riptutorial.com/ru/android/topic/4627/xposed

глава 96: Youtube-API

замечания

1. Прежде всего, вам нужно загрузить последнюю банку из ниже ссылки https://developers.google.com/youtube/android/player/downloads/
2. Вы должны включить эту банку в свой проект. Скопируйте и вставьте эту банку в папку libs и не забудьте добавить ее в зависимости от файла gradle {компилировать файлы ('libs / YouTubeAndroidPlayerApi.jar')}
3. Вам нужен ключ api для доступа к youtube api. Перейдите по этой ссылке: https://developers.google.com/youtube/android/player/register, чтобы сгенерировать ваш ключ api.
4. Очистите и создайте свой проект. Теперь вы готовы использовать YoutubeAndroidPlayerApi. Для воспроизведения видео с YouTube вы должны иметь соответствующий ему идентификатор видео, чтобы вы могли воспроизводить его на YouTube. Например, https://www.youtube.com/watch?v=B08iLAtS3AQ , B08iLAtS3AQ - это идентификатор видео, который вам нужно воспроизвести на YouTube.

Examples

Запуск StandAlonePlayerActivity

1. Запуск автономной работы с проигрывателем

```
     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);
```

Активность, расширяющая 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 в портретной активации

Следующий код реализует простой YoutubePlayerFragment. Макет активности заблокирован в портретном режиме, и когда ориентация изменяется или пользователь нажимает полный экран на YoutubePlayer, он поворачивается к lansscape с заполнением экрана YoutubePlayer. YoutubePlayerFragment не требует расширения активности, предоставляемой библиотекой Youtube. Для того, чтобы инициализировать YoutubePlayer, ему необходимо реализовать YouTubePlayer.OnInitializedListener. Итак, класс нашей деятельности следующий

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

YoutubePlayerFragment можно добавить в макет xaml

```
<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>
```

Наконец, вам нужно добавить следующие атрибуты в файл манифеста внутри тега активности

```
android:configChanges="keyboardHidden|orientation|screenSize"
android:screenOrientation="portrait"
```

API YouTube Player

Получение ключа API Android:

Сначала вам понадобится получить отпечаток SHA-1 на вашем компьютере с помощью java keytool. Выполните команду ниже в cmd / terminal, чтобы получить отпечаток 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);
    }
}
```

Теперь создайте файл `Config.java` . В этом файле содержится ключ разработчика API Google Консоли и идентификатор видео 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";
}
```

xml-файл

```
<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" />
```

Использование API данных YouTube для Android

Этот пример поможет вам получить данные списка воспроизведения с помощью API данных YouTube на Android.

Отпечаток SHA-1

Сначала вам нужно получить отпечаток SHA-1 для вашей машины. Существуют различные способы их извлечения. Вы можете выбрать любой метод, указанный в этом Q & A.

Консоль Google API и ключ YouTube для Android

Теперь, когда у вас есть отпечаток SHA-1, откройте консоль Google API и создайте проект.

Перейдите на эту страницу и создайте проект с использованием этого ключа SHA-1 и включите API данных YouTube. Теперь вы получите ключ. Этот ключ будет использоваться для отправки запросов от Android и получения данных.

Горловина

Вам нужно будет добавить следующие строки в ваш файл Gradle для API данных YouTube:

```
compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
```

Чтобы использовать собственный клиент YouTube для отправки запросов, мы должны добавить следующие строки в 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:+'
```

Следующая конфигурация также должна быть добавлена в Gradle, чтобы избежать конфликтов:

```
configurations.all {
    resolutionStrategy.force 'com.google.code.findbugs:jsr305:3.0.2'
}
```

Ниже показано, как будет выглядеть *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:+'
}
```

Теперь идет Java-часть. Поскольку мы будем использовать `HttpTransport` для сетей и `GsonFactory` для преобразования JSON в POJO, нам не нужна какая-либо другая библиотека для отправки каких-либо запросов.

Теперь я хочу показать, как получить плейлисты через API YouTube, предоставив идентификаторы плейлистов. Для этой задачи я буду использовать `AsyncTask` . Чтобы понять, как мы запрашиваем параметры и понимать поток, ознакомьтесь с API данных 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;
    }
}
```

Вышеупомянутая асинхронная задача вернет экземпляр `PlaylistListResponse` который является встроенным классом SDK YouTube. Он имеет все необходимые поля, поэтому нам не нужно создавать POJO.

Наконец, в нашей `MainActivity` нам нужно будет сделать следующее:

```
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);
    }
}
```

Прочитайте Youtube-API онлайн: https://riptutorial.com/ru/android/topic/7587/youtube-api

глава 97: Zip-файл в android

Examples

Почтовый файл на 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();
        }

      }
```

```
    }
```

Прочитайте Zip-файл в android онлайн: https://riptutorial.com/ru/android/topic/8137/zip-файл-в-android

глава 98: Автобус Отто

замечания

Отто устарел в пользу `RxJava` и `RxAndroid` . Эти проекты позволяют одной и той же модели программирования, управляемой событиями, как Otto, но они более способны и обеспечивают лучший контроль за потоками.

Examples

Передача события

В этом примере описывается передача события с использованием шины событий Otto .

Чтобы использовать Автобус Отто в **Android Studio,** вы должны вставить следующий файл в файл gradle:

```
dependencies {
    compile 'com.squareup:otto:1.3.8'
}
```

Событие, которое мы хотели бы передать, - это простой объект Java:

```
public class DatabaseContentChangedEvent {
    public String message;

    public DatabaseContentChangedEvent(String message) {
        this.message = message;
    }
}
```

Для отправки событий нам нужен автобус. Обычно это синглтон:

```
import com.squareup.otto.Bus;

public final class BusProvider {
    private static final Bus mBus = new Bus();

    public static Bus getInstance() {
        return mBus;
    }

    private BusProvider() {
    }
}
```

Для отправки события нам нужен только наш BusProvider, и это метод `post` . Здесь мы отправляем событие, если действие AsyncTask завершено:

```
public abstract class ContentChangingTask extends AsyncTask<Object, Void, Void> {

    ...

    @Override
    protected void onPostExecute(Void param) {
        BusProvider.getInstance().post(
            new DatabaseContentChangedEvent("Content changed")
        );
    }
}
```

Получение события

Для получения события необходимо реализовать метод с типом события в качестве параметра и аннотировать его с помощью `@Subscribe` . Кроме того, вам необходимо зарегистрировать / `BusProvider` регистрацию экземпляра вашего объекта в `BusProvider` (см. Пример « *Отправка события*»):

```
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);
    }
}
```

Важно. Чтобы получить это событие, должен существовать экземпляр класса. Обычно это не тот случай, когда вы хотите отправить результат из одной активности в другую. Поэтому проверьте свой вариант использования для шины событий.

Прочитайте Автобус Отто онлайн: https://riptutorial.com/ru/android/topic/6068/автобус-отто

глава 99: Автообновление TextViews

Вступление

TextView, который автоматически изменяет размер текста в соответствии с его границами.

Android O позволяет вам инструктировать TextView, чтобы размер текста расширялся или сокращался автоматически, чтобы заполнить его макет на основе характеристик и границ TextView.

Вы можете настроить автоматизацию TextView в коде или XML.

Существует два способа установки автоматической обработки TextView: **гранулярность** и **предустановленные размеры**

Examples

Зернистость

В Java:

Вызвать метод `setAutoSizeTextTypeUniformWithConfiguration()` :

```
setAutoSizeTextTypeUniformWithConfiguration(int autoSizeMinTextSize, int autoSizeMaxTextSize,
int autoSizeStepGranularity, int unit)
```

В XML:

Используйте `autoSizeMinTextSize` , `autoSizeMaxTextSize` и `autoSizeStepGranularity` для установки параметров автоматического калибровки в XML-файле макета:

```
<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" />
```

Для получения дополнительной информации просмотрите AutosizingTextViews-Demo в GitHub.

Предустановленные размеры

В Java:

Вызвать метод `setAutoSizeTextTypeUniformWithPresetSizes()` :

```
setAutoSizeTextTypeUniformWithPresetSizes(int[] presetSizes, int unit)
```

В XML:

Используйте атрибут `autoSizePresetSizes` в XML-файле макета:

```
<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" />
```

Чтобы получить доступ к массиву в качестве ресурса, определите массив в файле *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>
```

Для получения дополнительной информации просмотрите AutosizingTextViews-Demo в GitHub.

Прочитайте Автообновление TextViews онлайн: https://riptutorial.com/ru/android/topic/9652/автообновление-textviews

глава 100: Аниматоры

Examples

Встряхните анимацию ImageView

Под папкой res создайте новую папку под названием «anim» для хранения ресурсов анимации и поместите ее в эту папку.

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" />
```

Создать пустое действие под названием 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>
```

И метод анимации изображения на Landing.java

```
Context mContext;


 @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            mContext=this;
                setContentView(R.layout.activity_landing);

                AnimateBell();
        }
```

```
public void AnimateBell() {
        Animation shake = AnimationUtils.loadAnimation(mContext, R.anim.shakeanimation);
        ImageView imgBell= (ImageView) findViewById(R.id.imgBell);
        imgBell.setImageResource(R.mipmap.ic_notifications_active_white_48dp);
        imgBell.setAnimation(shake);
    }
```

Fade in / out animation

Чтобы получить представление о постепенном исчезновении или выходе из вида, используйте `ObjectAnimator` . Как видно из приведенного ниже кода, установите длительность с использованием `.setDuration(millis)` где параметр `millis` - это продолжительность (в миллисекундах) анимации. В приведенном ниже коде представления будут исчезать в / из более 500 миллисекунд или 1/2 секунды. Для того, чтобы начать `ObjectAnimator` анимации «s, вызовите `.start()` . После завершения анимации вызывается `onAnimationEnd(Animator animation)` . Вот хорошее место, чтобы настроить видимость `View.GONE` или `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();
}
```

Анимация TransitionDrawable

В этом примере показана транзакция для представления изображения только с двумя изображениями (можно использовать больше изображений, а затем один за другим для позиций первого и второго уровней после каждой транзакции в виде цикла)

- добавить массив изображений в `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` представляет простой способ анимировать значение (определенного типа, например, `int` , `float` и т. Д.).

Обычный способ использования:

1. Создайте `ValueAnimator` который будет анимировать значение от `min` до `max`
2. Добавьте `UpdateListener` в котором вы будете использовать рассчитанное анимированное значение (которое вы можете получить с помощью `getAnimatedValue()`)

Существует два способа создания `ValueAnimator` :

(пример кода анимируется `float` от `20f` до `40f` в `250ms`)

1. Из `xml` (поместите его в `/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. Из кода:

```
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` является подклассом `ValueAnimator` с добавленной возможностью установить вычисленное значение для свойства `target View` .

Как и в `ValueAnimator` , существует два способа создания `ObjectAnimator` :

(пример кода оживляет `alpha View` с `0.4f` до `0.2f` в `250ms`)

1. Из `xml` (поместите его в `/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. Из кода:

```
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` является упрощенным и оптимизированным способом анимировать свойства `View` .

Каждый `View` имеет объект `ViewPropertyAnimator` доступный через метод `ViewPropertyAnimator animate()` . Вы можете использовать это для одновременного анимации нескольких свойств с помощью простого вызова. Каждый метод `ViewPropertyAnimator` задает **целевое** значение определенного параметра, который должен представлять `ViewPropertyAnimator` .

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

Примечание. Вызов `start()` объекта `ViewPropertyAnimator` **НЕ** является обязательным. Если вы этого не сделаете, вы просто позволяете платформе обрабатывать запуск анимации в соответствующее время (следующий проход обработки анимации). Если вы действительно это сделаете (call `start()`), вы убедитесь, что анимация запускается немедленно.

Развернуть и свернуть анимацию View

```
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);
        }
    }
```

Прочитайте Аниматоры онлайн: https://riptutorial.com/ru/android/topic/1829/аниматоры

глава 101: Анимированная панель предупреждения AlertDialog

Вступление

Анимированный диалог оповещений, который отображает некоторые эффекты анимации. Вы можете получить некоторую анимацию для диалогового окна, такого как Fadein, SlideLift, Slidetop, SlideBottom, Sliderright, Fall, Newspager, Fliph, Flipv, RotateBottom, RotateLeft, Slit, Shake, Sidefill, чтобы сделать ваш приложение привлекательное ..

Examples

Поместите ниже код для анимационного диалога ...

```
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();
    }
}
```

Добавьте приведенные ниже строки в build.gradle, чтобы включить NifyBuilder (CustomView)

build.gradle

```
 dependencies {

     compile 'com.nineoldandroids:library:2.4.0'

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

 }
```

Ссылка ссылки: https://github.com/sd6352051/NiftyDialogEffects

Прочитайте Анимированная панель предупреждения AlertDialog онлайн: https://riptutorial.com/ru/android/topic/10654/анимированная-панель-предупреждения-alertdialog

глава 102: Аппаратная кнопка События / намерения (PTT, LWP и т. Д.)

Вступление

У некоторых устройств Android есть пользовательские кнопки, добавленные производителем. Это открывает новые возможности для разработчиков при работе с этими кнопками, особенно при создании приложений, предназначенных для аппаратных устройств.

В этом разделе находятся кнопки документов, которые имеют намерения, которые вы можете прослушивать с помощью приемников-приемников.

Examples

Устройства Sonim

У устройств Sonim различные модели варьируются от разных пользовательских кнопок:

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
```

Регистрация кнопок

Чтобы получить эти намерения, вам придется назначить кнопки в приложении в настройках телефона. Sonim имеет возможность автоматически регистрировать кнопки в приложении, когда он установлен. Для этого вам нужно будет связаться с ними и получить ключ для конкретного пакета, который будет включен в ваш манифест следующим образом:

```
<meta-data
    android:name="app_key_green_data"
    android:value="your-key-here" />
```

Устройства RugGear

Кнопка PTT

```
android.intent.action.PTT.down
android.intent.action.PTT.up
```

Подтверждено по: RG730, RG740A

Прочитайте Аппаратная кнопка События / намерения (PTT, LWP и т. Д.) онлайн: https://riptutorial.com/ru/android/topic/10418/аппаратная-кнопка-события---намерения--ptt--lwp-и-т--д--

глава 103: Архитектура MVP

Вступление

В этом разделе будет представлена архитектура Model-View-Presenter (MVP) для Android с различными примерами.

замечания

Существует много способов создания приложения для Android. Но не все из них проверяются и позволяют нам структурировать наш код, чтобы приложение было легко протестировано. Ключевой идеей проверяемой архитектуры является разделение частей приложения, что упрощает их обслуживание, расширение и тестирование отдельно друг от друга.

Определение MVP

модель

В приложении с хорошей многоуровневой архитектурой эта модель будет только шлюзом для домена или бизнес-логики. Смотрите его как поставщика данных, которые мы хотим отобразить в представлении.

Посмотреть

Просмотр, обычно реализуемый `Activity` или `Fragment` , будет содержать ссылку на *презентатора* . Единственное, что будет делать просмотр, это вызвать метод из Presenter каждый раз, когда есть действие интерфейса.

Ведущий

Ведущий несет ответственность за то, чтобы выступать в роли среднего человека между View и Model. Он извлекает данные из модели и возвращает их в формат View. Но в отличие от типичного MVC, он также решает, что происходит, когда вы взаимодействуете с View.

* Определения из статьи Антонио Лейва.

Рекомендуемая структура приложения (не требуется)

Приложение должно быть структурировано по *каждой функции* . Это улучшает читаемость и модулирует приложение таким образом, что его части могут быть изменены независимо друг от друга. Каждая ключевая функция приложения находится в собственном пакете Java.

Examples

Пример входа в шаблон модели Presenter (MVP)

Посмотрим MVP в действии, используя простой экран входа в систему. Для входа в систему есть две `Button` s-one, а другая - для экрана регистрации; два `EditText` s-one для электронной почты, а другой для пароля.

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 (Ведущий)

```
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);
    }
}
```

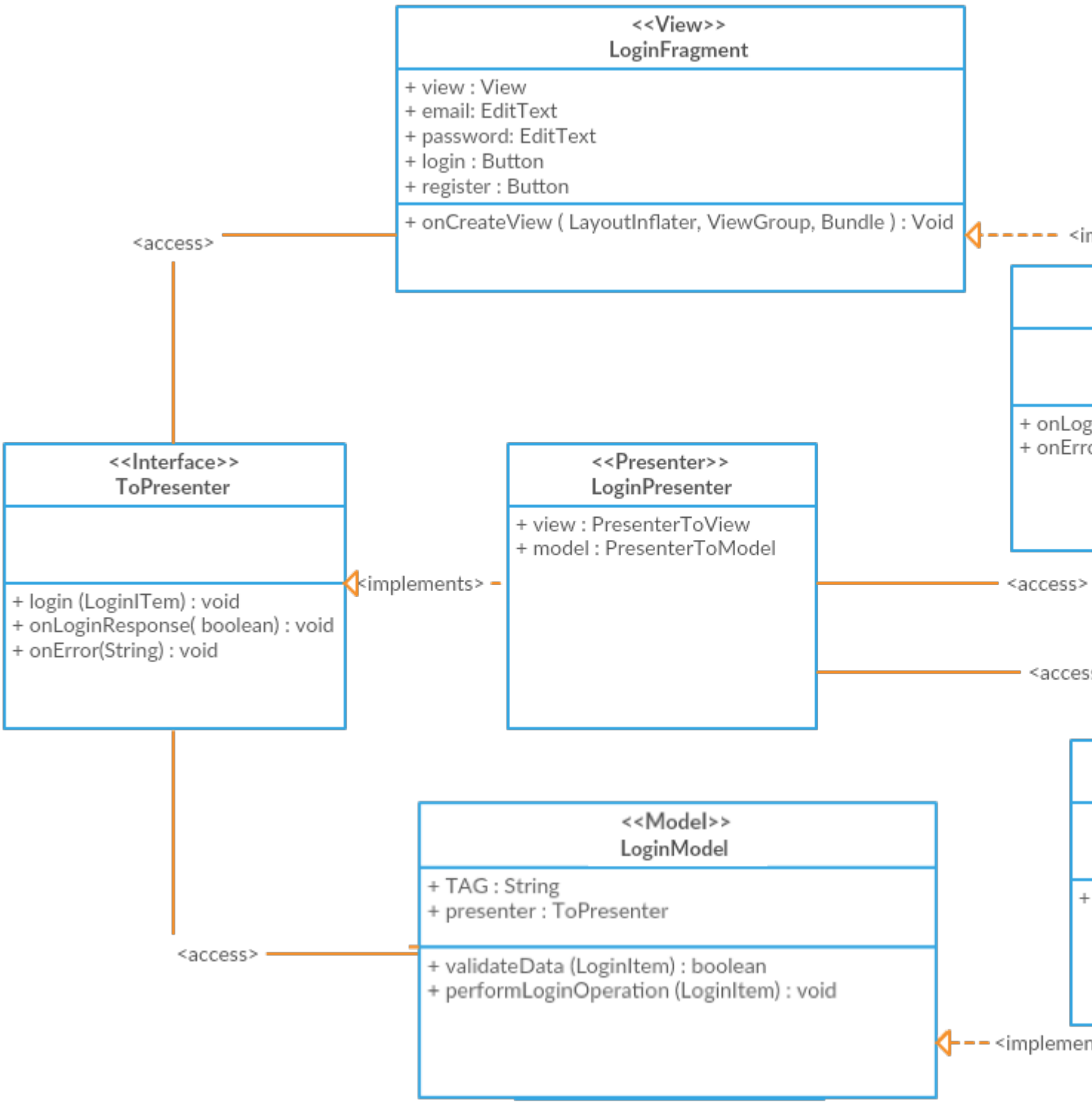
LoginModel (Модель)

```
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);
    }
}
```

Диаграмма классов

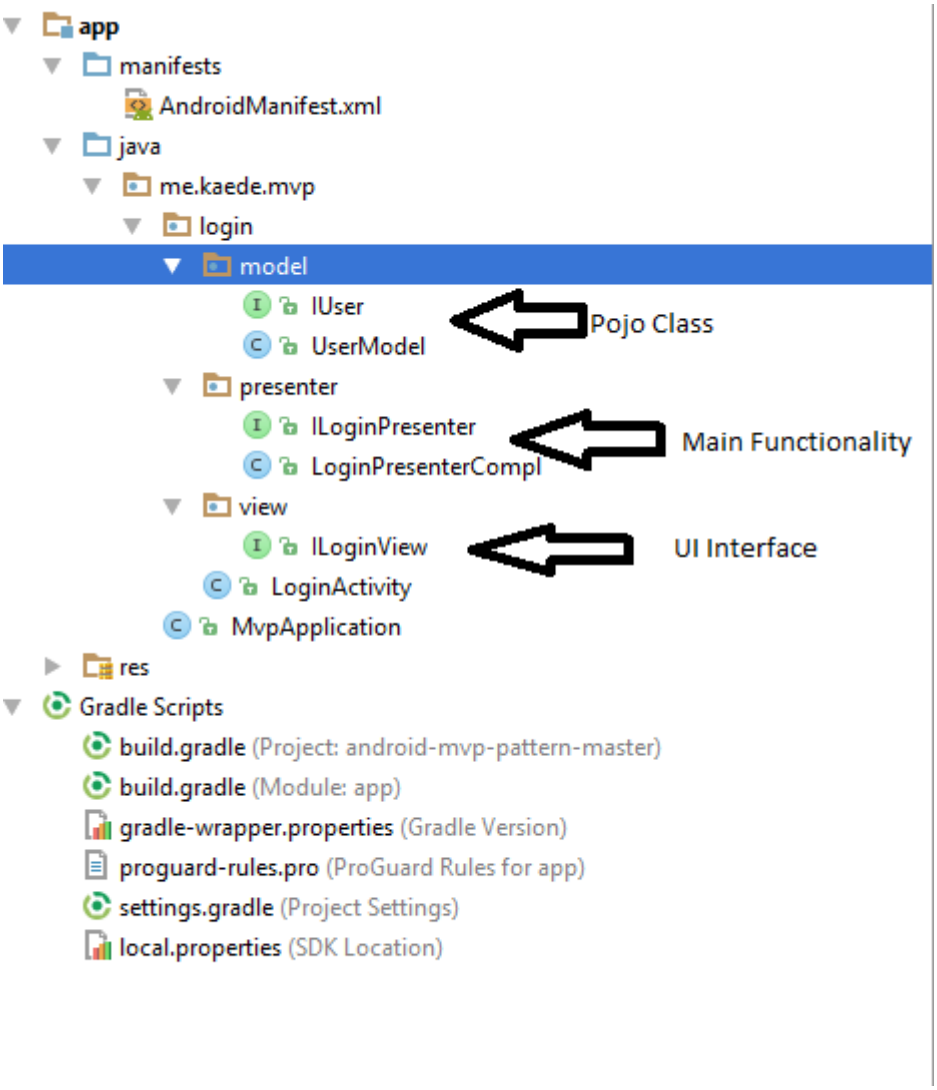
Рассмотрим действие в виде диаграммы классов.



Заметки:

- В этом примере используется волейбол для сетевой связи, но эта библиотека не требуется для MVP
- `UrlUtils` - это класс, который содержит все ссылки для моих конечных точек API
- `ResponseErrorListener.ErrorListener` - это `interface` который прослушивает ошибку в `ErrorResponse` которая `implements Response.ErrorListener` Volley; эти классы здесь не включены, поскольку они не являются непосредственно частью этого примера

Требуемая структура упаковки



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>
```

Класс активности 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);
    }
```

```
}
```

Создание интерфейса ILoginView

Создайте интерфейс `ILoginView` для информации о обновлении из папки просмотра Presenter в виде:

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

Создание интерфейса ILoginPresenter

Создайте интерфейс `ILoginPresenter` для связи с `LoginActivity` (Views) и создайте класс `LoginPresenterCompl` для обработки функций входа в систему и отправки отчетов в Activity. Класс `LoginPresenterCompl` реализует интерфейс `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");
    }
}
```

Создание UserModel

Создайте `UserModel` который похож на класс `LoginActivity` для `LoginActivity` . Создание `IUser` интерфейс для Pojo валидаций:

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;
}
```

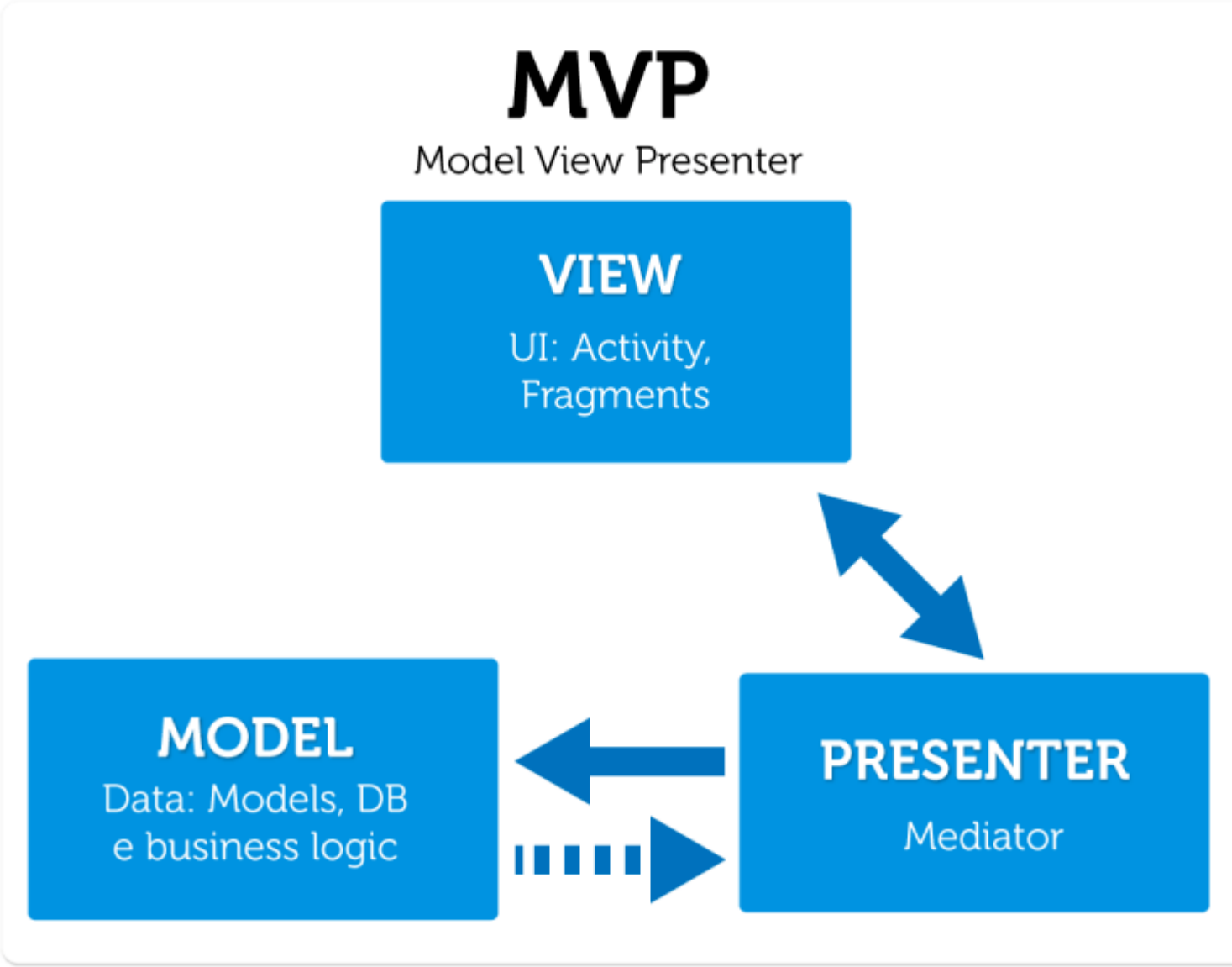
IUser.class

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

    String getPasswd();

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

MVP

Model-view-presenter (MVP) - это вывод архитектурного шаблона model-view-controller (MVC). Он используется в основном для создания пользовательских интерфейсов и предлагает следующие преимущества:

- Представления больше отделены от моделей. Ведущий является посредником между Model и View.
- Легче создавать модульные тесты.
- Как правило, существует взаимно однозначное сопоставление между View и Presenter, с возможностью использования нескольких презентаторов для сложных представлений.



Прочитайте Архитектура MVP онлайн: https://riptutorial.com/ru/android/topic/4615/архитектура-mvp

глава 104: Атрибуты инструментов

замечания

Android имеет выделенное пространство имен XML, предназначенное для инструментов, позволяющих записывать информацию в файл XML.

URI пространства имен:

`http://schemas.android.com/tools` и обычно связан с `tools:` префикс.

Examples

Атрибуты компоновки времени разработки

Эти атрибуты используются, когда макет отображается в Android Studio, но не влияет на время выполнения.

В общем, вы можете использовать любой атрибут фреймворка Android, просто используя пространство имен `tools:` namespace, а не пространство `android:` namespace для предварительного просмотра макета. Вы можете добавить атрибут `android:` namespace (который используется во время выполнения) и атрибут соответствия `tools:` атрибут (который переопределяет атрибут runtime только в предварительном просмотре макета).

Просто определите пространство имен инструментов, как описано в разделе примечаний.

Например, `text` атрибут:

```
<EditText
      tools:text="My Text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
```

Или атрибут `visibility` для отмены просмотра для предварительного просмотра:

```
<LinearLayout
        android:id="@+id/ll1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:visibility="gone" />
```

Или атрибут `context` для связывания макета с активностью или фрагментом

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    tools:context=".MainActivity" >
```

Или атрибут `showIn` чтобы увидеть и включить предварительный просмотр макета в другом макете

```
<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" />
```

Прочитайте Атрибуты инструментов онлайн: https://riptutorial.com/ru/android/topic/1676/атрибуты-инструментов

глава 105: База данных Firebase Realtime

замечания

Другие связанные темы:

- Firebase

Examples

Firebase Realtime DataBase обработчик событий

Первая инициализация FirebaseDatabase:

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

Напишите в базу данных:

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Читайте в базе данных:

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

Получить данные о событиях 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);
```

Быстрая установка

1. Заполните раздел « Установка и настройка», чтобы подключить ваше приложение к Firebase.
   Это создаст проект в Firebase.

2. Добавьте зависимость для базы данных Firebase Realtime к вашему файлу `build.gradle` уровне `build.gradle` :

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

3. Настройка правил базы данных Firebase

Теперь вы готовы работать с базой данных Realtime в Android.

Например, вы пишете сообщение `Hello World` в базу данных под ключом `message` .

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Проектирование и понимание того, как извлекать данные в реальном

времени из базы данных Firebase

В этом примере предполагается, что вы уже создали базу данных Firebase Realtime. Если вы начинаете, пожалуйста, сообщите здесь, как добавить Firebase в свой проект Android.

Во-первых, добавьте зависимость базы данных Firebase к файлу *build.gradle на* уровне *приложения* :

```
compile 'com.google.firebase:firebase-database:9.4.0'
```

Теперь давайте создадим чат-приложение, которое хранит данные в базе данных Firebase.

Шаг 1. Создайте класс с именем Чат

Просто создайте класс с некоторыми основными переменными, необходимыми для чата:

```
public class Chat{
    public String name, message;
}
```

Шаг 2: Создайте некоторые данные JSON

Для отправки / получения данных в / из базы данных Firebase вам необходимо использовать JSON. Предположим, что некоторые чаты уже хранятся на корневом уровне в базе данных. Данные этих чатов могут выглядеть следующим образом:

```
[
    {
        "name":"John Doe",
        "message":"My first Message"
    },
    {
        "name":"John Doe",
        "message":"Second Message"
    },
    {
        "name":"John Doe",
        "message":"Third Message"
    }
]
```

Шаг 3: Добавление слушателей

Существует три типа слушателей. В следующем примере мы будем использовать `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();
    }
});
```

Шаг 4: добавление данных в базу данных

Просто создайте объект класса Chat и добавьте значения следующим образом:

```
Chat chat=new Chat();
chat.name="John Doe";
chat.message="First message from android";
```

Теперь получите ссылку на узел чатов, как это сделано в сеансе извлечения:

```
DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference().child("chats");
```

Прежде чем приступать к добавлению данных, имейте в виду, что вам нужна еще одна глубокая ссылка, поскольку узел чата имеет еще несколько узлов и добавление нового чата означает добавление нового узла, содержащего детали чата. Мы можем сгенерировать новое и уникальное имя узла, используя функцию `push()` объекта `DatabaseReference` , которая вернет другой `DatabaseReference` , который, в свою очередь, указывает на вновь образованный узел, чтобы вставить данные чата.

пример

```
// The parameter is the chat object that was newly created a few lines above.
chatDb.push().setValue(chat);
```

Функция `setValue()` будет обеспечивать, чтобы все функции `onDataChanged` приложения `onDataChanged` (включая одно и то же устройство), что является приложенным слушателем узла «чаты».

Денормализация: плоская структура базы данных

Денормализация и плоская структура базы данных необходимы для эффективной загрузки отдельных вызовов. Со следующей структурой также можно поддерживать двусторонние отношения. Недостатком такого подхода является то, что вам всегда нужно обновлять данные в нескольких местах.

Например, представьте приложение, которое позволяет пользователю хранить сообщения для себя (заметки).

Желаемая плоская структура базы данных:

```
|--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"
```

Используемый класс memo

```
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;
    }
}
```

Получение заметок пользователя

```
//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) { }
    }
```

Создание заметки

```
//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);
```

После нажатия или базы данных выглядит следующим образом:

```
|--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"
```

Понимание базы данных JSON базы Firebase

Прежде чем мы соберём свои руки с кодом, я чувствую, что необходимо понять, как данные хранятся в firebase. В отличие от реляционных баз данных, firebase хранит данные в формате JSON. Подумайте о каждой строке в реляционной базе данных как о объекте JSON (который в основном является неупорядоченной парой ключ-значение). Таким образом, имя столбца становится ключевым, а значение, хранящееся в этом столбце для одной конкретной строки, является значением. Таким образом, вся строка представляется как объект JSON, а список из них представляет собой всю таблицу базы данных. Непосредственная польза, которую я вижу для этого, - это изменение схемы становится намного более дешевой по сравнению со старой РСУБД. Легче добавить еще несколько атрибутов в JSON, чем изменить структуру таблицы.

вот пример JSON, чтобы показать, как данные хранятся в 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
  }
```

Это ясно показывает, как данные, которые мы использовали для хранения в реляционных базах данных, могут храниться в формате JSON. Затем давайте посмотрим, как читать эти данные в устройствах Android.

Извлечение данных из базы

Я собираюсь предположить, что вы уже знаете о добавлении firebase зависимостей градиента в студии android. Если вы не просто следуете инструкциям отсюда . Добавьте свое приложение в консоль firebase, студию градиента синдрома android после добавления зависимостей. Все зависимости не нужны, просто база данных firebase и firebase auth.

Теперь, когда мы знаем, как хранятся данные и как добавить зависимости gradle, посмотрим, как использовать импортированный SDK для firebase android SDK для извлечения данных.

создать ссылку базы данных 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)
```

отсюда вы можете связать несколько вызовов метода child (), чтобы указать на интересующие вас данные. Например, если данные хранятся, как показано в предыдущем разделе, и вы хотите указать на пользователя Bruce Wayne, вы можете использовать:

```
DatabaseReference bruceWayneRef = userDBRef.child("371298");
// 371298 is key of bruce wayne user in JSON structure (previous section)
```

Или просто передайте всю ссылку на объект 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..."
```

Теперь, когда у нас есть ссылка на данные, которые мы хотим получить, мы можем использовать прослушиватели для извлечения данных в приложениях для Android. В отличие от традиционных вызовов, в которых вы запускаете вызовы API REST с использованием модификации или волейбола, здесь необходим простой прослушиватель

обратного вызова для получения данных. Firebase sdk вызывает методы обратного вызова, и все готово.

В основном есть два типа слушателей, которые вы можете присоединить, один - ValueEventListener, а другой - ChildEventListener (описан в следующем разделе). Для любого изменения данных в узле у нас есть ссылки и добавлены слушатели, функция event eventers возвращает всю структуру JSON, а прослушиватель дочерних событий возвращает конкретный ребенок, где произошло изменение. Оба они полезны по-своему. Для извлечения данных из firebase мы можем добавить одного или нескольких слушателей в ссылку базы данных firebase (список userDBRef, который мы создали ранее).

Вот пример кода (объяснение кода после кода):

```
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
    });
```

Вы заметили, что тип класса прошел. DataSnapshot может преобразовывать данные JSON в наши определенные POJO, просто передать правильный тип класса.

Если ваш прецедент не требует целых данных (в нашем случае user_base table) каждый раз, когда происходит какое-то небольшое изменение или вы хотите ***получить данные только один раз*** , вы можете использовать ***метод addListenerForSingleValueEvent ()*** ссылки базы данных. Это срабатывает только один раз.

```
userDBRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // Do something
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Do something about the error
    });
```

Над образцами вы получите значение узла JSON. Чтобы получить ключ, просто позвоните:

```
String myKey = dataSnapshot.getKey();
```

Прослушивание обновлений для детей

Возьмите пример использования, например, приложение чата или приложение для совместного использования продуктов (для которого в основном требуется список объектов для синхронизации между пользователями). Если вы используете базу данных firebase и добавляете прослушиватель событий значения к родительскому узлу чата или родительскому узлу списка продуктов, вы будете заканчивать всю структуру чата с самого начала (я имел в виду начало вашего чата) каждый раз, когда добавляется узел чата (т.е. кто-то говорит привет). То, что мы не хотим делать, нас интересует только новый узел или только старый узел, который был удален или изменен, неизменные не должны возвращаться.

В этом случае мы можем использовать ChildEvenListener . Без дальнейшего рассмотрения, вот пример кода (см. Предыдущие разделы для образцов данных JSON):

```
userDBRef.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {
        //If not dealing with ordered data forget about this
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
    });
```

Имя метода самоочевидно. Как вы можете видеть, когда новый пользователь добавлен или какое-либо свойство существующего пользователя модифицировано, или пользователь удален или удален, соответствующий метод обратного вызова дочернего прослушивателя событий вызывается с соответствующими данными. Поэтому, если вы сохраняете обновленный пользовательский интерфейс для приложения чата, получите JSON из синтаксиса onChildAdded () в POJO и вставьте его в свой пользовательский интерфейс. Просто не забудьте удалить слушателя, когда пользователь покидает экран.

onChildChanged () дает все дочернее значение с измененными свойствами (новые).

onChiledRemoved () возвращает удаленный дочерний узел.

Получение данных с разбиением на страницы

Когда у вас есть огромная база данных JSON, добавление прослушивателя событий

значения не имеет смысла. Он вернет огромный JSON и разберется, это потребует много времени. В таких случаях мы можем использовать разбиение на страницы и выборку данных, а также их отображение или обработку. Вроде как ленивая загрузка или как выбор старых чатов, когда пользователь нажимает на старую чат. В этом случае может использоваться запрос .

Давайте рассмотрим наш старый пример в предыдущих разделах. В базе пользователей есть 3 пользователя, если он произносит 3 сотни тысяч пользователей, и вы хотите получить список пользователей в партиях по 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
    });
```

Здесь можно добавить и прослушать значения или дочерние события. Вызовите запрос еще раз, чтобы получить следующие 50. Не **забудьте добавить метод orderByChild ()** , это не будет работать без этого. Firebase должен знать порядок, по которому вы разбиваете страницы.

Прочитайте База данных Firebase Realtime онлайн:
https://riptutorial.com/ru/android/topic/5511/база-данных-firebase-realtime

глава 106: Безопасность

Examples

Проверка подписи приложения - обнаружение несанкционированного доступа

Этот метод подробно описывает, как обеспечить, чтобы ваш .apk был подписан с вашим сертификатом разработчика, и использует тот факт, что сертификат остается согласованным и что только у вас есть к нему доступ. Мы можем разбить эту технику на 3 простых шага:

- Найдите свою подпись сертификата разработчика.
- Вставьте свою подпись в константу String в приложении.
- Убедитесь, что подпись во время выполнения соответствует нашей встроенной подписи разработчика.

Вот фрагмент кода:

```
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;
```

```
}
```

Прочитайте Безопасность онлайн: https://riptutorial.com/ru/android/topic/4664/безопасность

глава 107: Безопасные общие ресурсы

Вступление

Общие предпочтения - это **XML-файлы на основе ключа** . Он находится под `/data/data/package_name/shared_prefs/<filename.xml>` .

Таким образом, пользователь с привилегиями root может перейти к этому местоположению и может изменить его значения. Если вы хотите защитить значения в своих общих настройках, вы можете написать простой механизм шифрования и дешифрования.

Вы должны знать жесткий, что Shared Preferences никогда не были построены для обеспечения безопасности, это просто простой способ сохранить данные.

Синтаксис

1. public static String encrypt (String input);
2. public static String decrypt (String input);

параметры

параметр	Определение
вход	Значение строки для шифрования или дешифрования.

замечания

Общие настройки никогда не были созданы для обеспечения безопасности, это просто простой способ сохранения данных.

Не рекомендуется использовать общие настройки для хранения важной информации, такой как учетные данные пользователя. Чтобы сохранить учетные данные пользователя (например, пароли), вам необходимо использовать другие методы, такие как `AccountManager` Android.

Examples

Обеспечение совместного использования

Простой код

Здесь, чтобы проиллюстрировать принцип работы, мы можем использовать простое шифрование и дешифрование следующим образом.

```
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));
}
```

Техника внедрения

```
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);
```

Прочитайте Безопасные общие ресурсы онлайн: https://riptutorial.com/ru/android/topic/9887/безопасные-общие-ресурсы

глава 108: Безопасные общие ресурсы

Вступление

Общие предпочтения - это **XML-файлы на основе ключа** . Он находится под / data / data / package_name / shared_prefs / <filename.xml>.

Таким образом, пользователь с привилегиями root может перейти к этому местоположению и может изменить его значения. Если вы хотите защитить значения в своих общих настройках, вы можете написать простой механизм шифрования и дешифрования.

Вы должны знать жесткий, что Shared Preferences никогда не были построены для обеспечения безопасности, это просто простой способ сохранить данные.

Синтаксис

1. public static String encrypt (String input);
2. public static String decrypt (String input);

параметры

параметр	Определение
вход	Значение строки для шифрования или дешифрования.

замечания

Общие настройки никогда не были созданы для обеспечения безопасности, это просто простой способ сохранения данных.

Не рекомендуется использовать общие настройки для хранения важной информации, такой как учетные данные пользователя. Чтобы сохранить учетные данные пользователя (например, пароли), вам необходимо использовать другие методы, такие как `AccountManager` Android.

Examples

Обеспечение совместного использования

Простой код

Здесь, чтобы проиллюстрировать принцип работы, мы можем использовать простое шифрование и дешифрование следующим образом.

```
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));
}
```

Техника внедрения

```
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);
```

Прочитайте Безопасные общие ресурсы онлайн: https://riptutorial.com/ru/android/topic/9890/безопасные-общие-ресурсы

глава 109: Библиотека привязки данных

замечания

Настроить

Прежде чем использовать привязку данных, вы должны включить плагин в `build.gradle` .

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

Примечание: привязка данных была добавлена в плагин Android Gradle версии 1.5.0

Названия классов привязки

Плагин привязки данных генерирует имя класса привязки путем преобразования имени файла макета в файл Pascal и добавления «Связывание» до конца. Таким образом `item_detail_activity.xml` будет генерировать класс с именем `ItemDetailActivityBinding` .

Ресурсы

- Официальная документация

Examples

Связывание основного текстового поля

Конфигурация Gradle (Module: app)

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

Модель данных

```
public class Item {
    public String name;
    public String description;

    public Item(String name, String description) {
        this.name = name;
```

```
        this.description = description;
    }
}
```

Макет XML

Первым шагом является перенос вашего макета в `<layout>` , добавление элемента `<data>` и добавление элемента `<variable>` для вашей модели данных.

Затем вы можете привязать атрибуты XML к полям в модели данных, используя `@{model.fieldname}` , где `model` - это имя переменной, а `fieldname` - поле, к которому вы хотите получить доступ.

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>
```

Для каждого файла макета XML, правильно настроенного с привязками, плагин Android Gradle генерирует соответствующий класс: привязки. Поскольку у нас есть макет с именем *item_detail_activity* , соответствующий сгенерированный класс привязки называется `ItemDetailActivityBinding` .

Затем это связывание можно использовать в Деятельности, например:

```
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);
```

```
    }
}
```

Связывание с методом доступа

Если ваша модель имеет частные методы, библиотека привязки данных по-прежнему позволяет вам получить к ней доступ в вашем представлении без использования полного имени метода.

Модель данных

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Макет 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>
```

Ссылки на классы

Модель данных

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Макет XML

Вы должны импортировать ссылочные классы, как и в 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>
```

Примечание . Пакет `java.lang.*` Автоматически импортируется системой. (То же самое делает `JVM` для `Java`)

Сопоставление данных в фрагменте

Модель данных

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name){
        this.name = name;
    }

}
```

Макет 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>
```

Фрагмент

```
@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();
}
```

Встроенная двусторонняя привязка данных

Двусторонняя привязка данных поддерживает следующие атрибуты:

Элемент	свойства
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

Элемент	свойства
`TimePicker`	• `android:hour` • `android:minute`
`ToggleButton`	`android:checked`
`Switch`	`android:checked`

использование

```
<layout ...>
    <data>
        <variable type="com.example.myapp.User" name="user"/>
    </data>
    <RelativeLayout ...>
        <EditText android:text="@={user.firstName}" .../>
    </RelativeLayout>
</layout>
```

Обратите внимание, что выражение Binding `@={}` **имеет дополнительный** `=` , что необходимо для **двухстороннего связывания** . Невозможно использовать методы в двухсторонних выражениях привязки.

Связывание данных в адаптере RecyclerView

Также возможно использовать привязку данных в вашем адаптере `RecyclerView` .

Модель данных

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

XML-макет

```
<TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@{item.name}"/>
```

Класс адаптера

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

Нажмите прослушиватель с привязкой

Создать интерфейс для clickHandler

```
public interface ClickHandler {
    public void onButtonClick(View v);
}
```

Макет 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>
```

Обработка события в вашей деятельности

```
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();
    }
}
```

Пользовательское событие с использованием выражения лямбда

Определить интерфейс

```
public interface ClickHandler {
    public void onButtonClick(User user);
}
```

Создать класс модели

```
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;
    }
}
```

Макет 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>
```

Код мероприятия:

```
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();
    }
}
```

Для некоторого слушателя, который недоступен в коде xml, но может быть установлен в java-коде, его можно связать с пользовательской привязкой.

Пользовательский класс

```
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);
    }
}
```

Класс обработчика

```
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>
```

Значение по умолчанию в привязке данных

На панели «Предварительный просмотр» отображаются значения по умолчанию для выражений привязки данных, если они предоставлены.

Например :

```
android:layout_height="@{@dimen/main_layout_height, default=wrap_content}"
```

Это займет `wrap_content` при проектировании и будет действовать как `wrap_content` в области предварительного просмотра.

Другим примером является

```
android:text="@{user.name, default=`Preview Text`}"
```

Он отобразит `Preview Text` в области предварительного просмотра, но когда вы запустите его в устройстве / эмуляторе, будет отображаться фактический текст, привязанный к нему.

DataBinding с пользовательскими переменными (int, boolean)

Иногда нам нужно выполнять основные операции, такие как hide / show view, основанные на одном значении, для этой единственной переменной мы не можем создать модель, или это не очень хорошая практика для создания модели для этого. DataBinding поддерживает базовые типы данных для выполнения этих отпечатков.

```
<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>
```

и установить его значение из класса java.

```
binding.setSelected(true);
```

Связывание данных в диалоге

```
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();
}
```

Пропустить виджет в качестве ссылки в BindingAdapter

Макет 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>
```

Метод 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);
}
```

Прочитайте Библиотека привязки данных онлайн: https://riptutorial.com/ru/android/topic/111/библиотека-привязки-данных

глава 110: Биллинг в приложении

Examples

Расходуемые покупки в приложениях

Расходуемые управляемые продукты - это продукты, которые можно купить несколько раз, например, в игровой валюте, играх, бонусах и т. Д.

В этом примере мы собираемся реализовать 4 разных **управляемых продукта:** `"item1", "item2", "item3", "item4"` .

Шаги в резюме:

1. Добавьте библиотеку In-app Billing в свой проект (файл AIDL).
2. Добавьте требуемое разрешение в файл `AndroidManifest.xml` .
3. Разверните подписанный apk в Google Developers Console.
4. Определите свои продукты.
5. Внедрите код.
6. Тестирование In-app Billing (необязательно).

Шаг 1:

Прежде всего, нам нужно будет добавить файл AIDL в ваш проект, как это четко описано в Документации Google здесь .

`IInAppBillingService.aidl` - это файл интерфейса определения интерфейса Android (AIDL), который определяет интерфейс к службе In-app Billing Version 3. Вы будете использовать этот интерфейс для выполнения запросов на выставление счетов путем вызова вызовов метода IPC.

Шаг 2:

После добавления файла AIDL добавьте разрешение BILLING в `AndroidManifest.xml` :

```
<!-- Required permission for implementing In-app Billing -->
<uses-permission android:name="com.android.vending.BILLING" />
```

Шаг 3:

Создайте подписанный apk и загрузите его в Google Developers Console. Это необходимо для того, чтобы мы могли начать определять наши приложения в приложении.

Шаг 4:

Определите все ваши продукты с помощью другого идентификатора продукта и задайте цену для каждого из них. Существует 2 типа продуктов (управляемые продукты и подписки). Как мы уже говорили, мы собираемся реализовать 4 разных расходных **управляемых продукта** `"item1", "item2", "item3", "item4"` .

Шаг 5:

После выполнения всех вышеперечисленных шагов вы готовы начать внедрять сам код в свою собственную деятельность.

Основная деятельность:

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

Шаг 6:

После внедрения кода вы можете протестировать его, развернув apk на бета-альфа-канал и разрешите другим пользователям тестировать код для вас. Тем не менее, реальные покупки в приложении не могут быть выполнены в режиме тестирования. Сначала вы должны опубликовать свое приложение / игру в Play Store, чтобы все продукты были полностью активированы.

Более подробную информацию о тестировании биллинга In-app можно найти здесь .

(Сторонняя) библиотека In-App v3

Шаг 1: Прежде всего, выполните следующие два действия, чтобы добавить функциональность приложения:

1. Добавьте библиотеку, используя:

```
repositories {
            mavenCentral()
        }
        dependencies {
           compile 'com.anjlab.android.iab.v3:library:1.0.+'
        }
```

2. Добавьте разрешение в файл манифеста.

```
<uses-permission android:name="com.android.vending.BILLING" />
```

Шаг 2. Инициализация вашего платежного процессора:

```
BillingProcessor bp = new BillingProcessor(this, "YOUR LICENSE KEY FROM GOOGLE PLAY CONSOLE
HERE", this);
```

и реализовать Обработчик биллинга: BillingProcessor.IBillingHandler, который содержит 4 метода: a. onBillingInitialized (); б. onProductPurchased (String productId, TransactionDetails details): Здесь вам нужно обрабатывать действия, которые необходимо выполнить после успешной покупки c. onBillingError (int errorCode, Throwable error): обрабатывать любую ошибку, возникшую во время процесса покупки. d. onPurchaseHistoryRestored (): для восстановления при покупке приложений

Шаг 3: Как купить продукт.

Чтобы приобрести управляемый продукт:

```
bp.purchase(YOUR_ACTIVITY, "YOUR PRODUCT ID FROM GOOGLE PLAY CONSOLE HERE");
```

И купить подписку:

```
bp.subscribe(YOUR_ACTIVITY, "YOUR SUBSCRIPTION ID FROM GOOGLE PLAY CONSOLE HERE");
```

Шаг 4: Использование продукта.

Чтобы потреблять продукт, просто вызовите метод consumePurchase.

bp.consumePurchase («ИДЕНТИФИКАТОР ПРОДУКТА ИЗ GOOGLE PLAY CONSOLE ЗДЕСЬ»);

Для других методов, связанных с приложением, посетите github

Прочитайте Биллинг в приложении онлайн: https://riptutorial.com/ru/android/topic/2843/биллинг-в-приложении

глава 111: Быстрый способ настройки Retrolambda в проекте Android.

Вступление

Retrolambda - это библиотека, которая позволяет использовать Java-лямбда-выражения, ссылки на методы и предложения try-with-resources на Java 7, 6 или 5.

Модуль Gradle Retrolambda позволяет интегрировать Retrolambda в сборку на основе Gradle. Это позволяет, например, использовать эти конструкции в приложении для Android, поскольку стандартная разработка Android в настоящее время пока не поддерживает Java 8.

Examples

Настройка и пример использования:

Шаги настройки:

1. Загрузите и установите jdk8.
2. Добавьте в основной проект проекта build.gradle следующее:

```
buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
        classpath 'me.tatarka:gradle-retrolambda:3.2.3'
    }
}
```

3. Теперь добавьте это в build.gradle вашего модуля приложения

```
apply plugin: 'com.android.application' // or apply plugin: 'java'
apply plugin: 'me.tatarka.retrolambda'
```

4. Добавьте эти строки в build.gradle вашего модуля приложения, чтобы сообщить IDE уровня языка:

```
android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
```

```
    }
}
```

Пример:

Итак, такие вещи:

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        log("Clicked");
    }
});
```

Станьте этим:

```
button.setOnClickListener(v -> log("Clicked"));
```

Прочитайте Быстрый способ настройки Retrolambda в проекте Android. онлайн: https://riptutorial.com/ru/android/topic/8822/быстрый-способ-настройки-retrolambda-в-проекте-android-

глава 112: вводимый коэффициент

Examples

Оттенок

Выбираемый может быть окрашен определенным цветом. Это полезно для поддержки различных тем в вашем приложении и уменьшения количества доступных файлов ресурсов.

Использование интерфейсных API на SDK 21+:

```
Drawable d = context.getDrawable(R.drawable.ic_launcher);
d.setTint(Color.WHITE);
```

Использование библиотеки android.support.v4 на 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
```

Пожалуйста, не то, что `int color` **не** относится к цветному ресурсу, однако вы не ограничены теми цветами, которые определены в классе «Цвет». Когда у вас есть цвет, определенный в вашем XML, который вы хотите использовать, вы должны сначала получить его значение.

Вы можете заменить использование `Color.WHITE` используя приведенные ниже методы

При настройке более старых API:

```
getResources().getColor(R.color.your_color);
```

Или по более новым целям:

```
ContextCompat.getColor(context, R.color.your_color);
```

Сделать вид с закругленными углами

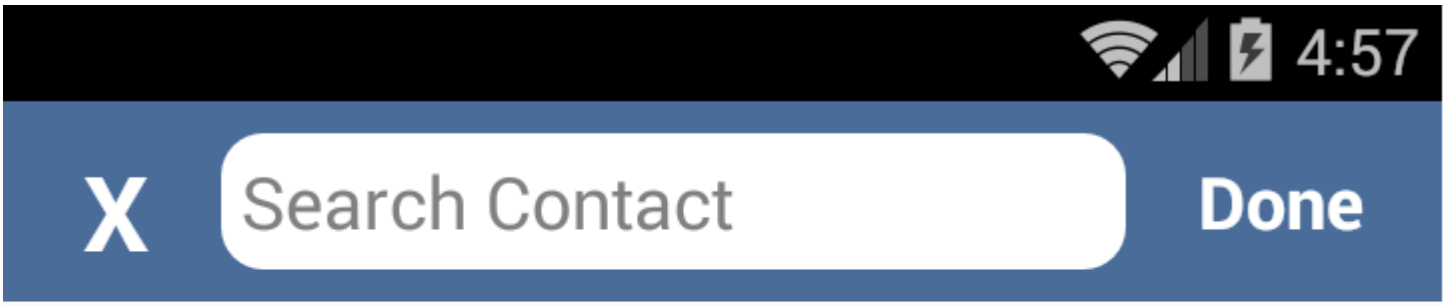
Создать **Drawable** файл с именем , **custom_rectangle.xml** в **вытяжке** папке:

```
<?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>
```

Теперь применим **прямоугольный фон** в **представлении** :

```
mView.setBackGround(R.drawlable.custom_rectangle);
```

Справочный снимок экрана:



Круговой обзор

Для кругового представления (в данном случае `TextView`) создайте drawble **round_view.xml** в папке **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>
```

Назначьте выделение для вида:

```
    <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" />
```

Теперь он должен выглядеть как оранжевый круг:

Пользовательский Drawable

Расширьте свой класс с помощью Drawable и переопределите эти методы

```
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;
    }
}
```

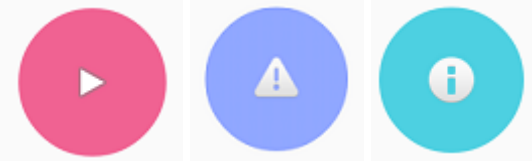
Объявите ImageView в своем макете

```
<ImageView
   android:layout_width="80dp"
   android:id="@+id/imageView"
   android:layout_height="80dp" />
```

Задайте свой графический интерфейс для ImageView

```
 IconDrawable iconDrawable=new
IconDrawable(ContextCompat.getDrawable(this,android.R.drawable.ic_media_play),ContextCompat.getColor(th

 imageView.setImageDrawable(iconDrawable);
```

Скриншот



Прочитайте вводимый коэффициент онлайн: https://riptutorial.com/ru/android/topic/4841/вводимый-коэффициент

глава 113: Векторные иллюстрации

Вступление

Как следует из названия, векторные чертежи основаны на векторной графике. Векторная графика - это способ описания графических элементов с использованием геометрических фигур. Это позволяет создавать чертежи на основе векторной графики XML. Теперь нет необходимости создавать изображение разного размера для mdpi, hdpi, xhdpi и т. Д. С помощью Vector Drawable вам нужно создать изображение только один раз в виде XML-файла, и вы можете масштабировать его для всех dpi и для разных устройств. Это также не экономит место, но также упрощает обслуживание.

параметры

параметр	подробности
`<vector>`	Используется для определения векторного вектора
`<group>`	Определяет группу путей или подгрупп, а также информацию о преобразовании. Преобразования определяются в тех же координатах, что и окно просмотра. И преобразования применяются в порядке масштаба, затем вращаются, а затем переводятся.
`<path>`	Определяет пути для рисования.
`<clip-path>`	Определяет путь к текущему клипу. Обратите внимание, что путь клипа применяется только к текущей группе и ее дочерним элементам.

замечания

Обновите файл **build.gradle** .

```
dependencies {
    ...
   compile 'com.android.support:appcompat-v7:23.2.1'
}
```

Если вы используете **v2.0 или выше плагина Gradle** , добавьте следующий код.

```
// Gradle Plugin 2.0+
 android {
   defaultConfig {
     vectorDrawables.useSupportLibrary = true
```

```
    }
 }
```

Если вы используете **v1.5 или ниже плагина Gradle** , добавьте следующий код.

```
// Gradle Plugin 1.5
 android {
   defaultConfig {
     generatedDensities = []
  }

  // This is handled for you by the 2.0+ Gradle Plugin
  aaptOptions {
    additionalParameters "--no-version-vectors"
  }
 }
```

Подробнее читайте в разделе «Информация о поддержке Android 23.2» .

ПРИМЕЧАНИЕ. Даже с *AppCompat* векторные Drawables не будут работать вне вашего приложения в старых версиях Android. Например, вы не можете передавать векторные чертежи в качестве значков уведомлений, поскольку они обрабатываются системой, а не приложением. См. Этот ответ для обходного пути.

Examples

Пример использования VectorDrawable

Вот пример векторного актива, который мы фактически используем в AppCompat:

Рез / рисуем / 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>
```

С помощью этого чертежа пример объявления `ImageView` будет выглядеть следующим образом:

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    app:srcCompat="@drawable/ic_search"/>
```

Вы также можете установить его во время выполнения:

```
ImageView iv = (ImageView) findViewById(...);
iv.setImageResource(R.drawable.ic_search);
```

Тот же атрибут и вызовы также работают для `ImageButton` .

Пример VectorDrawable xml

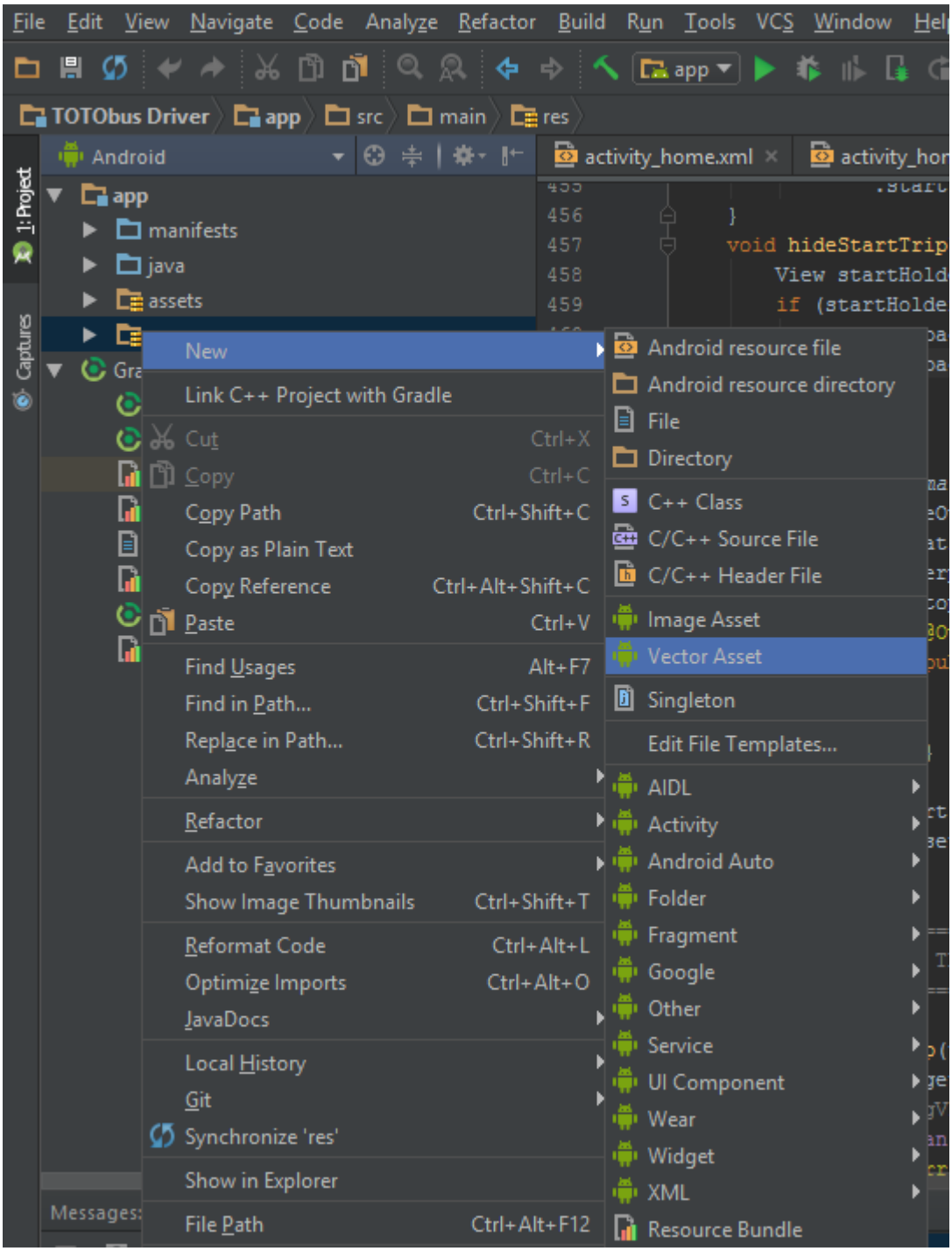
Вот простой `VectorDrawable` в этом файле **vectordrawable.xml** .

```
<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>
```

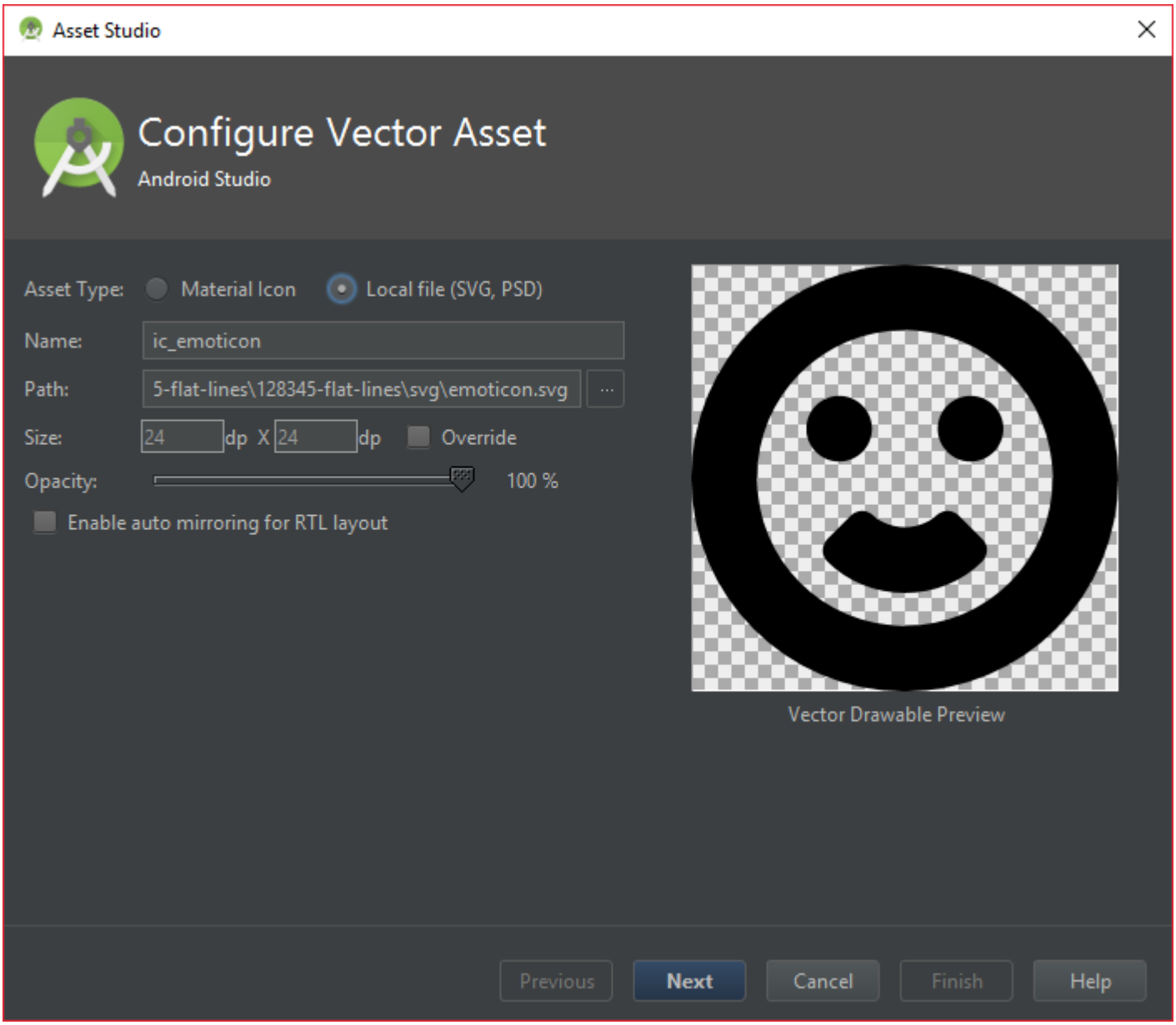
Импорт SVG-файла в формате VectorDrawable

Вы можете импортировать *SVG*- файл в качестве `VectorDrawable` в Android Studio, выполните следующие действия:

«Щелкните правой кнопкой мыши» в папке `res` и выберите **новый** > **Векторный объект** .



Выберите параметр « **Локальный файл**» и перейдите к вашему .svg-файлу. Измените параметры по своему вкусу и нажмите «Далее». Готово.



Прочитайте Векторные иллюстрации онлайн: https://riptutorial.com/ru/android/topic/8194/векторные-иллюстрации

глава 114: Версии Android

замечания

название	Версия для Android	Дата выхода	API уровня	Build.VERSION_CODES
Ангел-пирог (альфа)	1,0	23 сентября 2008 г.	1	БАЗА
Баттенберг (бета)	1,1	9 февраля 2009 г.	2	BASE_1_1
Кекс	1,5	30 апреля 2009 г.	3	CUPCAKE
Пончик	1,6	15 сентября 2009 г.	4	ПОНЧИК
Эклер	2,0	26 октября 2009 г.	5	ЭКЛЕР
	2.0.1	3 декабря 2009 г.	6	ECLAIR_0_1
	2,1	12 января 2010 г.	7	ECLAIR_MR1
Froyo	2,2	20 мая 2010 г.	8	Froyo
Имбирный пряник	2,3	6 декабря 2010 г.	9	ИМБИРНЫЙ ПРЯНИК
	2.3.3	9 февраля 2011 г.	10	GINGERBREAD_MR1
ячеистый	3.0	22 февраля 2011 г.	11	СОТОВОЙ
	3,1	10 мая 2011	12	HONEYCOMB_MR2

название	Версия для Android	Дата выхода	API уровня	Build.VERSION_CODES
		г.		
	3,2	15 июля 2011 г.	13	HONEYCOMB_MR1
Сэндвич с мороженым	4,0	19 октября 2011 г.	14	ICE_CREAM_SANDWICH
	4.0.3	16 декабря 2011 г.	15	ICE_CREAM_SANDWICH_MR1
Жевательные конфеты	4,1	9 июля 2012 г.	16	ЖЕВАТЕЛЬНЫЕ КОНФЕТЫ
	4,2	13 ноября 2012 г.	17	JELLY_BEAN_MR1
	4,3	24 июля 2013 г.	18	JELLY_BEAN_MR2
Кит-Кат	4,4	31 октября 2013 г.	19	КИТ-КАТ
		25 июля 2014 года	20	KITKAT_WATCH
леденец	5.0	17 октября 2014 года	21	ЛЕДЕНЕЦ
	5,1	9 марта 2015 г.	22	LOLLIPOP_MR1
зефирка	6,0	5 октября 2015 г.	23	M
нуга	7,0	22 августа 2016 года	24	N
	7.1.1	5 декабря 2016 года	25	N_MR1

Examples

Проверка версии Android на устройстве во время выполнения

`Build.VERSION_CODES` - это перечисление известных в настоящее время кодов версии SDK.

Чтобы условно запустить код на основе версии Android устройства, используйте аннотацию `TargetApi` чтобы избежать ошибок Lint, и проверьте версию сборки перед запуском кода, специфичного для уровня API.

Ниже приведен пример использования класса, который был введен в API-23, в проекте, который поддерживает уровни API ниже 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);
        //.....................
    }
}
```

Прочитайте Версии Android онлайн: https://riptutorial.com/ru/android/topic/3264/версии-android

глава 115: Версия проекта SDK

Вступление

Приложение Android необходимо запускать на всех устройствах. Каждое устройство может иметь другую версию на Android, работающем на нем.

Теперь каждая версия Android может не поддерживать все функции, которые требуется вашему приложению, и поэтому при создании приложения вам нужно учитывать минимальную и максимальную версию Android.

параметры

параметр	подробности
Версия SDK	Версия SDK для каждого поля - это целое число SDK API для Android. Например, Froyo (Android 2.2) соответствует уровню API 8. Эти целые числа также определены в `Build.VERSION_CODES` .

замечания

В каждом проекте есть четыре версии SDK:

- `targetSdkVersion` - это последняя версия Android, с которой вы протестировали.

  Структура будет использовать `targetSdkVersion` чтобы определить, когда включить определенные поведения совместимости. Например, ориентированный на API уровень 23 или выше позволит вам выбрать модель разрешений времени выполнения .

- `minSdkVersion` - это минимальная версия Android, поддерживаемая вашим приложением. Пользователи, запускающие любую версию Android старше этой версии, не смогут установить ваше приложение или просмотреть его в Play Маркете.

- `maxSdkVersion` - это максимальная версия Android, поддерживаемая вашим приложением. Пользователи, запускающие любую версию Android, новее, чем эта версия, не смогут установить приложение или просмотреть его в Play Маркете. Обычно это не должно использоваться, так как большинство приложений будут работать в новых версиях Android без каких-либо дополнительных усилий.

- `compileSdkVersion` - это версия Android SDK, с которой будет скомпилировано ваше приложение. Как правило, это последняя версия Android, которая была публично выпущена. Это определяет, какие API-интерфейсы вы можете получить при

написании кода. Вы не можете вызывать методы, введенные в уровне API 23, если для вашего `compileSdkVersion` установлено значение 22 или ниже.

Examples

Определение версий SDK проекта

В файле `build.gradle` основного модуля (**приложения**) определите минимальный и целевой номер версии.

```
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
    }
}
```

Прочитайте Версия проекта SDK онлайн: https://riptutorial.com/ru/android/topic/162/версия-проекта-sdk

глава 116: вибрация

Examples

Начало работы с вибрацией

Разрешение на выдачу гранта

перед тем, как вы начнете реализовывать код, вы должны добавить разрешение в манифест андроида:

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

Импортировать библиотеку вибраций

```
import android.os.Vibrator;
```

Получить экземпляр вибратора из контекста

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

Проверить устройство на вибратор

```
void boolean isHaveVibrate(){
    if (vibrator.hasVibrator()) {
        return true;
    }
    return false;
}
```

Вибрировать Неопределенно

используя *вибрирующий (длинный [] шаблон, 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);
```

Вибрационные шаблоны

Вы можете создавать вибрационные шаблоны, передавая массив длин, каждый из которых представляет продолжительность в миллисекундах. Первое число - это время начала. Каждый элемент массива затем чередуется между вибрацией, сном, вибрацией, сном и т. Д.

Следующий пример демонстрирует этот шаблон:

- вибрировать 100 миллисекунд и спать 1000 миллисекунд
- вибрировать 200 миллисекунд и спать 2000 миллисекунд

```
long[] pattern = {0, 100, 1000, 200, 2000};
```

Чтобы заставить шаблон повторяться, перейдите в индекс в массив шаблонов, для которого нужно запустить повтор, или `-1` чтобы отключить повторение.

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(pattern, -1); // does not repeat
vibrator.vibrate(pattern,  0); // repeats forever
```

Остановить вибрацию

Если вы хотите остановить вибрацию, звоните:

```
vibrator.cancel();
```

Вибрация за один раз

используя *вибрацию (длинные миллисекунды)*

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(500);
```

Прочитайте вибрация онлайн: https://riptutorial.com/ru/android/topic/3359/вибрация

глава 117: Виджеты

замечания

SDV

Examples

Декларация манифестации -

`AppWidgetProvider` класс `AppWidgetProvider` в файле `AndroidManifest.xml` вашего приложения. Например:

```
<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>
```

Метаданные

Добавьте метаданные AppWidgetProviderInfo в `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>
```

Класс AppWidgetProvider

Самый важный `AppWidgetProvider` вызов **`onUpdate()`** - **`onUpdate()`** . Он называется каждый раз, когда добавляется 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()` вызывается при размещении или изменении виджета.

`onDeleted(Context, int[])` вызывается при удалении виджета.

Два виджета с разным объявлением макетов

1. Объявите два приемника в файле манифеста:

```
<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. Создайте два макета
    - `@xml/widget_1x1`
    - `@xml/widget_2x2`
3. `UVMateWidget2x2` подкласс `UVMateWidget2x2` из класса `UVMateWidget` с расширенным поведением:

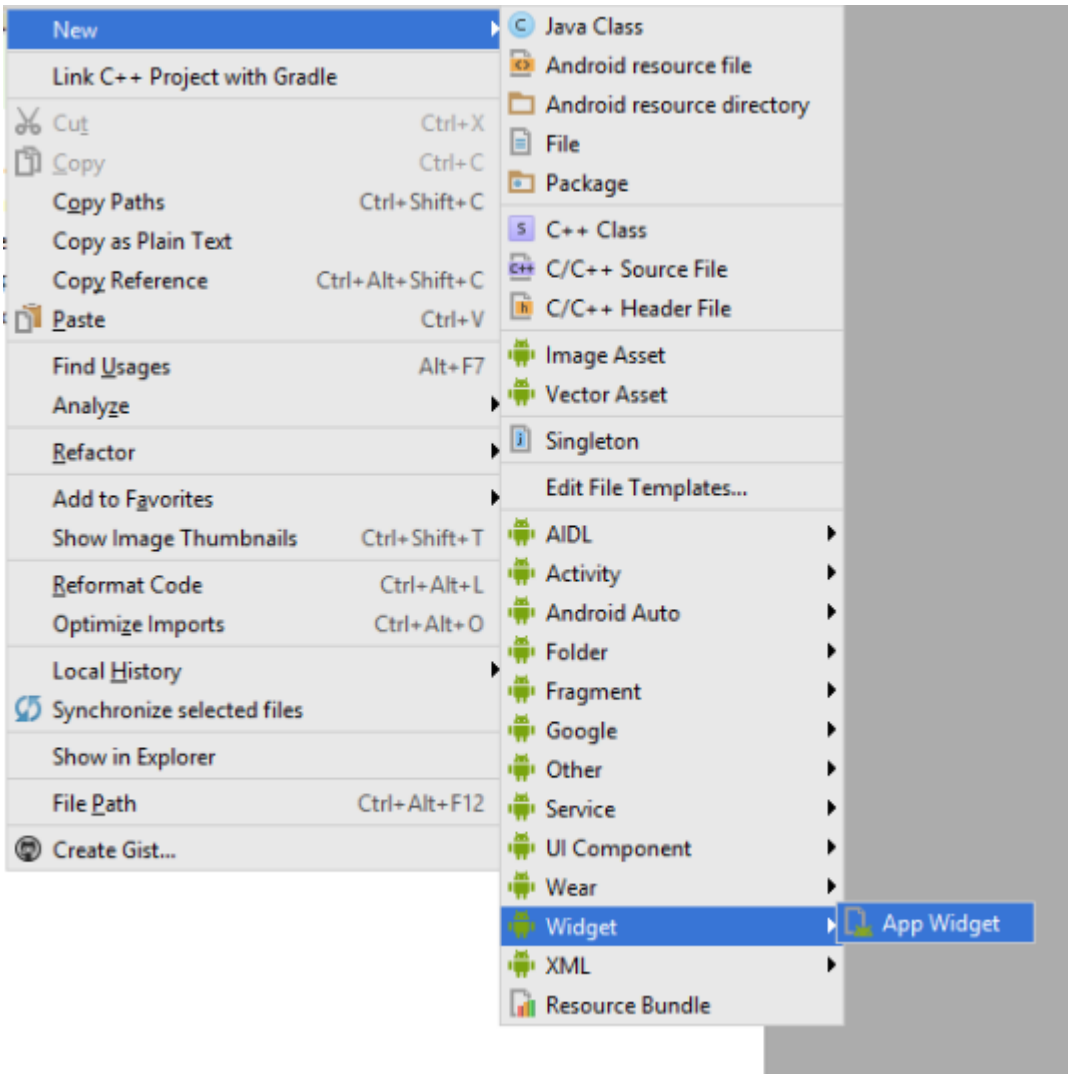
```
package au.com.aershov.uvmate;

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);
    }
}
```

Создание / интеграция базового виджета с помощью Android Studio

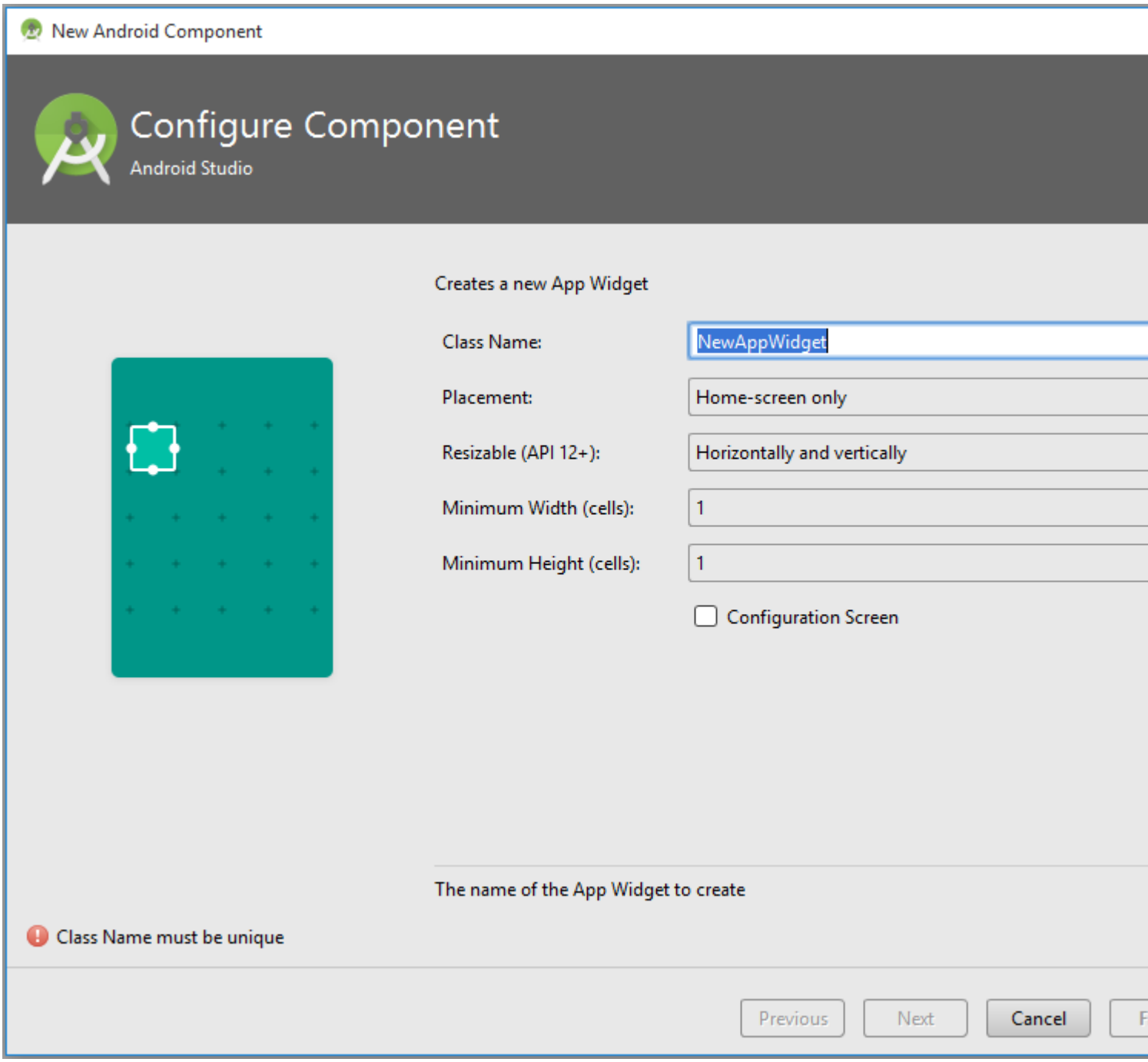
Последняя Android Studio создаст и интегрирует базовый виджет в ваше приложение за 2 шага.

Прямо на вашем приложении ==> Новый ==> Виджет ==> Виджет приложений

,



Он отобразит экран, как показано ниже, и заполните поля



Это сделано.

Он **создаст и интегрирует** в ваше приложение **базовый виджет HelloWorld** (включая файл макета, файл метаданных, декларацию в файле манифеста и т. Д.).

Прочитайте Виджеты онлайн: https://riptutorial.com/ru/android/topic/2812/виджеты

глава 118: волчок

Examples

Добавление счетчика в вашу деятельность

В /res/values/strings.xml:

```
<string-array name="spinner_options">
    <item>Option 1</item>
    <item>Option 2</item>
    <item>Option 3</item>
</string-array>
```

В макете XML:

```
<Spinner
    android:id="@+id/spinnerName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_options" />
```

В работе:

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

Пример основного Spinner

Spinner Это тип выпадающего ввода. Во-первых, в макете

```
  <Spinner
    android:id="@+id/spinner"     <!-- id to refer this spinner from JAVA-->
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

</Spinner>
```

Теперь во-вторых, заполнять значения в прядильнике. Есть два способа заполнения значений в `spinner` .

1. Из самого XML создайте **array.xml** в каталоге **значений** под **res** . Создайте этот `array`

```
  <string-array name="defaultValue">
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
</string-array>
```

Теперь добавьте эту строку в sppiner XML

```
            android:entries="@array/defaultValue"
```

2. Вы также можете добавлять значения через JAVA

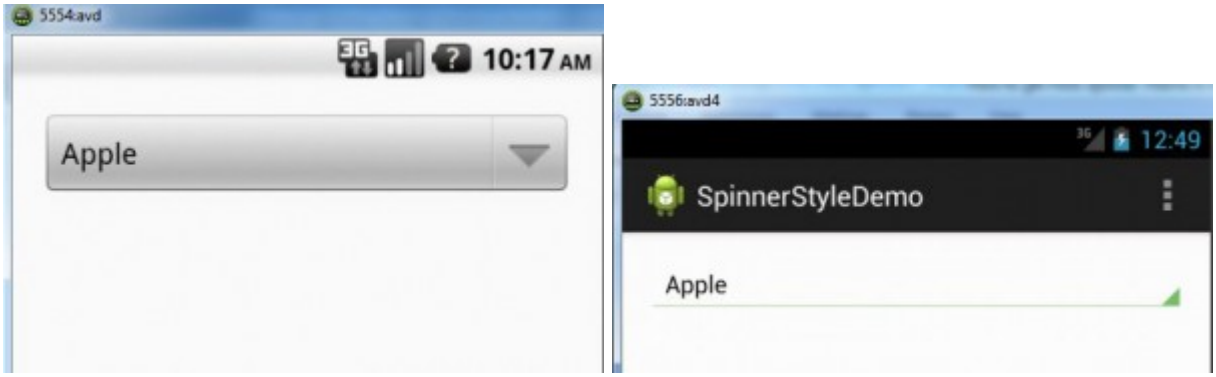
если вы используете в `activity` cityArea = (Spinner) findViewById (R.id.cityArea); иначе, если вы используете `fragment`

```
cityArea = (Spinner) findViewById(R.id.cityArea);
```

Теперь создадим `arrayList` of `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));
```

Это будет выглядеть



По версии Android-устройства устройство будет отображать стиль

Ниже приведены некоторые темы по умолчанию

Если приложение явно не запрашивает тему в своем манифесте, система Android определит тему по умолчанию, основанную на целевом целевом приложении appdkVersion, чтобы поддерживать первоначальные ожидания приложения:

Версия Android SDK	Тема по умолчанию
Версия <11	@android: стиль / Стиль
Версия от 11 до 13	@android: стиль / Theme.Holo

Версия Android SDK	Тема по умолчанию
14 и выше	@android: стиль / Theme.DeviceDefault

`Spinner` можно легко настроить с помощью xml, например

```
 android:background="@drawable/spinner_background"

 android:layout_margin="16dp"

 android:padding="16dp"
```

Создайте собственный фон в XML и используйте его.

легко получить позицию и другие детали выбранного элемента в прядильщике

```
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) {

        }
    });
```

Изменение цвета текста выбранного элемента в прядильщике

Это можно сделать двумя способами в XML

```
<item android:state_activated="true" android:color="@color/red"/>
```

Это изменит выбранный цвет элемента во всплывающем окне.

и от JAVA сделайте это (в setOnItemSelectedListener (...))

```
 @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long
id) {
                 ((TextView) parent.getChildAt(0)).setTextColor(0x00000000);
// similarly change `background color` etc.
            }
```

Прочитайте волчок онлайн: https://riptutorial.com/ru/android/topic/3459/волчок

глава 119: Время Утилиты

Examples

Преобразование формата даты в миллисекунды

Чтобы преобразовать дату в формате dd / MM / yyyy в миллисекунды, вы вызываете эту функцию с данными как 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();
}
```

Этот метод преобразует миллисекунды в дату даты:

```
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;
}
```

Этот метод преобразует данные в определенный день, месяц и год в миллисекунды. Это будет очень `Timpicker` при использовании `Timpicker` или `Datepicker`

```
public static long getTimeInMillis(int day, int month, int year) {
    Calendar calendar = Calendar.getInstance();
    calendar.set(year, month, day);
    return calendar.getTimeInMillis();
}
```

Он вернет миллисекунды с даты

```
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;
}
```

Он вернет текущую дату

```
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;
}
```

Примечание. Java. Обеспечивает поддержку формата даты.

Проверить в течение периода

Этот пример поможет проверить данное время в течение определенного периода или нет.

Чтобы проверить время на сегодняшний день, мы можем использовать класс ***DateUtils***

```
boolean isToday = DateUtils.isToday(timeInMillis);
```

Чтобы проверить время в течение недели,

```
private static boolean isWithinWeek(final long millis) {
    return System.currentTimeMillis() - millis <= (DateUtils.WEEK_IN_MILLIS -
DateUtils.DAY_IN_MILLIS);
}
```

Чтобы проверить время в течение года,

```
private static boolean isWithinYear(final long millis) {
    return System.currentTimeMillis() - millis <= DateUtils.YEAR_IN_MILLIS;
}
```

Чтобы проверить время, в течение дня, включая день,

```
public static boolean isWithinDay(long timeInMillis, int day) {
    long diff = System.currentTimeMillis() - timeInMillis;

    float dayCount = (float) (diff / DateUtils.DAY_IN_MILLIS);

    return dayCount < day;
}
```

Примечание. DateUtils - это **android.text.format.DateUtils**

GetCurrentRealTime

Это вычисляет текущее время устройства и добавляет / вычитает разницу между реальным и временем устройства

```
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;
}
```

получить часовой пояс, основанный на UTC.

```
public static TimeZone getUTCTimeZone() {
    return TimeZone.getTimeZone("GMT");
}
```

Прочитайте Время Утилиты онлайн: https://riptutorial.com/ru/android/topic/7138/время-утилиты

глава 120: Выбор даты и времени

Examples

Материал DatePicker

добавьте ниже зависимости для `build.gradle` в разделе зависимости. (это неофициальная библиотека для выбора даты)

```
  compile 'com.wdullaer:materialdatetimepicker:2.3.0'
```

Теперь нам нужно открыть `DatePicker` на событие click Button.

Поэтому создайте одну кнопку на xml-файле, как показано ниже.

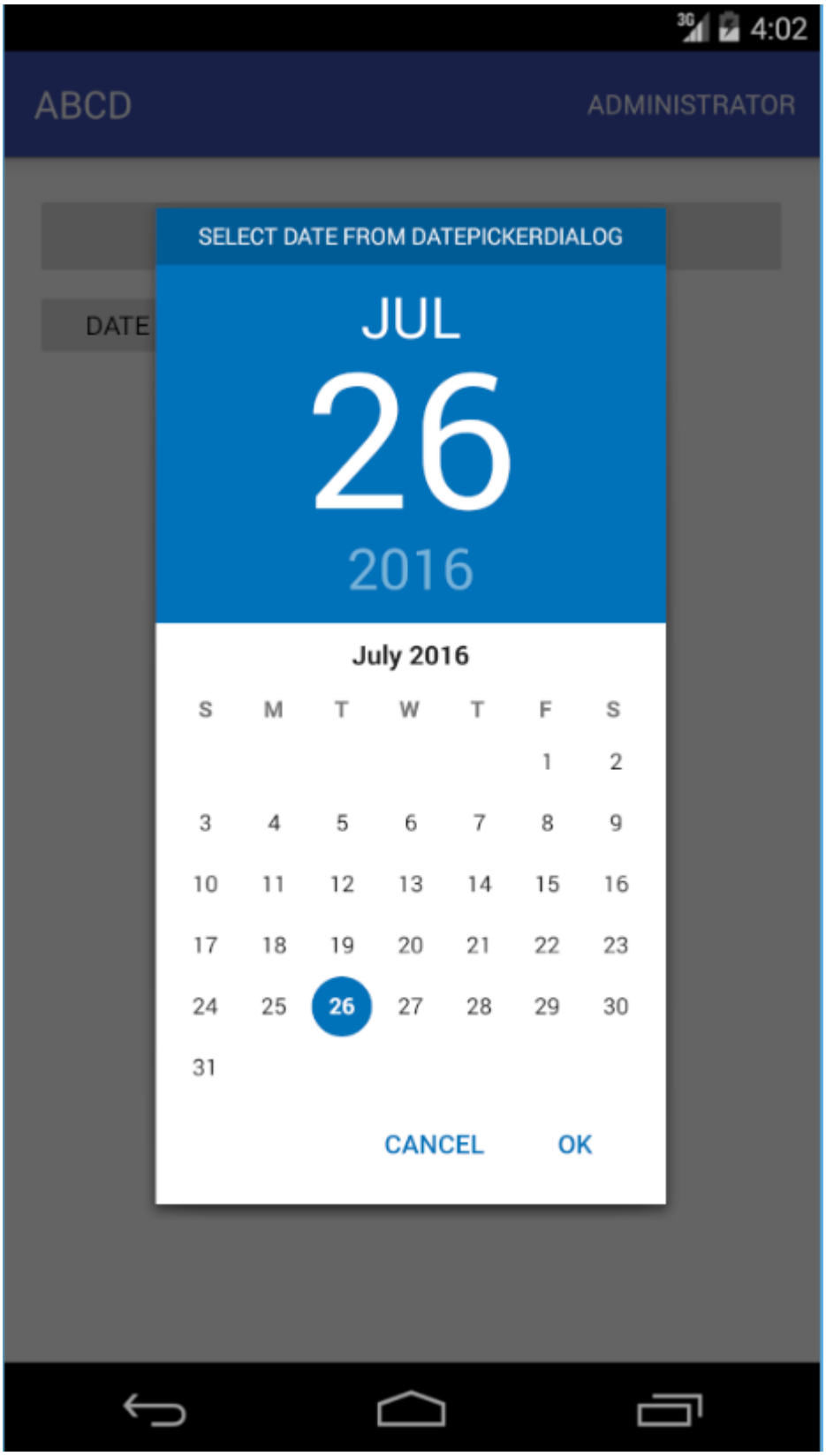
```
     <Button
            android:id="@+id/dialog_bt_date"
            android:layout_below="@+id/resetButton"
            android:layout_width="wrap_content"
            android:layout_height="40dp"
            android:textColor="#FF000000"
            android:gravity="center"
            android:text="DATE"/>
```

и в MainActivity используйте этот способ.

```
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;
    }

}
```

Выход :



Диалог выбора даты

Это диалоговое окно, в котором пользователю предлагается выбрать дату с помощью `DatePicker` . Диалог требует контекста, начального года, месяца и дня, чтобы показать диалог со стартовой датой. Когда пользователь выбирает дату, она `DatePickerDialog.OnDateSetListener` обратные вызовы через `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();

    }
```

Обратите внимание, что месяц - это int, начинающийся с 0 для января до 11 декабря

Прочитайте Выбор даты и времени онлайн: https://riptutorial.com/ru/android/topic/2836/выбор-даты-и-времени

глава 121: Декорации RecyclerView

Синтаксис

- RecyclerView addItemDecoration (RecyclerView.ItemDecoration)
- RecyclerView addItemDecoration (RecyclerView.ItemDecoration decoration, int index)

параметры

параметр	подробности
украшение	украшение элемента для добавления в `RecyclerView`
индекс	индекс в списке украшений для этого `RecyclerView` . Это порядок, в котором `getItemOffset` и `onDraw` . Более поздние вызовы могут перегружать предыдущие.

замечания

Украшения являются статическими

Поскольку декорации только рисуются, невозможно добавить к ним зрителей кликов или другие функции пользовательского интерфейса.

Многократные украшения

Добавление нескольких украшений в `RecyclerView` будет работать в некоторых случаях, но в настоящее время нет общедоступного API для учета других возможных украшений при измерении или рисовании. Вы можете получить границы обзора или границы, украшенные просмотром, где декорированные границы являются суммой всех применений декоративных смещений.

Другие связанные темы:

RecyclerView
RecyclerView onClickListeners

Официальный javadoc

https://developer.android.com/reference/android/support/v7/widget/RecyclerView.ItemDecoration.html

Examples

Рисование разделителя

Это будет рисовать линию внизу каждого представления, но последнее будет действовать как разделитель между элементами.

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

Поля на товар с ItemDecoration

Вы можете использовать `RecyclerView.ItemDecoration` для добавления дополнительных полей вокруг каждого элемента в RecyclerView. Это может в некоторых случаях очистить как реализацию адаптера, так и XML-представление вашего элемента.

```
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);
    }
}
```

Чтобы включить оформление, просто добавьте его в свой RecyclerView:

```
// in your onCreate()
RecyclerView rv = (RecyclerView) findItemById(R.id.myList);
rv.addItemDecoration(new MyItemDecoration(context));
```

Добавить разделитель в RecyclerView

Прежде всего вам нужно создать класс, который расширяет `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);
    }
}

}
```

Добавьте `divider_blue.xml` в свою папку:

```
<?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>
```

Затем используйте его так:

```
recyclerView.addItemDecoration(new SimpleBlueDivider(context));
```

Результат будет таким:


28% 14:20
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)

Это изображение является лишь примером того, как работают разделители, если вы хотите следовать спецификациям Material Design при добавлении разделителей, пожалуйста, взгляните на эту ссылку: разделители и спасибо @Brenden Kromhout , предоставив ссылку.

Как добавить разделители и DividerItemDecoration

`DividerItemDecoration` - это `RecyclerView.ItemDecoration` которая может использоваться как разделитель между элементами.

```
DividerItemDecoration mDividerItemDecoration = new DividerItemDecoration(context,
             mLayoutManager.getOrientation());
recyclerView.addItemDecoration(mDividerItemDecoration);
```

Он поддерживает обе ориентации с помощью `DividerItemDecoration.VERTICAL` и `DividerItemDecoration.HORIZONTAL` .

ItemOffsetDecoration для GridLayoutManager в RecycleView

Следующий пример поможет предоставить равное пространство элементу в 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;
    }
}
```

Вы можете вызвать ItemDecoration, как показано ниже.

```
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);
```

и смещение позиции элемента

```
<dimen name="item_offset">5dp</dimen>
```

Прочитайте Декорации RecyclerView онлайн: https://riptutorial.com/ru/android/topic/506/декорации-recyclerview

глава 122: Деятельность

Вступление

Действие представляет собой один экран с пользовательским **интерфейсом (UI)** . Например, приложение для Android может иметь более одного действия. Приложение электронной почты может иметь одно действие для отображения всех электронных писем, другого действия для отображения содержимого электронной почты, а также другого действия для составления нового сообщения электронной почты. Все действия в приложении работают вместе, чтобы создать идеальный пользовательский интерфейс.

Синтаксис

- void onCreate (Bundle savedInstanceState) // Вызывается, когда начинается действие.
- void onPostCreate (Bundle savedInstanceState) // Вызывается, когда завершен запуск активности (после того, как были вызваны onStart () и onRestoreInstanceState (Bundle)).
- void onStart () // Вызывается после onCreate (Bundle) - или после onRestart (), когда действие было остановлено, но теперь снова отображается пользователю.
- void onResume () // Вызывается после onRestoreInstanceState (Bundle), onRestart () или onPause (), чтобы ваша активность начала взаимодействовать с пользователем.
- void onPostResume () // Вызывается, когда завершено возобновление активности (после вызова функции onResume ()).
- void onRestart () // Вызывается после onStop (), когда текущая активность повторно отображается пользователю (пользователь перешел обратно к ней).
- void onPause () // Вызывается как часть жизненного цикла активности, когда активность перемещается в фоновый режим, но еще не убита.
- void onStop () // Вызывается, когда пользователь больше не отображается.
- void onDestroy () // Выполнять окончательную очистку до того, как действие будет уничтожено.
- void onNewIntent (намерение намерения) // Это вызвано для действий, которые устанавливают launchMode на «singleTop» в их пакете, или если клиент использовал флаг FLAG_ACTIVITY_SINGLE_TOP при вызове startActivity (Intent).
- void onSaveInstanceState (Bundle outState) // Вызывается для извлечения состояния каждого экземпляра из активности перед тем, как его убить, чтобы состояние можно

было восстановить в onCreate (Bundle) или onRestoreInstanceState (Bundle) (набор, заполненный этим методом, будет передан как для).

- void onRestoreInstanceState (Bundle savedInstanceState) // Этот метод вызывается после onStart (), когда активность повторно инициализируется из ранее сохраненного состояния, указанного здесь в файле savedInstanceState.

параметры

параметр	подробности
умысел	Может использоваться с startActivity для запуска Activity
сверток	Отображение из строковых ключей на различные значения Parcelable .
контекст	Интерфейс к глобальной информации о среде приложения.

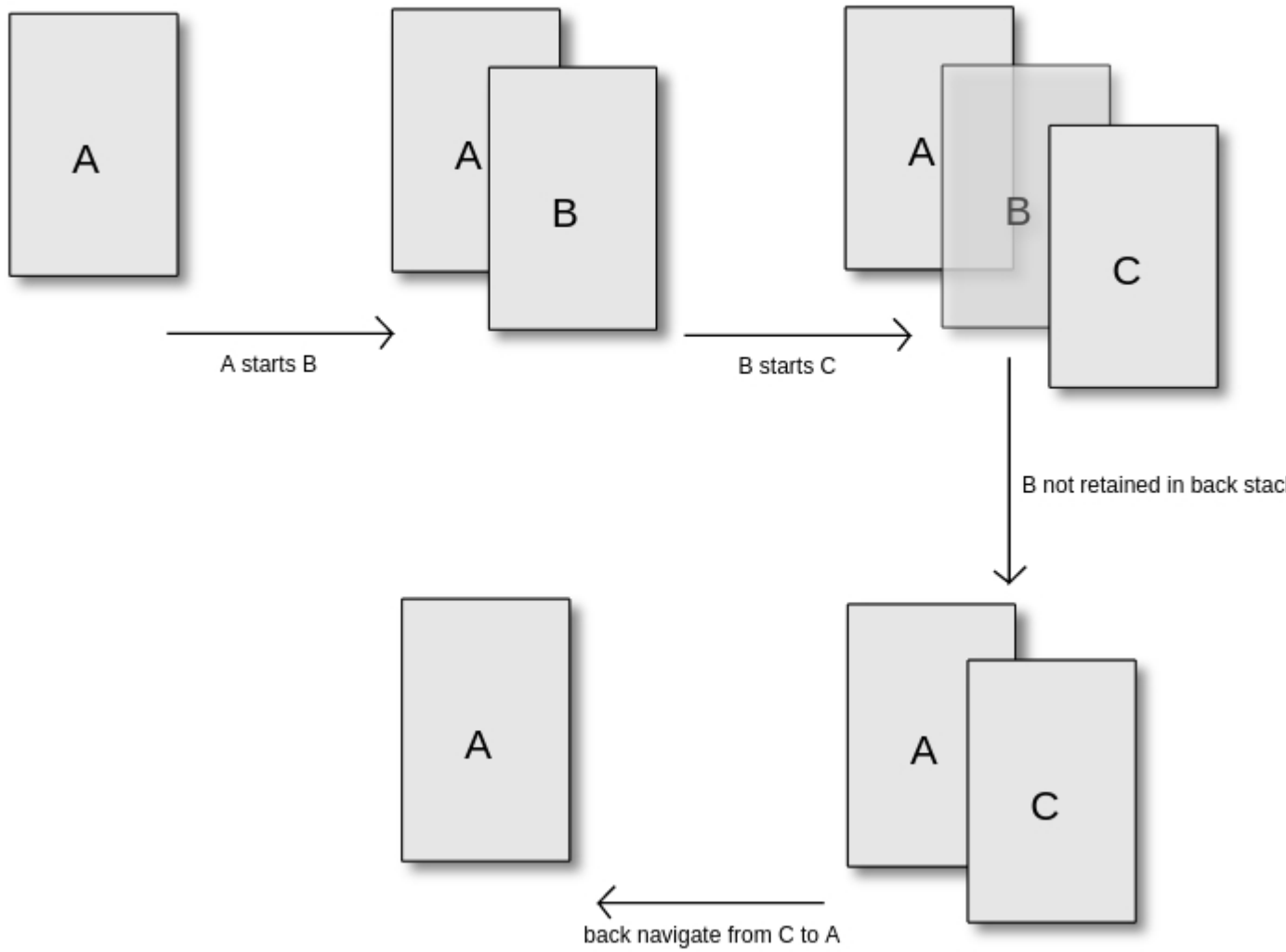
замечания

Activity - это компонент приложения, который предоставляет экран, с помощью которого пользователи могут взаимодействовать, чтобы что-то сделать, например, набрать телефон, сделать фотографию, отправить электронное письмо или просмотреть карту. Каждому действию присваивается окно, в котором можно нарисовать свой пользовательский интерфейс. Окно обычно заполняет экран, но может быть меньше экрана и плавать поверх других окон.

Examples

Исключить действие из истории back-stack

Пусть есть активность `B` которая может быть открыта, и может начать больше действий. Но пользователь не должен сталкиваться с этим при навигации в задачах задачи.



Самое простое решение - установить для атрибута `noHistory` значение `true` для этого тэга `<activity>` в `AndroidManifest.xml` :

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

Такое же поведение также возможно из кода, если `B` вызывает `finish()` перед началом следующего действия:

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

Типичное использование флага `noHistory` - «Splash Screen» или «Действия входа».

Объяснение жизненного цикла Android

Предположим, приложение с MainActivity, которое может вызвать следующую активность,

нажав кнопку.

```
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);
    } }
```

а также

```
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");
    } }
```

Когда приложение создано впервые
D / MainActivity: вызов onCreate из MainActivity
D / MainActivity: вызов onStart из MainActivity
D / MainActivity: вызов onResume из MainActivity
называются

Когда экран спит
08: 11: 03.142 D / MainActivity: вызов onPause из MainActivity
08: 11: 03.192 D / MainActivity: вызов onStop из MainActivity
называются. И снова, когда он просыпается
08: 11: 55.922 D / MainActivity: вызов onRestart из MainActivity
08: 11: 55.962 D / MainActivity: вызов onStart из MainActivity
08: 11: 55.962 D / MainActivity: вызов onResume из MainActivity
называются

Случай 1: когда из основной деятельности ***вызывается*** следующая активность
D / MainActivity: вызов следующей операции

D / MainActivity: вызов onPause из MainActivity
D / NextActivity: вызов onCreate из следующей активности
D / NextActivity: вызов onStart из следующей активности
D / NextActivity: вызов onResume из следующей активности
D / MainActivity: вызов onStop из MainActivity

Возврат к основной активности из следующего действия с помощью кнопки «Назад»
D / NextActivity: вызов onPause из следующей активности
D / MainActivity: вызов onRestart из MainActivity
D / MainActivity: вызов onStart из MainActivity
D / MainActivity: вызов onResume из MainActivity
D / NextActivity: вызов onStop из следующей активности
D / NextActivity: вызов onDestroy из следующей активности

Случай 2: когда активность частично скрыта (когда нажата кнопка обзора) или Когда приложение переходит в фоновый режим, а другое приложение полностью скрывает его
D / MainActivity: вызов onPause из MainActivity
D / MainActivity: вызов onStop из MainActivity
и когда приложение вернется на передний план, готовый принять пользовательские входы,
D / MainActivity: вызов onRestart из MainActivity
D / MainActivity: вызов onStart из MainActivity
D / MainActivity: вызов onResume из MainActivity
называются

Случай 3: когда действие вызывается для выполнения неявного намерения, и пользователь делает выбор. Например, когда нажата кнопка совместного доступа, и пользователь должен выбрать приложение из списка показанных приложений
D / MainActivity: вызов onPause из MainActivity

Активность видна, но не активна. Когда выбор сделан и приложение активно
D / MainActivity: вызов onResume из MainActivity
называется

Case4:
Когда приложение будет убито в фоновом режиме (чтобы освободить ресурсы для другого приложения переднего плана), *onPause* (для устройства с предварительной *сотой*) или *onStop* (для сотового устройства) будет последним, которое будет *вызвано* до того, как приложение будет завершено.

onCreate и onDestroy будут называться предельно один раз при каждом запуске приложения. Но onPause, onStop, onRestart, onStart, onResume может быть вызван много раз в течение жизненного цикла.

Действие launchMode

Режим запуска определяет поведение новой или существующей активности в задаче. Возможны следующие режимы запуска:

- стандарт
- SingleTop
- singleTask
- SingleInstance

Он должен быть определен в android в элементе `<activity/>` как атрибут `android:launchMode` .

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

Стандарт:

Значение по умолчанию. Если этот режим установлен, новая активность всегда будет создаваться для каждого нового намерения. Таким образом, можно получить много видов деятельности одного типа. Новая деятельность будет размещена в верхней части задачи. Существует некоторая разница для разных версий Android: если активность начинается с другого приложения, в андроидах <= 4.4 он будет помещен в ту же задачу, что и стартерное приложение, но при> = 5.0 будет создано новое задание.

SingleTop:

Этот режим практически не отличается от `standard` . Можно создать много экземпляров активности singleTop. Разница заключается в том, что если экземпляр активности уже существует в верхней части текущего стека, вместо нового экземпляра будет вызываться `onNewIntent()` .

SingleTask:

Активность в этом режиме запуска может иметь только один экземпляр **в системе** . Будет создана новая задача для активности, если она не существует. В противном случае задача с активностью будет перемещена вперёд и `onNewIntent` .

SingleInstance:

Этот режим похож на `singleTask` . Разница - это задача, которая имеет активность с `singleInstance` может иметь только эту активность и не более того. Когда `singleInstance` activity создает другое действие, будет создана новая задача для размещения этой активности.

Представление пользовательского интерфейса с помощью setContentView

Класс Activity заботится о создании окна для вас, в котором вы можете разместить свой пользовательский интерфейс с помощью `setContentView` .
Существует три метода `setContentView` :

- `setContentView(int layoutResID)` - установка содержимого активности из ресурса макета.
- `setContentView(View view)` - Установить содержимое активности в явное представление.
- `setContentView(View view, ViewGroup.LayoutParams params)` - Установите содержимое активности в явное представление с предоставленными параметрами.

Когда `setContentView` , это представление помещается непосредственно в иерархию представлений активности. Он может быть сложной иерархией представлений.

Примеры

Установить содержимое из файла ресурсов:

Добавьте файл ресурсов (main.xml в этом примере) с иерархией представлений:

```
<?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>
```

Задайте его как содержимое в действии:

```
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);
    }
}
```

Задайте содержимое для явного представления:

```
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);
    }
}
```

Очистите текущий стек активности и запустите новую операцию

Если вы хотите очистить текущий стек активности и запустить новое действие (например, выход из приложения и запуск журнала в Activity), существует, по-видимому, два подхода.

1. Цель (API> = 16)

Вызов `finishAffinity()` из Activity

2. Цель (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();
```

Завершить приложение с исключением из ректоратов

Сначала определите ExitActivity в AndroidManifest.xml

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

После этого класс 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);
    }
}
```

Навигация по действиям

Навигация вверх осуществляется в android, добавив `android:parentActivityName=""` в Manifest.xml к тегу активности. В основном с помощью этого тега вы сообщаете системе о родительской активности.

Как это делается?

```
<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>
```

Теперь, когда я нажму на стрелку внутри панели инструментов HomeActivity, она вернет меня к родительской активности.

Код Java

Здесь я напишу соответствующий Java-код, необходимый для этой функции.

```
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.
}
}
```

Если вы запустите этот код, вы увидите, когда вы нажмете кнопку «Назад», он вернет вас в MainActivity. Для более глубокого понимания Up Navigation я бы рекомендовал читать документы

Вы можете более точно настроить это поведение по своим потребностям, переопределив

```
@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);
}
```

Простой взлом

Это простой хак, который в основном используется для перехода к родительской активности, если родитель находится в backstack. `onBackPressed()` если id равен `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);
    }
```

Прочитайте Деятельность онлайн: https://riptutorial.com/ru/android/topic/1481/деятельность

глава 123: Джексон

Вступление

Джексон - многоцелевая библиотека Java для обработки JSON. Джексон стремится быть наилучшим сочетанием быстрого, правильного, легкого и эргономичного для разработчиков.

Особенности Джексона:

Режим многопроцессорной обработки и очень хорошее сотрудничество

Не только аннотации, но и смешанные аннотации

Полностью поддерживайте общие типы

Поддержка полиморфных типов

Examples

Полный пример привязки данных

Данные JSON

```
{
  "name" : { "first" : "Joe", "last" : "Sixpack" },
  "gender" : "MALE",
  "verified" : false,
  "userImage" : "keliuyue"
}
```

Для превращения его в экземпляр пользователя требуется две строки Java:

```
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;
    }
}
```

Маршаллинг назад к JSON аналогичен прямому:

```
mapper.writeValue(new File("user-modified.json"), user);
```

Прочитайте Джексон онлайн: https://riptutorial.com/ru/android/topic/10878/джексон

глава 124: диалог

параметры

Линия	Описание
шоу();	Показывает диалог
setContentView (R.layout.yourlayout);	устанавливает ContentView диалогового окна в ваш собственный макет.
отклонить ()	Закрывает диалог

замечания

- Диалогу в первом примере (Dialog) не нужно вызывать `show()` когда он создается, поскольку он обрабатывается в конструкторе
- Диалоги оповещений должны быть построены через новый экземпляр класса `AlertDialog.Builder()` . Следуя шаблону Builder , все члены AlertDialog.Builder могут быть привязаны к методу, чтобы «создать» экземпляр диалога.
- Конструктор Alert Dialog может напрямую `show()` диалог - вам не нужно вызывать `create()` затем `show()` в экземпляре AlertDialog

Examples

Диалоговое окно оповещений

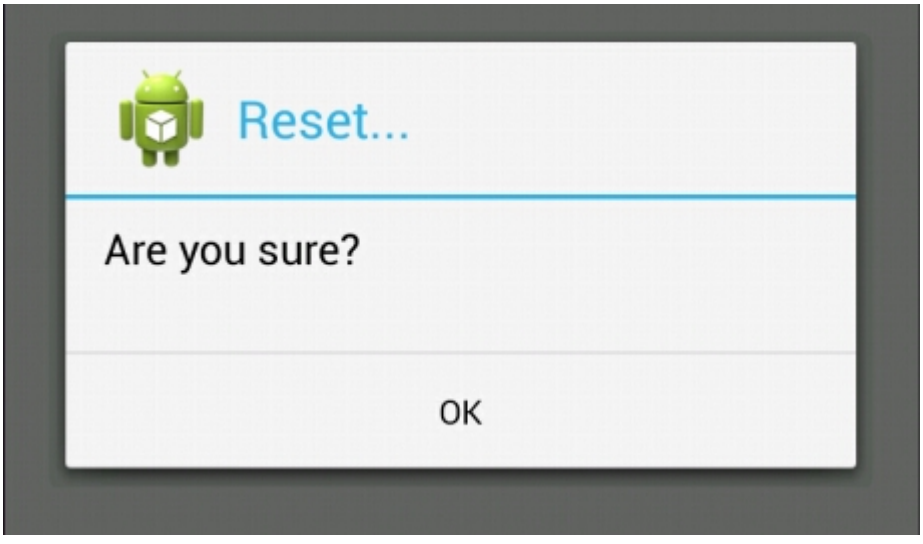
```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
                MainActivity.this);

        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();
```

Базовый диалог оповещений

```
    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.
```

Теперь этот код достигнет этого:



(*Источник изображения: WikiHow*)

Выбор даты в DialogFragment

xml диалога:

```
<?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>
```

Диалоговый класс:

```
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;
    }

}
```

Активность вызова диалога:

```
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` - это самый простой способ использования `DatePicker` , потому что вы можете показывать диалог в любом месте приложения. Вам не нужно реализовывать свой собственный макет с `DatePicker` виджета `DatePicker` .

Как показать диалог:

```
DatePickerDialog datePickerDialog = new DatePickerDialog(context, listener, year, month, day);
datePickerDialog.show();
```

Вы можете получить виджет `DataPicker` из диалогового окна выше, чтобы получить доступ к большему количеству функций и, например, установить минимальную дату в миллисекундах:

```
DatePicker datePicker = datePickerDialog.getDatePicker();
datePicker.setMinDate(System.currentTimeMillis());
```

DatePicker

`DatePicker` позволяет пользователю выбрать дату. Когда мы создаем новый экземпляр `DatePicker` , мы можем установить начальную дату. Если мы не укажем начальную дату, текущая дата будет установлена по умолчанию.

Мы можем показать `DatePicker` пользователю с помощью `DatePickerDialog` или путем создания нашего собственного макета с `DatePicker` виджета `DatePicker` .

Также мы можем ограничить диапазон дат, который пользователь может выбрать.

Установив минимальную дату в миллисекундах

```
//In this case user can pick date only from future
datePicker.setMinDate(System.currentTimeMillis());
```

Установив максимальную дату в миллисекундах

```
//In this case user can pick date only, before following week.
datePicker.setMaxDate(System.currentTimeMillis() + TimeUnit.DAYS.toMillis(7));
```

Чтобы получать информацию о том, какая дата была выбрана пользователем, мы должны использовать `Listener` .

Если мы используем `DatePickerDialog` , мы можем установить `OnDateSetListener` в конструкторе, когда мы создаем новый экземпляр `DatePickerDialog` :

Пример использования `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) {

    }
}
```

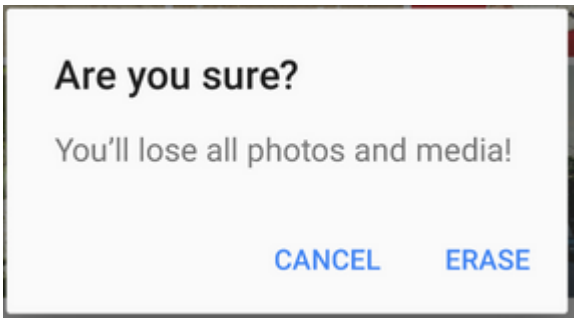
В противном случае, если мы создаем собственный макет с `DatePicker` , нам также нужно создать собственный слушатель, как показано в другом примере

Добавление дизайна материала AlertDialog в приложение с помощью Appcompat

`AlertDialog` - это подкласс `Dialog` который может отображать одну, две или три кнопки. Если вы хотите отобразить строку в этом диалоговом окне, используйте метод `setMessage()` .

Пакет `AlertDialog` из `android.app` отображается по-разному в разных версиях ОС Android.

Библиотека Android V7 Appcompat предоставляет реализацию `AlertDialog` которая будет отображаться с помощью Material Design на всех поддерживаемых версиях ОС Android, как показано ниже:



Сначала вам нужно добавить библиотеку V7 Appcompat в свой проект. вы можете сделать это в файле build.gradle на уровне приложения:

```
dependencies {
    compile 'com.android.support:appcompat-v7:24.2.1'
    //........
}
```

Обязательно импортируйте правильный класс:

```
import android.support.v7.app.AlertDialog;
```

Затем создайте AlertDialog следующим образом:

```
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 в AlertDialog

Мы всегда можем использовать `ListView` или `RecyclerView` для выбора из списка элементов, но если у нас есть небольшое количество вариантов и среди этих вариантов мы хотим, чтобы пользователь `AlertDialog.Builder setAdapter` один, мы можем использовать `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();
    }
```

Возможно, если нам не нужен какой-либо конкретный `ListView` , мы можем использовать базовый способ:

```
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();
```

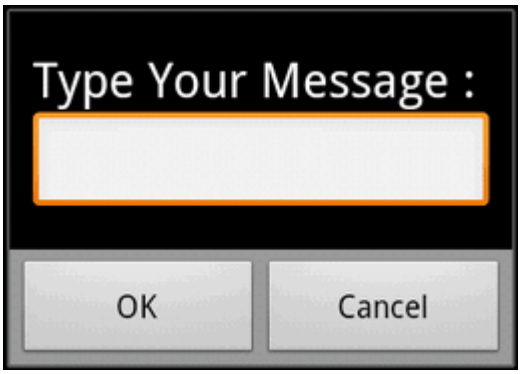
Пользовательский диалог оповещений с помощью EditText

```
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();
    }
```

Файл 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>
```



Полноэкранный пользовательский диалог без фона и без заголовка

в `styles.xml` добавьте свой собственный стиль:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppBaseTheme" parent="@android:style/Theme.Light.NoTitleBar.Fullscreen">
    </style>
</resources>
```

Создайте свой собственный макет для диалога: `fullscreen.xml` :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

</RelativeLayout>
```

Затем в java-файле вы можете использовать его для Activity или Dialog и т. Д .:

```
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();
    }
}
```

Диалоговое окно оповещений с многострочным заголовком

Метод setCustomTitle () для AlertDialog.Builder позволяет указать произвольное представление, которое будет использоваться для заголовка диалога. Одним из распространенных способов использования этого метода является создание диалогового окна предупреждения с длинным заголовком.

```
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();
```

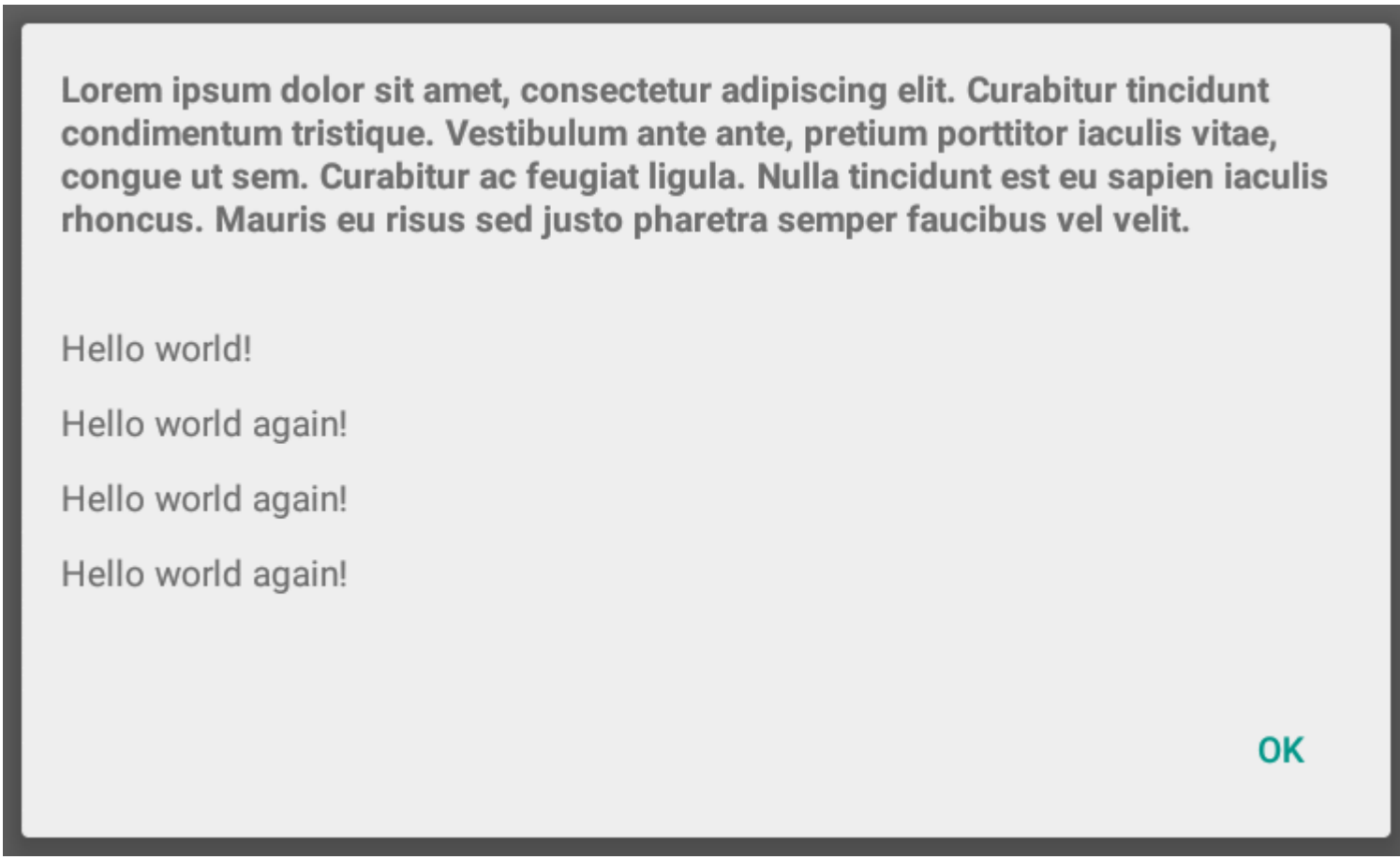
my_dialog_title.xml:

```
<?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>
```



Прочитайте диалог онлайн: https://riptutorial.com/ru/android/topic/1225/диалог

глава 125: Дизайн материалов

Вступление

Material Design - это всестороннее руководство по визуальному дизайну, дизайну движения и взаимодействия между платформами и устройствами.

замечания

Также см. Оригинальную запись в блоге Android, представляющую библиотеку поддержки дизайна

Официальная документация

https://developer.android.com/design/material/index.html

Руководство по проектированию материалов

https://material.io/guidelines

Другие проектные ресурсы и библиотеки

https://design.google.com/resources/

Examples

Применение темы AppCompat

Библиотека поддержки AppCompat предоставляет темы для создания приложений с использованием спецификации Material Design . Тема с родителем `Theme.AppCompat` также требуется для действия для расширения `AppCompatActivity` .

Первым шагом является настройка цветовой палитры вашей темы, чтобы автоматически раскрасить ваше приложение.
В `res/styles.xml` вашего приложения вы можете определить:

```
<!-- 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>
```

Вместо `Theme.AppCompat` , который имеет темный фон, вы также можете использовать `Theme.AppCompat.Light` или `Theme.AppCompat.Light.DarkActionBar` .

Вы можете настроить тему своими собственными цветами. Хороший выбор в цветовой карте спецификации дизайна материалов и в палитре материалов . Цвета «500» - хороший выбор для первичного (синий пример 500); выберите «700» того же оттенка для темного; и оттенок от другого оттенка, как цвет акцента. Основной цвет используется для панели инструментов вашего приложения и его записи на экране обзора (последние приложения), более темного варианта для оттенка строки состояния и цвета акцента для выделения некоторых элементов управления.

Создав эту тему, примените ее к своему приложению в `AndroidManifest.xml` а также примените тему к любому конкретному действию. Это полезно для применения темы `AppTheme.NoActionBar` , которая позволяет реализовывать конфигурации панели инструментов, отличные от настроек по умолчанию.

```
<application android:theme="@style/AppTheme"
    ...>
    <activity
        android:name=".MainActivity"
        android:theme="@style/AppTheme" />
</application>
```

Вы также можете применять темы к отдельным представлениям с помощью `android:theme` и `ThemeOverlay` . Например, с помощью `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" />
```

или `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>
```

Добавление панели инструментов

`Toolbar` представляет собой обобщение `ActionBar` для использования в макетах приложений. Хотя `ActionBar` традиционно является частью непрозрачного декоративного окна `Activity's` контролируемого каркасом, `Toolbar` может быть размещена на любом произвольном уровне вложенности внутри иерархии представлений. Его можно добавить, выполнив следующие шаги:

1. Убедитесь, что следующая зависимость добавлена в файл **build.gradle** вашего модуля (например, app) в зависимостях:

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

2. Задайте тему для вашего приложения тем, у кого **нет** `ActionBar` . Для этого отредактируйте файл **styles.xml** под `res/values` и установите тему `Theme.AppCompat` . В этом примере мы используем `Theme.AppCompat.NoActionBar` качестве родителя вашего `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>
```

   Вы также можете использовать `Theme.AppCompat.Light.NoActionBar` или `Theme.AppCompat.DayNight.NoActionBar` или любую другую тему, которая по своей сути не имеет `ActionBar`

3. Добавьте `Toolbar` в макет своей деятельности:

```
<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"/>
```

   Ниже `Toolbar` вы можете добавить остальную часть вашего макета.

4. В своей `Activity` установите `Toolbar` в качестве `ActionBar` для этого `Activity` . Если вы используете библиотеку *appcompat* и `AppCompatActivity` , вы должны использовать метод `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);

    //...
```

```
}
```

После выполнения вышеуказанных шагов вы можете использовать метод `getSupportActionBar()` для управления `Toolbar` которая задана как `ActionBar` .

Например, вы можете установить заголовок, как показано ниже:

```
getSupportActionBar().setTitle("Activity Title");
```

Например, вы также можете установить цвет заголовка и фона, как показано ниже:

```
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)));
```

Добавление FloatingActionButton (FAB)

В дизайне материала кнопка Floating action представляет собой основное действие в Activity.
Они отличаются круговым значком, плавающим над пользовательским интерфейсом, и имеют поведение движения, которое включает в себя морфинг, запуск и передающую опорную точку.

Убедитесь, что в файл build.gradle вашего приложения добавлена следующая зависимость:

```
compile 'com.android.support:design:25.3.1'
```

Теперь добавьте `FloatingActionButton` в ваш файл макета:

```
<android.support.design.widget.FloatingActionButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_margin="16dp"
      android:src="@drawable/some_icon"/>
```

где атрибут `src` ссылается на значок, который должен использоваться для плавающего действия.
Результат должен выглядеть примерно так (предполагая, что ваш цвет акцента - Material Pink):

По умолчанию цвет фона вашего `FloatingActionButton` будет установлен на цвет акцента вашей темы. Кроме того, обратите внимание, что `FloatingActionButton` требует, чтобы маржа вокруг него работала должным образом. Рекомендуемая маржа для дна - `16dp` для телефонов и `24dp` для планшетов.

Вот свойства, которые вы можете использовать для дальнейшей настройки `FloatingActionButton` (при условии, что `xmlns:app="http://schemas.android.com/apk/res-auto` объявляется как пространство имен верхней частью вашего макета):

- **`app:fabSize`** : может быть установлено в `normal` или `mini` для переключения между обычной или меньшей версией.
- **`app:rippleColor`** : Устанавливает цвет эффекта пульсации вашего `FloatingActionButton` . Может быть цветным или шестнадцатеричным.
- **`app:elevation`** : может быть строкой, целым числом, логическим значением, значением цвета, с плавающей точкой, значением измерения.
- **`app:useCompatPadding`** : включить совместимость. Может быть, логическое значение, например `true` или `false` . Установите значение `true` чтобы использовать дополнение сопоставления на api-21 и позже, чтобы поддерживать последовательный внешний вид со старыми уровнями api.

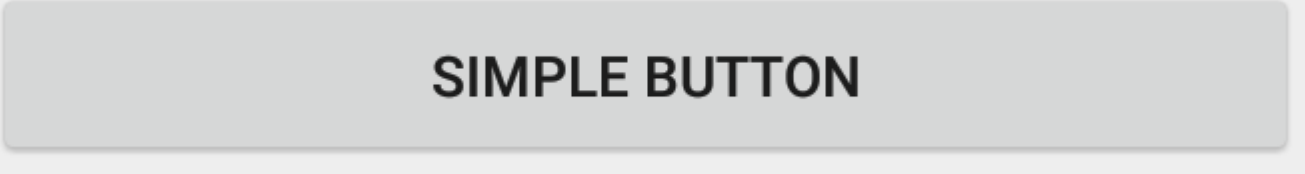
Вы можете найти больше примеров о FAB здесь .

Кнопки в стиле Material Design

Библиотека поддержки AppCompat определяет несколько полезных стилей для кнопок , каждая из которых расширяет базовый стиль `Widget.AppCompat.Button` который применяется ко всем кнопкам по умолчанию, если вы используете тему `AppCompat` . Этот стиль помогает гарантировать, что все кнопки выглядят одинаково по умолчанию в соответствии со спецификацией Material Design .

В этом случае цвет акцента розовый.

1. Простая кнопка: `@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. Цветная кнопка: `@style/Widget.AppCompat.Button.Colored`
   Стиль `Widget.AppCompat.Button.Colored` расширяет стиль `Widget.AppCompat.Button` и автоматически применяет **цвет акцента, который** вы выбрали в своей теме приложения.


```
<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"/>
```

   Если вы хотите настроить цвет фона без изменения цвета акцента в *основной теме,* вы можете создать *собственную тему* (расширяющую тему `ThemeOverlay`) для своей `Button` и присвоить ее атрибуту `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"/>
```

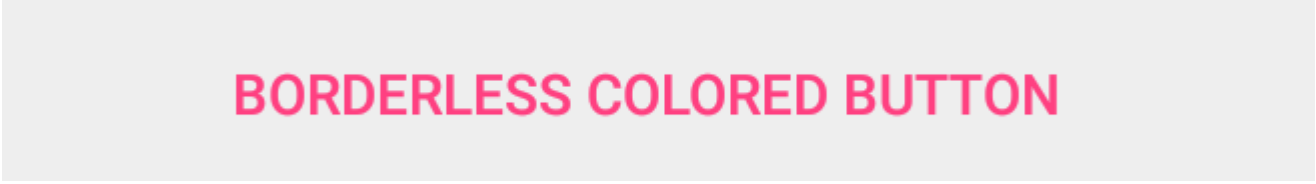
   Определите тему в `res/values/themes.xml` :

```
 <style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
      <item name="colorAccent">@color/my_color</item>
 </style>
```

3. Кнопка Borderless: `@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. Без полей Цветная кнопка: `@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"/>
```

Как использовать TextInputLayout

Убедитесь, что в файл `build.gradle` вашего приложения добавлена `build.gradle` зависимость:

```
compile 'com.android.support:design:25.3.1'
```

Показывать подсказку из EditText как плавающей метки при вводе значения.

```
<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>
```

Для отображения значка глаза для отображения пароля с помощью TextInputLayout мы можем использовать следующий код:

```
<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>
```

где требуется `app:passwordToggleEnabled="true"` и `android:inputType="textPassword"` .

`app` должно использовать пространство имен `xmlns:app="http://schemas.android.com/apk/res-auto"`

Вы можете найти более подробную информацию и примеры в отдельной теме .

Добавление TabLayout

TabLayout предоставляет горизонтальную компоновку для отображения вкладок и обычно используется вместе с ViewPager .

Убедитесь, что в файл `build.gradle` вашего приложения добавлена `build.gradle` зависимость:

```
compile 'com.android.support:design:25.3.1'
```

Теперь вы можете добавлять элементы в TabLayout в свой макет, используя класс TabItem .

Например:

```
<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>
```

Добавьте `OnTabSelectedListener` для уведомления, когда вкладка `TabLayout` выбрана / не выбрана / повторно выбрана:

```
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) {
```

```
    }
});
```

Вкладки можно также добавлять / удалять из `TabLayout` программно.

```
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` имеет два режима: фиксированный и прокручиваемый.

```
tabLayout.setTabMode(TabLayout.MODE_FIXED);
tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);
```

Они также могут применяться в XML:

```
<android.support.design.widget.TabLayout
    android:id="@+id/tabLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:tabMode="fixed|scrollable" />
```

Примечание. Режимы `TabLayout` являются взаимоисключающими, что означает, что только один может быть активным одновременно.

Цвет индикатора вкладки - это цвет акцента, определенный для вашей темы Material Design.
Вы можете переопределить этот цвет, указав пользовательский стиль в `styles.xml` а затем применив стиль к вашему TabLayout:

```
<style name="MyCustomTabLayoutStyle" parent="Widget.Design.TabLayout">
    <item name="tabIndicatorColor">@color/your_color</item>
</style>
```

Затем вы можете применить стиль к виду, используя:

```
<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

Эффект эффекта пульсации был введен с материальным дизайном в Android 5.0 (уровень API 21), а анимация реализована новым классом RippleDrawable .

> Drawable, который показывает эффект пульсации в ответ на изменения состояния. Позиция привязки пульсации для заданного состояния может быть задана путем вызова `setHotspot(float x, float y)` с соответствующим идентификатором атрибута состояния.

5.0

В общем, эффект пульсации для **обычных кнопок работает по умолчанию** в API 21 и выше, а для других осязаемых представлений это может быть достигнуто путем указания:

```
android:background="?android:attr/selectableItemBackground">
```

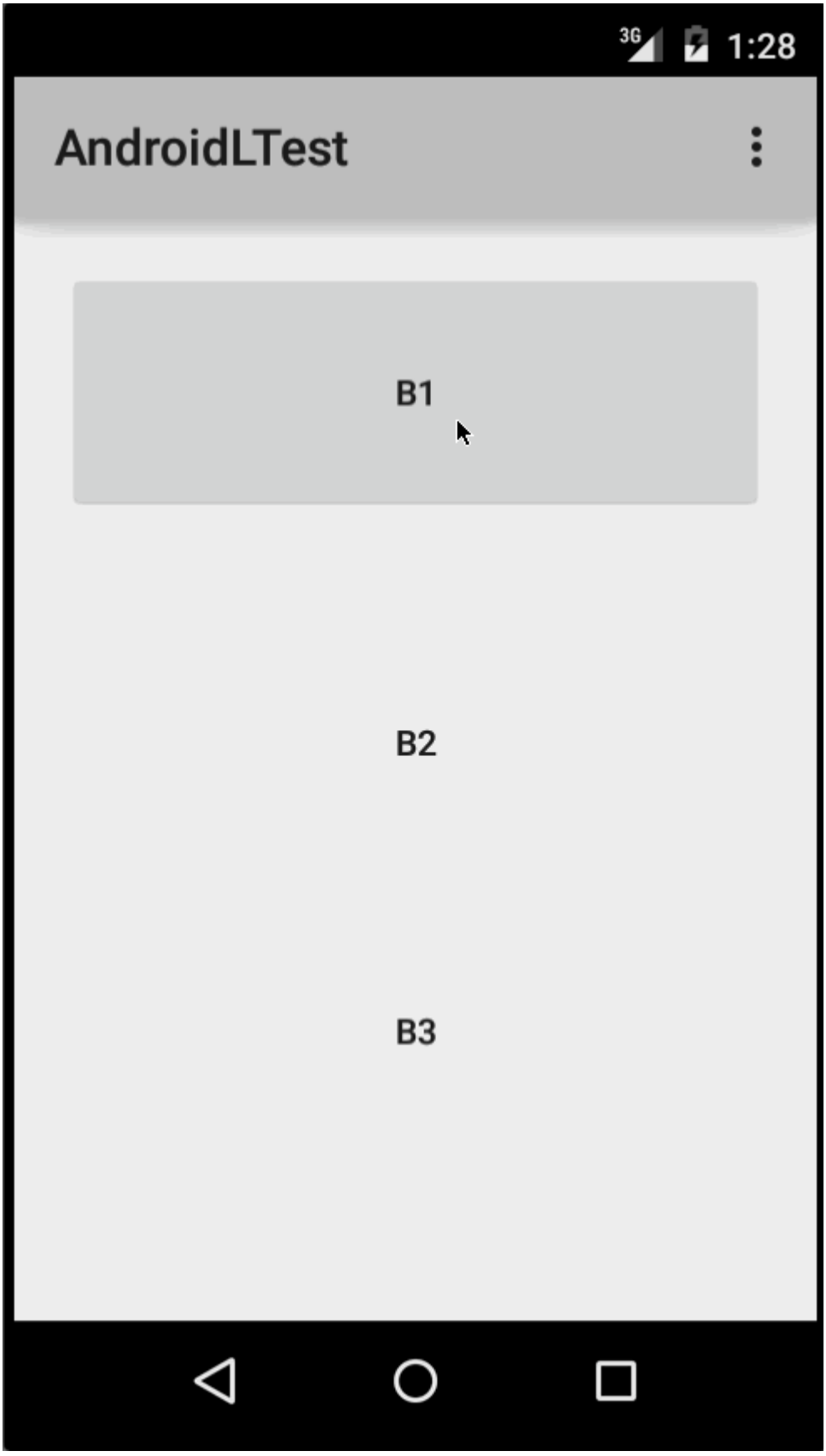
для рябь, содержащейся в представлении, или:

```
android:background="?android:attr/selectableItemBackgroundBorderless"
```

для ряби, которые простираются за пределы взгляда.

Например, на изображении ниже,

- B1 - кнопка, которая не имеет фона,
- B2 настроен с помощью `android:background="android:attr/selectableItemBackground"`
- B3 настроен с помощью `android:background="android:attr/selectableItemBackgroundBorderless"`



(Любезность изображения: http://blog.csdn.net/a396901990/article/details/40187203)

Вы можете сделать то же самое в коде, используя:

```
int[] attrs = new int[]{R.attr.selectableItemBackground};
TypedArray typedArray = getActivity().obtainStyledAttributes(attrs);
```

```
int backgroundResource = typedArray.getResourceId(0, 0);
myView.setBackgroundResource(backgroundResource);
```

Ряды также могут быть добавлены в представление с использованием атрибута `android:foreground` же, как указано выше. Как следует из названия, в случае, если рябь добавляется на передний план, рябь будет отображаться над любым видом, к `LinearLayout` она добавлена (например, `ImageView` , `LinearLayout` содержащая несколько просмотров и т. Д.).

Если вы хотите настроить эффект пульсации в представлении, вам нужно создать новый `XML` файл внутри доступного каталога.

Вот несколько примеров:

Пример 1 : Неограниченная рябь

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#ffff0000" />
```

Пример 2 : пульсация с маской и цветом фона

```
<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>
```

Если есть `view` с фоном, *уже заданным* с помощью `shape` , `corners` и любых других тегов, чтобы добавить рябь к этому представлению, используйте `mask layer` и установите рябь в качестве фона представления.

Пример :

```
 <?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>
```

Пример 3 : рябь сверху доступного ресурса

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:color="#ff0000ff">
    <item android:drawable="@drawable/my_drawable" />
</ripple>
```

Использование. Чтобы прикрепить ваш XML-файл пульсации к любому представлению, установите его как фоновый, как указано ниже (если ваш файл пульсаций называется `my_ripple.xml`):

```
<View
    android:id="@+id/myViewId"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/my_ripple" />
```

Селектор:

Выравнивание пульсации также можно использовать вместо селекторов списка состояний цвета, если ваша целевая версия v21 или выше (вы также можете поместить селектор пульсаций в папку `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>
```

В этом случае цвет состояния по умолчанию вашего представления будет белым, а нажатое состояние отобразит выровненную рябь.

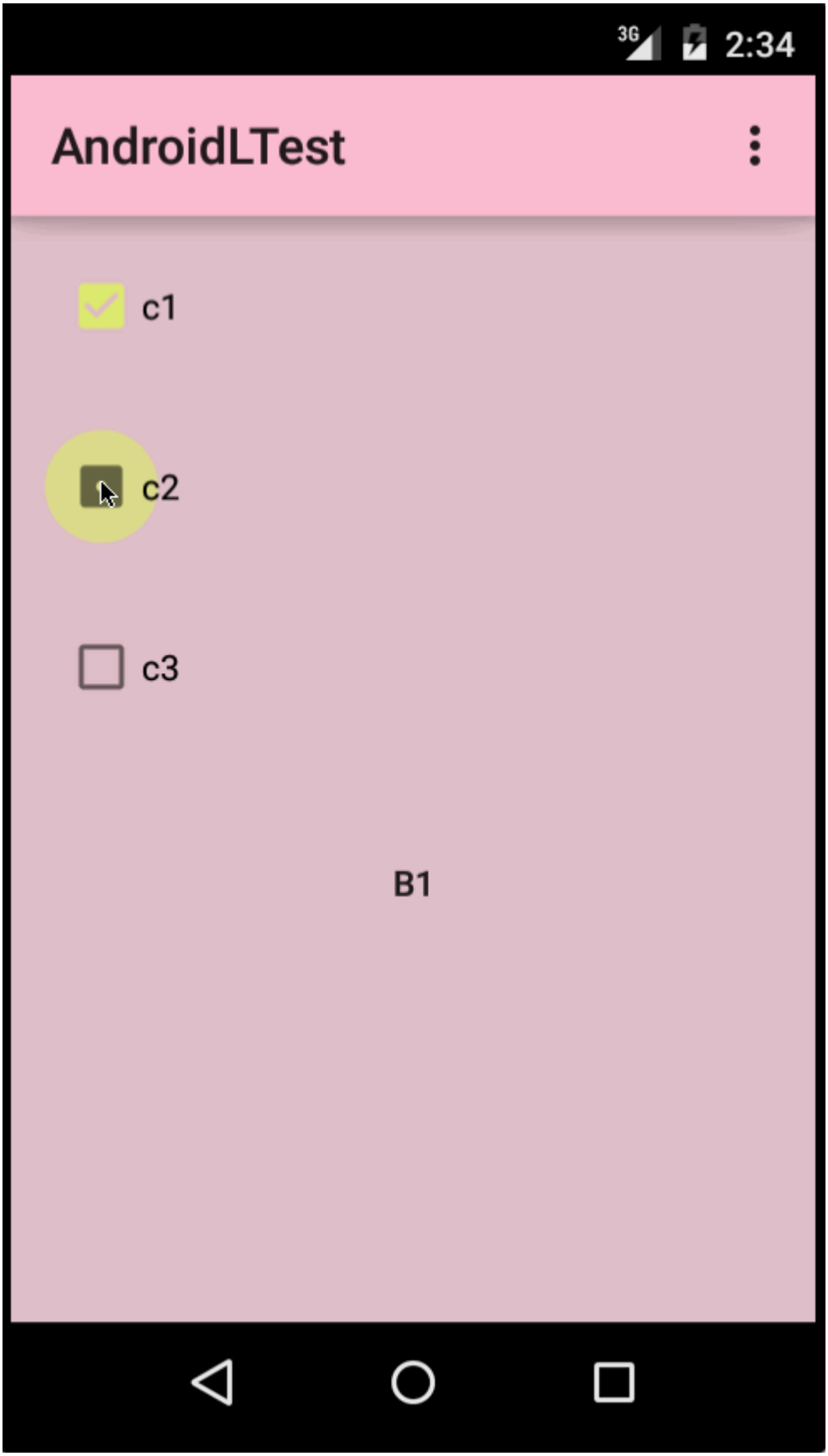
Обратите внимание: использование `?android:colorControlHighlight` придаст рябь того же цвета, что и встроенные ряби в вашем приложении.

Чтобы изменить только цвет пульсации, вы можете настроить цветной `android:colorControlHighlight` в своей теме:

```
<?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>
```

а затем используйте эту тему в своих действиях и т. д. Эффект будет похож на

изображение ниже:



(Любезность изображения: http://blog.csdn.net/a396901990/article/details/40187203)

Добавить ящик навигации

Навигационные ящики используются для перехода к адресатам верхнего уровня в приложении.

Убедитесь, что вы добавили библиотеку поддержки дизайна в файл `build.gradle` в зависимости от зависимостей:

```
 dependencies {
    // ...
    compile 'com.android.support:design:25.3.1'
}
```

Затем добавьте `DrawerLayout` и `NavigationView` в файл ресурсов макета XML. `DrawerLayout` - это просто фантастический контейнер, который позволяет `NavigationView` , фактическому навигационному `DrawerLayout` слева или справа от экрана. Примечание. Для мобильных устройств стандартный размер ящика составляет 320 дп.

```
<!-- 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>
```

Теперь, если хотите, создайте **файл заголовка,** который будет служить верхней частью вашего навигационного ящика. Это используется, чтобы придать гораздо более элегантный вид ящику.

```
<!-- 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>
```

Он упоминается в теге `NavigationView` в `app:headerLayout="@layout/drawer_header"` .
Это `app:headerLayout` раздувает указанный макет в заголовке. Это можно сделать во время выполнения с помощью:

```
// 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);
```

Чтобы автоматически заполнить навигационный ящик навигационными элементами, совместимыми с материалами, создайте файл меню и добавьте элементы по мере необходимости. Примечание: в то время как значки для элементов не требуются, они предлагаются в спецификации Material Design .
Он упоминается в теге `NavigationView` в `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>
```

Чтобы разделить элементы на группы, поместите их в `<menu>` вложенные в другой `<item>` с атрибутом `android:title` или оберните их тегом `<group>` .

Теперь, когда макет сделан, перейдите к коду `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();
```

Теперь вы можете делать все, что хотите, в виде заголовка `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);
```

Представление заголовка ведет себя подобно любому другому `View` , поэтому, как только вы используете `findViewById()` и добавляете другие файлы `View` s в свой файл макета, вы можете установить свойства чего-либо в нем.

Вы можете найти более подробную информацию и примеры в отдельной теме .

Нижние листы в библиотеке поддержки дизайна

Нижние листы скользят вверх от нижней части экрана, чтобы показать больше контента. Они были добавлены в библиотеку поддержки Android в версии v.1.1.1.0 и поддерживают, прежде всего, версии.

Убедитесь, что в файл build.gradle вашего приложения добавлена следующая зависимость:

```
compile 'com.android.support:design:25.3.1'
```

Стойкие нижние листы

Вы можете достичь `BottomSheetBehavior` листа, содержащего `BottomSheetBehavior` для ребенка. Вид `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>
```

Затем в вашем коде вы можете создать ссылку, используя:

```
// The View with the BottomSheetBehavior
View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

Вы можете установить состояние своего метода BottomSheetBehavior с помощью метода setState () :

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

Вы можете использовать одно из следующих состояний:

- `STATE_COLLAPSED` : это `STATE_COLLAPSED` состояние по умолчанию и показывает только

часть макета по дну. Высота можно контролировать с помощью атрибута `app:behavior_peekHeight` (по умолчанию 0)

- `STATE_EXPANDED` : полностью расширенное состояние нижнего листа, где видна вся нижняя `STATE_EXPANDED` (если ее высота меньше, чем содержащая `CoordinatorLayout`) или заполняется весь `CoordinatorLayout`

- `STATE_HIDDEN` : отключено по умолчанию (и включено с атрибутом `app:behavior_hideable`), что позволяет пользователям прокручивать нижний лист, чтобы полностью скрыть нижний лист

Кроме того, чтобы открыть или закрыть BottomSheet при щелчке View по вашему выбору, кнопка, скажем, вот как переключить поведение листа и обновление.

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

Но поведение `BottomSheet` также имеет функцию, при которой пользователь может взаимодействовать со щелчком вверх или вниз с помощью движения DRAG. В таком случае мы не сможем обновить зависимый вид (например, кнопку выше). Если состояние листа изменилось. В этом случае вы хотите получать обратные вызовы изменений состояния, поэтому вы можете добавить `BottomSheetCallback` для прослушивания событий пользовательского салфетки:

```
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
   }
 });
```

И если вы хотите, чтобы ваш нижний лист был виден только в режимах COLLAPSED и EXPANDED и никогда не использовал HIDE:

```
mBottomSheetBehavior2.setHideable(false);
```

Диалоговое окно с нижним листом

Вы также можете отобразить BottomSheetDialogFragment вместо представления в нижнем листе. Для этого вам сначала нужно создать новый класс, который расширяет BottomSheetDialogFragment.

Внутри `setupDialog()` вы можете раздуть новый файл макета и получить представление BottomSheetBehavior в представлении контейнера в своей деятельности. После того, как у вас есть это поведение, вы можете создать и связать BottomSheetCallback с ним, чтобы убрать фрагмент, когда лист скрыт.

```
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);
        }
    }
}
```

Наконец, вы можете вызвать show () в экземпляре вашего фрагмента, чтобы отобразить

его в нижнем листе.

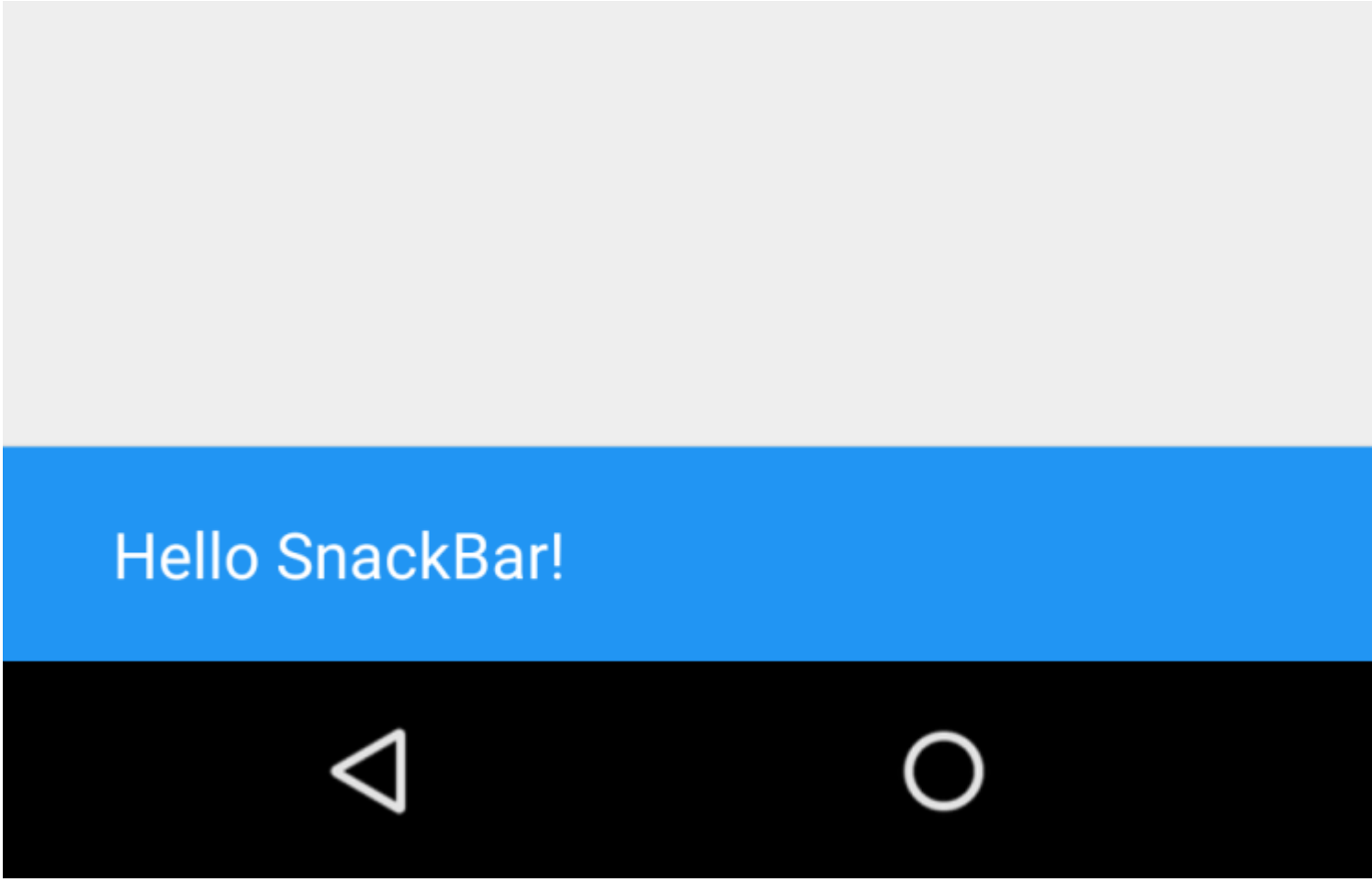
```
BottomSheetDialogFragment bottomSheetDialogFragment = new BottomSheetDialogFragmentExample();
bottomSheetDialogFragment.show(getSupportFragmentManager(),
bottomSheetDialogFragment.getTag());
```

Вы можете найти более подробную информацию по теме

Добавить Закусочную

Одной из основных особенностей Material Design является добавление `Snackbar` , который теоретически заменяет предыдущий `Toast` . Согласно документации на Android:

> Закусочная содержит одну строку текста, непосредственно связанную с выполненной операцией. Они могут содержать текстовое действие, но без значков. Тосты в основном используются для системных сообщений. Они также отображаются в нижней части экрана, но не могут быть выведены за пределы экрана.



Тосты все еще могут использоваться в Android для отображения сообщений пользователям, однако, если вы решили выбрать использование материалов в своем приложении, рекомендуется использовать закусочную. Вместо того, чтобы отображаться как наложение на вашем экране, `Snackbar` появляется снизу.

Вот как это делается:

```
Snackbar snackbar = Snackbar
        .make(coordinatorLayout, "Here is your new Snackbar", Snackbar.LENGTH_LONG);
snackbar.show();
```

Что касается времени, чтобы показать `Snackbar` , у нас есть варианты, похожие на те, которые предлагаются `Toast` или мы можем установить индивидуальную продолжительность в миллисекундах:

- `LENGTH_SHORT`
- `LENGTH_LONG`
- `LENGTH_INDEFINITE`
- `setDuration()` (начиная с версии `22.2.1`)

Вы также можете добавить динамические функции в свою `Snackbar` такую как `ActionCallback` или собственный цвет. Однако обратите внимание на руководство по дизайну, предлагаемое Android при настройке `Snackbar` .

Однако использование `Snackbar` имеет одно ограничение. Родительский макет представления, который вы собираетесь внедрить в `Snackbar` должен быть `CoordinatorLayout` . Это позволяет сделать фактическое всплывающее окно со дна.

Вот как определить `CoordinatorLayout` в вашем XML-файле макета:

```
<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>
```

Затем `CoordinatorLayout` необходимо определить в методе `onCreate` в вашей деятельности, а затем использовать при создании самой `Snackbar` .

Для получения дополнительной информации о `Snackbar` , пожалуйста, проверьте официальную документацию или специальную тему в документации.

Прочитайте Дизайн материалов онлайн: https://riptutorial.com/ru/android/topic/124/дизайн-материалов

глава 126: Добавление FuseView в проект Android

Вступление

Экспортируйте Fuse.View из fusetools и используйте его в существующем проекте Android.

Наша цель - экспортировать все приложение примера hikr и использовать его внутри `Activity` .

Окончательная работа может быть найдена @ lucamtudor / hikr-fuse-view

Examples

hikr, просто еще один android.view.View

Предпосылки

- у вас должен быть установлен предохранитель (https://www.fusetools.com/downloads)
- вы должны были сделать введение учебника
- в терминале: `fuse install android`
- в терминале: `uno install Fuse.Views`

Шаг 1

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

Шаг 2. Добавьте ссылку пакета на `Fuse.Views`

Найдите файл `hikr.unoproj` внутри корневой папки проекта и добавьте `"Fuse.Views"` в массив `"Packages"` .

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

Шаг 3. Сделайте компонент `HikrApp` для хранения всего приложения.

3.1. В корневой папке проекта создайте новый файл `HikrApp.ux` и вставьте содержимое `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. В `HikrApp.ux`

- замените теги `<App>` на `<Page>`
- добавьте `ux:Class="HikrApp"` к открытию `<Page>`
- удалите `<ClientPanel>` , нам больше не нужно беспокоиться о строке состояния или нижних навигационных кнопках

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 Используйте вновь созданный `HikrApp` компонент внутри `MainView.ux`

Замените содержимое файла `MainView.ux` на:

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

Наше приложение возвращается к нормальному поведению, но теперь мы извлекли его в отдельный компонент `HikrApp`

Шаг 4 Внутри `MainView.ux` замените теги `<App>` на `<ExportedViews>` и добавьте `ux:Template="HikrAppView"` в `<HikrApp />`

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

Помните шаблон `HikrAppView` , потому что нам понадобится его, чтобы получить ссылку на наш взгляд с Java.

Примечание . Из документов плавких предохранителей:

> `ExportedViews` будет вести себя как `App` при выполнении обычного `fuse preview` и `uno build`

Не правда. Вы получите эту ошибку при предварительном просмотре из Fuse Studio:

> Ошибка: не удалось найти тег приложения в любом из включенных файлов UX. Вы забыли включить UX-файл, содержащий тег приложения?

Шаг 5 `SplashPage.ux` `<DockPanel>` в `<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>
```

Шаг 6 Экспортируйте проект плавкого предохранителя в виде aar library

- в терминале, в корневой папке проекта: `uno clean`
- в терминале, в корневой папке проекта: `uno build -t=android -DLIBRARY`

Шаг 7 Подготовьте свой проект Android

- скопируйте aar из `.../rootHikeProject/build/Android/Debug/app/build/outputs/aar/app-debug.aar .../androidRootProject/app/libs`

  `.../rootHikeProject/build/Android/Debug/app/build/outputs/aar/app-debug.aar` для `.../androidRootProject/app/libs`
- добавьте `flatDir { dirs 'libs' }` в файл 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'
        }
    }
}

...
```

- add `compile(name: 'app-debug', ext: 'aar')` для зависимостей в `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'
}
```

- добавьте следующие свойства к активности внутри `AndroidManifest.xml`

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

Ваш `AndroidManifest.xml` будет выглядеть так:

```
<?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>
```

Шаг 8 : Покажите `Fuse.View HikrAppView` в вашей `Activity`

- обратите внимание, что ваша `Activity` должна наследовать `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

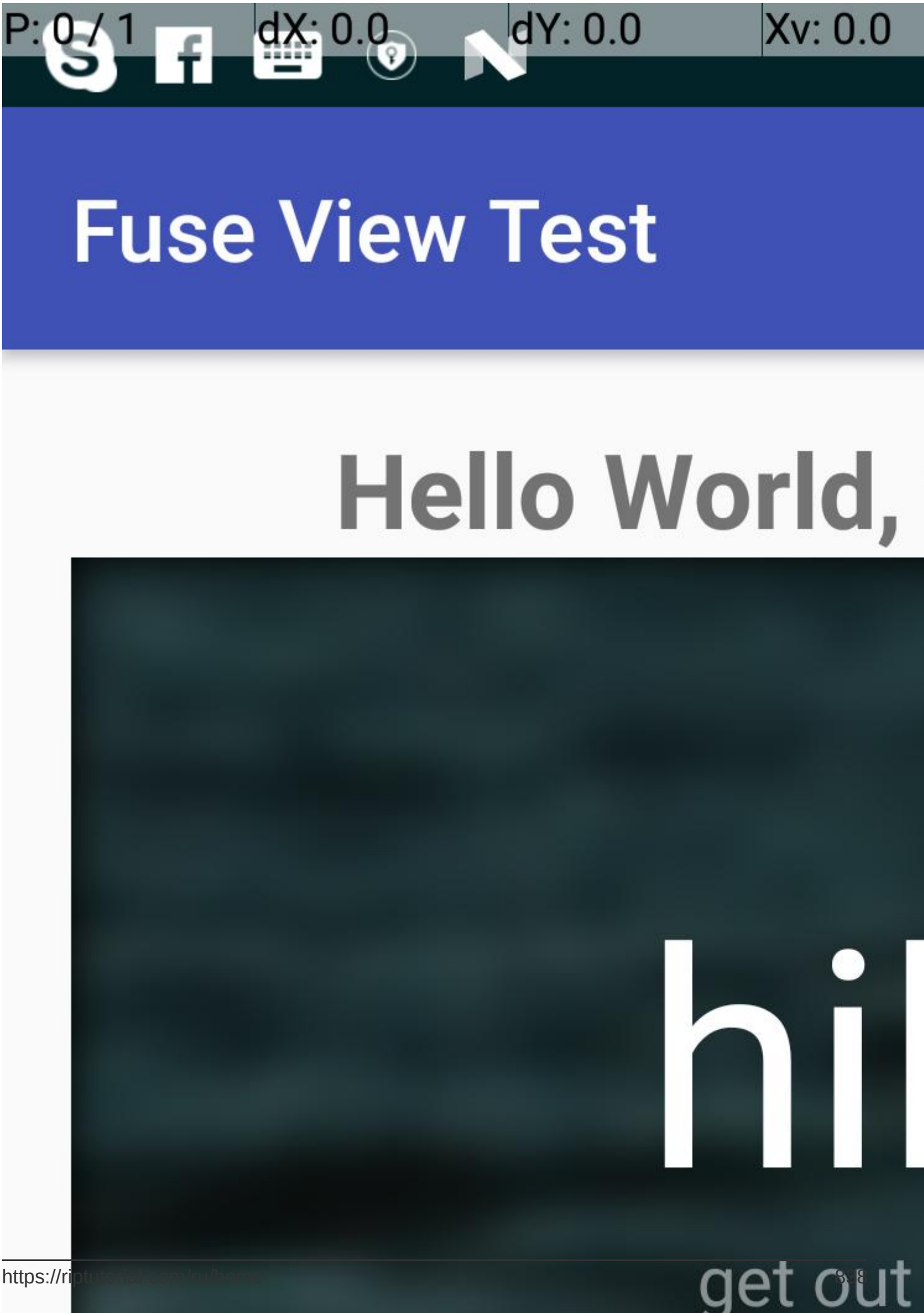
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"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

Заметка

Когда вы нажмете кнопку «Назад», на Android появится сообщение об ошибке. Вы можете следить за этим вопросом на форуме предохранителей .

```
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
```

И конечный результат - это нечто подобное. Вы также можете найти короткий клип на github .


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

https://riptutorial.com/ru/android/topic/10052/добавление-fuseview-в-проект-android

глава 127: Доступ к базам данных SQLite с использованием класса ContentValues

Examples

Вставка и обновление строк в базе данных SQLite

Во-первых, вам нужно открыть базу данных SQLite, которая может быть выполнена следующим образом:

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

После открытия базы данных вы можете легко вставлять или обновлять строки, используя класс `ContentValues` . В следующих примерах предполагается, что первое имя дается `str_edtfname` и последнее имя `str_edtlname` . Вам также нужно заменить `table_name` именем вашей таблицы, которое вы хотите изменить.

Вставка данных

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

Обновление данных

```
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});
```

Прочитайте Доступ к базам данных SQLite с использованием класса ContentValues онлайн: https://riptutorial.com/ru/android/topic/10154/доступ-к-базам-данных-sqlite-с-использованием-класса-contentvalues

глава 128: Жизненный цикл пользовательского интерфейса

Examples

Сохранение данных об обрезке

```
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);
    }
}
```

объяснение

Итак, что здесь происходит?

Система Android всегда будет пытаться очистить как можно больше памяти. Итак, если ваша деятельность не соответствует фону, а другая деятельность переднего плана требует своей доли, система Android будет вызывать `onTrimMemory()` для вашей активности.

Но это не означает, что все ваши свойства должны исчезнуть. Что вы должны сделать, так это сохранить их в объект Bundle. Объект Bundle намного лучше обрабатывает память. Внутри пакета каждый объект идентифицируется уникальной текстовой последовательностью - в приведенном выше примере целочисленная переменная значения `mArg` выполняется под ссылочным именем `EXAMPLE_ARG` . И когда активность воссоздается, извлеките старые значения из объекта Bundle, а не воссоздавайте их с нуля

Прочитайте Жизненный цикл пользовательского интерфейса онлайн: https://riptutorial.com/ru/android/topic/3440/жизненный-цикл-пользовательского-интерфейса

глава 129: Закусочная

Синтаксис

- Snackbar make (View view, текст CharSequence, int duration)
- Snackbar make (View view, int resId, int duration)

параметры

параметр	Описание
Посмотреть	Вид: представление для поиска родителя.
текст	CharSequence: текст для показа. Можно форматировать текст.
RESID	int: Идентификатор ресурса используемого строкового ресурса. Можно форматировать текст.
продолжительность	int: Как долго показывать сообщение. Это может быть LENGTH_SHORT, LENGTH_LONG или LENGTH_INDEFINITE

замечания

Закусочная обеспечивает легкую обратную связь об операции. Он отображает краткое сообщение в нижней части экрана на мобильном телефоне и внизу слева на больших устройствах. Закуски отображаются выше всех других элементов на экране, и только один может отображаться за раз.

Они автоматически исчезают после таймаута или после взаимодействия с пользователем в другом месте экрана, особенно после взаимодействий, вызывающих новую поверхность или активность. Закусочную можно снять с экрана.

Перед использованием `SnackBar` вы должны добавить зависимость библиотеки поддержки дизайна в файле `build.gradle` :

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

Официальная документация

https://developer.android.com/reference/android/support/design/widget/Snackbar.html

Examples

Создание простой закусочной

Создание `Snackbar` можно сделать следующим образом:

```
Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG).show();
```

`view` используется , чтобы найти подходящий родитель , чтобы использовать для отображения `Snackbar` . Обычно это будет `CoordinatorLayout` который вы определили в своем XML, что позволяет добавлять такие функции, как прокрутка, чтобы отклонять и автоматически перемещать другие виджеты (например, `FloatingActionButton`). Если нет `CoordinatorLayout` тогда используется контент содержимого окна.

Очень часто мы также добавляем действие в `Snackbar` . Общим вариантом использования будет действие «Отменить».

```
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();
```

Вы можете создать `Snackbar` и показать его позже:

```
Snackbar snackbar = Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG);
snackbar.show();
```

Если вы хотите изменить цвет текста `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();
```

По умолчанию `Snackbar` от правильного swipe.This пример демонстрирует, как отклонить snackBar на его левый салфетки .

Специальная закусочная

Функция для настройки закуски

```
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;

    }
```

Вызвать функцию из фрагмента или действия

```
 SnackBar.makeText(MyActivity.this, "Please Locate your address at Map",
Snackbar.LENGTH_SHORT).show();
```

Закусочная с обратным вызовом

Вы можете использовать Snackbar.Callback для прослушивания, если снэк-бар был уволен пользователем или тайм-аутом.

```
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();
```

Специальная закусочная

В этом примере показана белая закусочная с пользовательским значком отмены.

```
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();
```

Закуски против тостов: Какой из них я должен использовать?

Тосты обычно используются, когда мы хотим отобразить пользователю информацию о некоторых действиях, которые успешно (или нет) произошли, и это действие не требует от пользователя каких-либо других действий. Например, когда отправлено сообщение, например:

```
Toast.makeText(this, "Message Sent!", Toast.LENGTH_SHORT).show();
```

Закуски также используются для отображения информации. Но на этот раз мы можем дать пользователю возможность принять меры. Например, допустим, пользователь удалил изображение по ошибке, и он хочет вернуть его. Мы можем предоставить Snackbar с действием «Отменить». Как это:

```
Snackbar.make(getCurrentFocus(), "Picture Deleted", Snackbar.LENGTH_SHORT)
        .setAction("Undo", new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Return his picture
            }
```

```
        })
        .show();
```

Вывод: Тосты используются, когда нам не нужно взаимодействие с пользователем. Закуски используются, чтобы позволить пользователям выполнить другое действие или отменить предыдущий.

Пользовательская закуска (нет необходимости)

Создание Snackbar без необходимости просмотра прохода в Snackbar, все макеты создают в android в 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();
    }
}
```

для класса вызовов

новый CustomSnackBar (findViewById (CustomSnackBar.VIEW_PARENT), «сообщение», CustomSnackBar.STATE_ERROR);

Прочитайте Закусочная онлайн: https://riptutorial.com/ru/android/topic/1500/закусочная

глава 130: залп

Вступление

Volley - это Android-HTTP-библиотека, которая была представлена Google для упрощения сетевых вызовов. По умолчанию все сетевые вызовы Volley производятся асинхронно, обрабатывая все в фоновом потоке и возвращая результаты на переднем плане с использованием обратных вызовов. Поскольку выборка данных по сети является одной из наиболее распространенных задач, выполняемых в любом приложении, библиотека Volley была создана для упрощения разработки приложений для Android.

Синтаксис

- RequestQueue queue = Volley.newRequestQueue (контекст); // настройка очереди
- Запрос запроса = новый SomeKindOfRequestClass (Request.Method, String url, Response.Listener, Response.ErrorListener); // настраиваем какой-то запрос, точный тип и аргументы изменяются для каждого типа запроса
- queue.add (запрос); // добавить запрос в очередь; соответствующий ответчик ответа будет вызываться после завершения запроса (или прекращения по какой-либо причине)

замечания

Монтаж

Вы можете создать Volley из официального исходного кода Google . Некоторое время это был единственный вариант. Или используя одну из сторонних готовых версий. Однако Google наконец выпустил официальный пакет maven на jcenter.

В файле `build.gradle` уровне `build.gradle` добавьте это в список зависимостей:

```
dependencies {
    ...
    compile 'com.android.volley:volley:1.0.0'
}
```

Убедитесь, что разрешение `INTERNET` установлено в манифесте вашего приложения:

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

Официальная документация

Google не предоставил очень обширную документацию по этой библиотеке, и они не затрагивали ее в течение многих лет. Но то, что доступно, можно найти по адресу:

https://developer.android.com/training/volley/index.html

Существует неофициальная документация, размещенная на GitHub, хотя должно быть лучшее место для размещения этого в будущем:

https://pablobaxter.github.io/volley-docs/

Examples

Basic StringRequest с использованием метода 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);
```

Отменить запрос

```
// 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);
```

Добавление атрибутов времени разработки в NetworkImageView

Есть несколько дополнительных атрибутов, которые Volley `NetworkImageView` добавляет к стандарту `ImageView` . Однако эти атрибуты могут быть установлены только в коде. Ниже приведен пример того, как создать класс расширения, который будет `NetworkImageView` атрибуты из вашего файла макета XML и применять их к экземпляру `NetworkImageView` для вас.

В каталоге `~/res/xml` добавьте файл с именем `attrx.xml` :

```
<resources>
    <declare-styleable name="MoreNetworkImageView">
        <attr name="defaultImageResId" format="reference"/>
        <attr name="errorImageResId" format="reference"/>
    </declare-styleable>
</resources>
```

Добавьте новый файл класса в свой проект:

```
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);
        }
    }
}
```

Пример файла макета, показывающий использование пользовательских атрибутов:

```
<?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>
```

Запросить 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);
```

Добавление пользовательских заголовков к вашим запросам [например, для основного auth]

Если вам нужно добавить пользовательские заголовки к вашим запросам волейбола, вы не сможете сделать это после инициализации, так как заголовки сохраняются в частной переменной.

Вместо этого вам необходимо переопределить метод `getHeaders()` для `Request.class` как таковой:

```
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;
    }
};
```

Объяснение параметров:

- `REQUEST_METHOD` - любой из `Request.Method.*` .
- `REQUEST_URL` - полный URL-адрес для отправки вашего запроса.
- `REQUEST_BODY` - `JSONObject` содержащий POST-тело для отправки (или null).
- `RESP_LISTENER` - объект `Response.Listener<?>` , `onResponse(T data)` вызван успешным завершением.
- `ERR_LISTENER` - объект `Response.ErrorListener` , `onErrorResponse(VolleyError e)` вызван неудачным запросом.

Если вы хотите создать собственный запрос, вы можете также добавить в него заголовки:

```
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;
    }
    ...
```

```
}
```

Класс помощника для обработки ошибок волейбола

```
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);
        }
```

Аутентификация удаленного сервера с помощью метода StringRequest через метод POST

Для этого примера предположим, что у нас есть сервер для обработки запросов POST, которые мы будем делать из нашего приложения для 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();
}
```

Использование запросов Volley для HTTP

Добавьте зависимость gradle в app-level build.gradle

```
compile 'com.android.volley:volley:1.0.0'
```

Кроме того, добавьте разрешение android.permission.INTERNET в манифест вашего приложения.

** Создайте экземпляр Singleline RequestLueue во время вашего приложения **

```
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;
}
}
```

Теперь вы можете использовать экземпляр volley с помощью метода getInstance () и добавить новый запрос в очередь с помощью
`InitApplication.getInstance().addToQueue(request);`

Простым примером запроса JsonObject с сервера является

```
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));
```

Для обработки тайм-аута Volley вам необходимо использовать `RetryPolicy` . Политика повтора используется в случае, если запрос не может быть завершен из-за сбоя сети или некоторых других случаев.

Volley предоставляет простой способ реализовать `RetryPolicy` для ваших запросов. По умолчанию Volley устанавливает все таймауты сокетов и соединений на 5 секунд для всех запросов. `RetryPolicy` - это интерфейс, в котором вам необходимо реализовать свою логику того, как вы хотите повторить определенный запрос, когда произойдет тайм-аут.

Конструктор принимает следующие три параметра:

- `initialTimeoutMs` - `initialTimeoutMs` тайм-аут сокета в миллисекундах для каждой попытки повтора.
- `maxNumRetries` - количество попыток повторения попыток.
- `backoffMultiplier` - множитель, который используется для определения экспоненциального времени, установленного в socket для каждой попытки повтора.

Логический ответ переменной с сервера с запросом json в залпе

вы можете настраивать класс ниже одного

```
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;
        }
    }
}
```

используйте это с вашей деятельностью

```
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();
    }
```

Использовать JSONArray в качестве тела запроса

Запросы по умолчанию, интегрированные в залп, не позволяют передать `JSONArray` качестве тела запроса в запросе `POST` . Вместо этого вы можете передавать объект `JSON` в качестве параметра.

Однако вместо передачи объекта `JSON` в качестве параметра конструктору запроса вам необходимо переопределить метод `getBody()` класса `Request.class` . Вы должны передать `null` как третий параметр:

```
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;
        }
    }
};
```

Объяснение параметров:

- `REQUEST_URL` - полный URL-адрес для отправки вашего запроса.
- `RESP_LISTENER` - объект `Response.Listener<?>` , `onResponse(T data)` вызван успешным завершением.
- `ERR_LISTENER` - объект `Response.ErrorListener` , `onErrorResponse(VolleyError e)` вызван неудачным запросом.

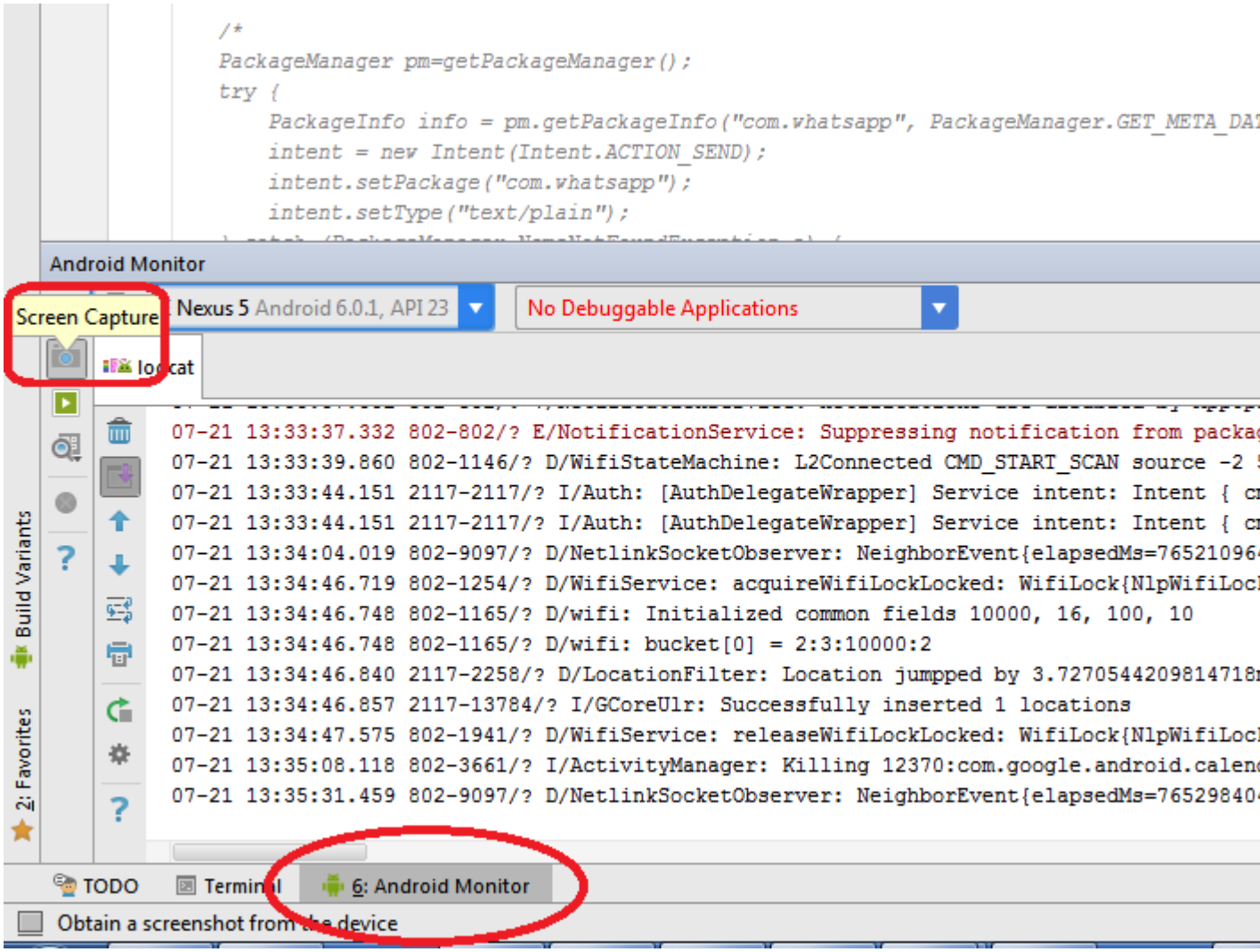
Прочитайте залп онлайн: https://riptutorial.com/ru/android/topic/2800/залп

глава 131: Захват скриншотов

Examples

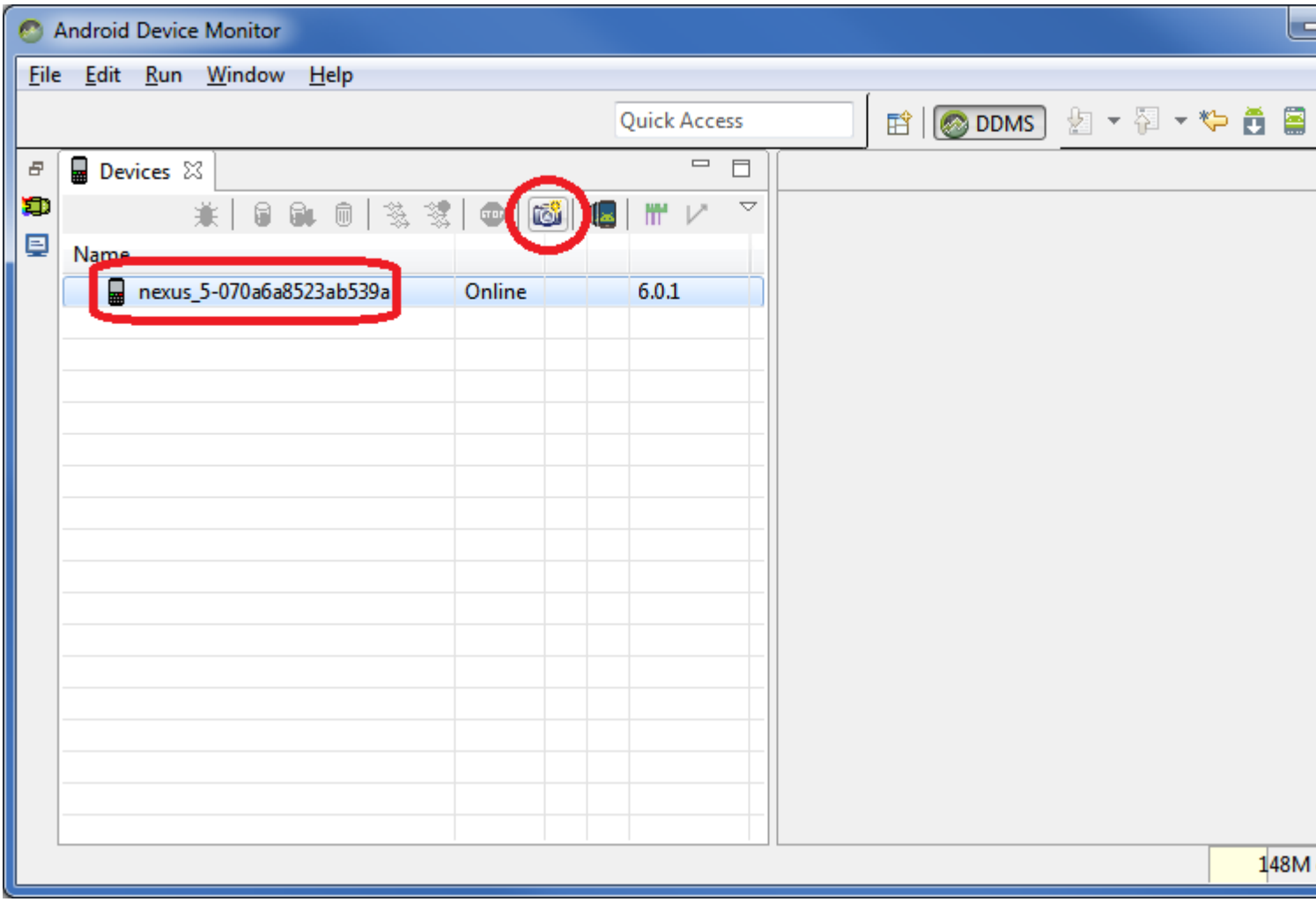
Захват скриншота через Android Studio

1. Открыть вкладку Android Monitor
2. Нажмите кнопку Screen Capture



Захват снимка экрана с помощью Android-устройства

1. Откройте Android Device Monitor (*например, C: <ANDROID_SDK_LOCATION> \ tools \ monitor.bat*)
2. Выберите устройство
3. Нажмите кнопку Screen Capture



Захват Скриншота через АБР

Пример ниже сохраняет скриншот внутреннего хранилища устройств.

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

Захват Скриншота через АБР и сохранение непосредственно на вашем ПК

Если вы используете Linux (или Windows с Cygwin), вы можете запустить:

```
adb shell screencap -p | sed 's/\r$//' > screenshot.png
```

Снимок экрана определенного вида

Если вы хотите сделать снимок экрана для конкретного View `v` , вы можете использовать следующий код:

```
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();
}
```

Прочитайте Захват скриншотов онлайн: https://riptutorial.com/ru/android/topic/4506/захват-скриншотов

глава 132: Звук и мультимедиа Android

Examples

Как выбрать изображение и видео для api> 19

Вот проверенный код для изображения и видео. Он будет работать для всех API менее 19 и более 19.

Образ:

```
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);
                    }
```

Видео:

```
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();
        }
```

Воспроизведение звуков через 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;
        }
}
```

Прочитайте Звук и мультимедиа Android онлайн: https://riptutorial.com/ru/android/topic/4730/звук-и-мультимедиа-android

глава 133: Звуковая дорожка

Examples

Генерировать тон определенной частоты

Чтобы воспроизвести звук с определенным тоном, мы сначала должны создать звук синусоидальной волны. Это делается следующим образом.

```
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
}
```

Теперь мы должны настроить AudioTrack для воспроизведения в соответствии с созданным буфером. Это делается следующим образом

```
AudioTrack audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC,
                    sampleRate, AudioFormat.CHANNEL_OUT_MONO,
                    AudioFormat.ENCODING_PCM_16BIT, buffer.length,
                    AudioTrack.MODE_STATIC);
```

Напишите сгенерированный буфер и воспроизведите трек

```
audioTrack.write(buffer, 0, buffer.length);
audioTrack.play();
```

Надеюсь это поможет :)

Прочитайте Звуковая дорожка онлайн: https://riptutorial.com/ru/android/topic/9155/звуковая-дорожка

глава 134: Изменения ориентации

замечания

Ссылка: https://guides.codepath.com/android/Handling-Configuration-Changes#references

Examples

Сохранение и восстановление состояния активности

По мере того как ваше действие начинает останавливаться, система вызывает `onSaveInstanceState()` поэтому ваша активность может сохранять информацию о состоянии с помощью набора пар ключ-значение. По умолчанию реализация этого метода автоматически сохраняет информацию о состоянии иерархии представлений активности, такую как текст в `EditText` или позиции прокрутки `ListView` .

Чтобы сохранить дополнительную информацию о состоянии вашей деятельности, вы должны реализовать `onSaveInstanceState()` и добавить пары ключ-значение в объект Bundle. Например:

```
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);
    }
}
```

Система вызовет этот метод до того, как действие будет уничтожено. Затем система будет вызывать `onRestoreInstanceState` где мы можем восстановить состояние из пакета:

```
@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);
}
```

Состояние экземпляра также можно восстановить в стандартном методе Activity # onCreate, но это удобно делать в `onRestoreInstanceState` который гарантирует, что вся

инициализация выполнена, и позволяет подклассам решать, использовать ли реализацию по умолчанию. Подробнее читайте в этой статье stackoverflow .

Обратите внимание, что `onSaveInstanceState` и `onRestoreInstanceState` не могут быть вызваны вместе. Android вызывает `onSaveInstanceState()` когда есть вероятность, что активность может быть уничтожена. Однако бывают случаи, когда `onSaveInstanceState` но действие не уничтожается и в результате `onRestoreInstanceState` не вызывается.

Сохранение и восстановление состояния фрагмента

Фрагменты также имеют метод `onSaveInstanceState()` который вызывается, когда их состояние необходимо сохранить:

```
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);
    }
}
```

Затем мы можем вытащить данные из этого сохраненного состояния в `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;
   }
}
```

Чтобы состояние фрагмента сохранялось должным образом, мы должны быть уверены, что мы не будем излишне воссоздавать фрагмент при изменении конфигурации. Это означает быть осторожным, чтобы не повторно инициализировать существующие фрагменты, когда они уже существуют. Любые фрагменты, которые инициализируются в Activity, должны быть просмотрены тегом после изменения конфигурации:

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

Это требует от нас тщательного включения тега для поиска, когда вы добавляете фрагмент в операцию внутри транзакции:

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

С помощью этого простого шаблона мы можем правильно повторно использовать фрагменты и восстановить их состояние при изменении конфигурации.

Сохраняющиеся фрагменты

Во многих случаях мы можем избежать проблем при восстановлении активности путем простого использования фрагментов. Если ваши представления и состояние находятся внутри фрагмента, мы можем легко сохранить фрагмент при повторном создании действия:

```
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;
    }
}
```

Этот подход позволяет уничтожить фрагмент во время жизненного цикла активности. Вместо этого они сохраняются внутри диспетчера фрагментов. Смотрите Android официальные документы для получения дополнительной информации .

Теперь вы можете проверить, не существует ли фрагмент уже тегом перед его созданием, а фрагмент сохранит его состояние при изменении конфигурации. См Handling выполнения Changes руководство для получения более подробной информации .

Фиксирование ориентации экрана

Если вы хотите заблокировать изменение ориентации экрана на любом экране (активности) вашего приложения для Android, вам просто нужно установить свойство `android:screenOrientation` для `<activity>` в **AndroidManifest.xml** :

```
<activity
    android:name="com.techblogon.screenorientationexample.MainActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
    <!-- ... -->
</activity>
```

Теперь эта активность всегда отображается в режиме « **портрет** ».

Управление изменениями вручную

Если вашему приложению не требуется обновлять ресурсы во время определенного изменения конфигурации, и у вас есть ограничение по производительности, которое требует от вас перезапуска активности, вы можете объявить, что ваша активность сама обрабатывает сами изменения конфигурации, что предотвращает перезапуск системы деятельность.

Однако этот метод следует рассматривать как последнее средство, когда вы должны избегать перезапуска из-за изменения конфигурации и не рекомендуется для большинства приложений. Чтобы воспользоваться этим подходом, мы должны добавить узел `android:configChanges` в действие в **AndroidManifest.xml** :

```
<activity android:name=".MyActivity"
          android:configChanges="orientation|screenSize|keyboardHidden"
          android:label="@string/app_name">
```

Теперь, когда одна из этих конфигураций изменяется, активность не перезапускается, а вместо этого получает вызов `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();
    }
}
```

См. Документы « Обработка документов изменения» . Подробнее о том, какие изменения конфигурации вы можете обрабатывать в своей деятельности, см . В документации android: configChanges и классе Configuration .

Обработка AsyncTask

Проблема:

- Если после `AsyncTask` происходит поворот экрана, активность владельца уничтожается и воссоздается.
- Когда `AsyncTask` завершает работу, он хочет обновить пользовательский интерфейс, который больше недействителен.

Решение:

Используя Loaders , можно легко преодолеть разрушение / отдых.

Пример:

Основная деятельность:

```
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;
    }
}
```

Замечания:

Важно использовать библиотеку совместимости v4 или нет, но не используйте часть одной и части другой, так как это приведет к ошибкам компиляции. Чтобы проверить, вы можете посмотреть на импорт для `android.support.v4.content` и `android.content` (у вас не должно быть

обоих).

Заблокировать экранирование экрана программно

Очень часто во время разработки может оказаться **очень полезным блокировать** / **разблокировать экран устройства в определенных частях кода** .

Например, при отображении диалога с информацией разработчик может захотеть ***заблокировать*** *поворот экрана, чтобы предотвратить отклонение диалога и текущую активность от перестроить, чтобы* ***разблокировать*** *его снова, когда диалог отклонен.*

Хотя мы можем добиться блокировки вращения из манифеста, выполнив:

```
<activity
    android:name=".TheActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
</activity>
```

Можно сделать это также программно, выполнив следующие действия:

```
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);
        }
    }
}
```

А затем вызывая следующее, чтобы соответственно блокировать и разблокировать вращение устройства

```
lockDeviceRotation(true)
```

а также

```
lockDeviceRotation(false)
```

Прочитайте Изменения ориентации онлайн: https://riptutorial.com/ru/android/topic/4621/изменения-ориентации

глава 135: Измерение размеров

замечания

Обратите внимание, что экземпляр `ViewTreeObserver` связанный с экземпляром `View` может стать недействительным, пока этот `View` все еще жив. Из `View.getViewTreeObserver` View.getViewTreeObserver:

```
// The returned ViewTreeObserver observer is not guaranteed to remain
// valid for the lifetime of this View. If the caller of this method keeps
// a long-lived reference to ViewTreeObserver, it should always check for
// the return value of {@link ViewTreeObserver#isAlive()}.
```

Таким образом, если вы ранее добавили слушателя к экземпляру `ViewTreeObserver` и теперь хотите его удалить, проще всего вызвать `getViewTreeObserver` в соответствующем экземпляре `View` чтобы получить новый экземпляр `ViewTreeObserver` . (Проверка `isAlive` на существующий экземпляр больше подходит для небольшой пользы, если `ViewTreeObserver` больше не жив, вы все равно будете получать эту новую ссылку!)

Examples

Вычисление начальных размеров View в Activity

```
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);
    }

}
```

Прочитайте Измерение размеров онлайн: https://riptutorial.com/ru/android/topic/115/измерение-размеров

глава 136: Индикатор

замечания

Официальная документация: ProgressBar

Examples

Неопределенный прогрессBar

Неопределенный ProgressBar показывает циклическую анимацию без указания прогресса.

Основной неопределенный ProgressBar (прядильное колесо)

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Горизонтальный неопределенный ProgressBar (плоский стержень)

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Другие встроенные стили ProgressBar

```
style="@android:style/Widget.ProgressBar.Small"
style="@android:style/Widget.ProgressBar.Large"
style="@android:style/Widget.ProgressBar.Inverse"
style="@android:style/Widget.ProgressBar.Small.Inverse"
style="@android:style/Widget.ProgressBar.Large.Inverse"
```

Использовать неопределенный ProgressBar в Activity

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setVisibility(View.VISIBLE);
progressBar.setVisibility(View.GONE);
```

Определить ProgressBar

Определенный ProgressBar показывает текущий прогресс в отношении определенного максимального значения.

Горизонтальное определение ProgressBar

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="10dp"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Вертикальный определитель ProgressBar

```
<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"/>
```

Рез / рисуем / 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>
```

Кольцо определило 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"/>
```

Рез / рисуем / 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>
```

Использовать определённый ProgressBar в Activity.

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setSecondaryProgress(100);
progressBar.setProgress(10);
progressBar.setMax(100);
```

Индивидуальная панель прогресса

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 :

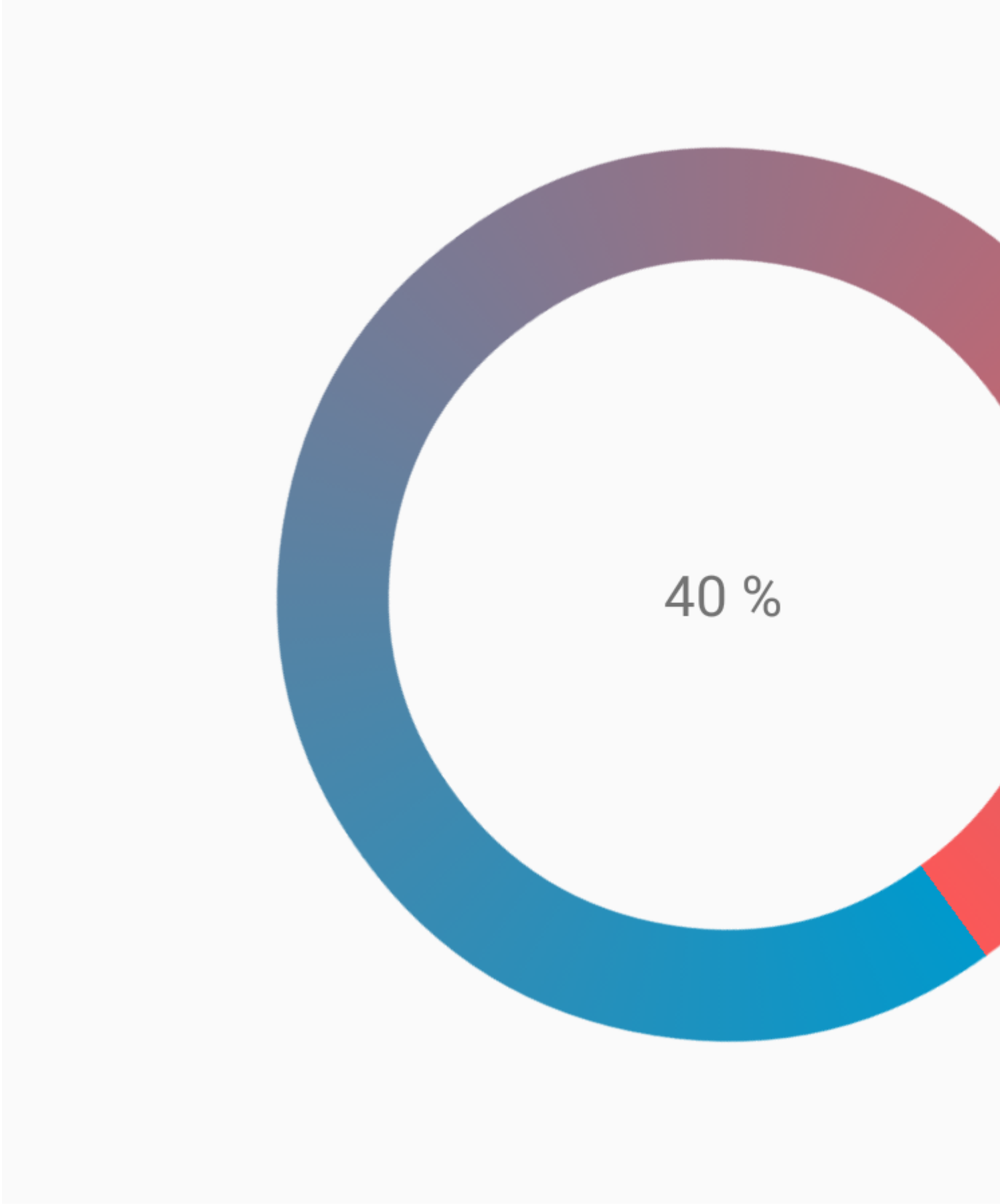
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    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>
```

Справочный снимок экрана:



Тонировка ProgressBar

Используя тему AppCompat, то `ProgressBar` цвет «s будет `colorAccent` вы определили.

5.0

Чтобы изменить цвет `ProgressBar` без изменения цвета акцента, вы можете использовать атрибут `android:theme` переопределяющий цвет акцента:

```
<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>
```

Чтобы отточить `ProgressBar` вы можете использовать в xml-файле атрибуты `android:indeterminateTintMode` и `android:indeterminateTint`

```
<ProgressBar
    android:indeterminateTintMode="src_in"
    android:indeterminateTint="@color/my_color"
/>
```

Материал линейный ProgressBar

Согласно Документам по материалам :

> Линейный индикатор прогресса всегда должен заполняться от 0% до 100% и никогда не уменьшаться.
> Он должен быть представлен штрихами на краю заголовка или листа, которые появляются и исчезают.

Чтобы использовать материал Linear ProgressBar, просто используйте в своем xml:

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

неопределенный

Чтобы создать **неопределенный** ProgressBar, установите для параметра `android:indeterminate` атрибут значение `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"/>
```

детерминированный

Чтобы создать **определенный** ProgressBar, установите для параметра `android:indeterminate` значение `false` и используйте атрибуты `android:max` и `android:progress` :

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"/>
```

Просто используйте этот код для обновления значения:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
```

буфер

Чтобы создать эффект **буфера** с помощью ProgressBar, установите для параметра `android:indeterminate` значение `false` и используйте параметры `android:max` , `android:progress` и `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"/>
```

Значение буфера определяется атрибутом `android:secondaryProgress` .
Просто используйте этот код для обновления значений:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
progressBar.setSecondaryProgress(50);
```

Неопределенный и определяемый

Для получения такого рода ProgressBar просто используйте неопределенный ProgressBar, используя атрибут `android:indeterminate` для true.

```
<ProgressBar
    android:id="@+id/progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="true"/>
```

Затем, когда вам нужно переключиться с неопределенного на определенный прогресс, используйте `setIndeterminate()` .

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setIndeterminate(false);
```

Создание настраиваемого диалогового окна выполнения

С помощью создания диалогового окна Custom Progress Dialog диалоговое окно можно использовать для отображения в экземпляре пользовательского интерфейса без повторного создания диалога.

Сначала создайте настраиваемый класс диалога выполнения.

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;
        }
    }
}
```

Теперь создаем настраиваемый макет выполнения

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>
```

Это оно. Теперь для вызова диалога в коде

```
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();
```

Прочитайте Индикатор онлайн: https://riptutorial.com/ru/android/topic/3353/индикатор

глава 137: Инструменты отчетов о сбоях

замечания

Самая полная полная вики доступна здесь, в github .

Examples

Ткань - Crashlytics

Fabric - это модульная мобильная платформа, которая предоставляет полезные наборы, которые вы можете комбинировать для создания своего приложения. **Crashlytics** - это инструмент отчетов о **сбоях** и проблемах, предоставляемый Fabric, который позволяет вам отслеживать и контролировать ваши приложения в деталях.

Как настроить Ткань-Crashlytics

Шаг 1. Измените файл `build.gradle` :

Добавьте плагин-репо и плагин 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.+'
  }
}
```

Примените плагин:

```
apply plugin: 'com.android.application'
//Put Fabric plugin after Android plugin
apply plugin: 'io.fabric'
```

Добавьте репозиторий Fabric:

```
repositories {
  maven { url 'https://maven.fabric.io/public' }
}
```

Добавьте комплект Crashlyrics:

```
dependencies {

  compile('com.crashlytics.sdk.android:crashlytics:2.6.6@aar') {
    transitive = true;
  }
}
```

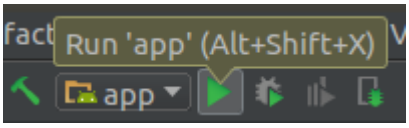
Шаг 2. Добавьте свой **ключ API** и разрешение **INTERNET** в `AndroidManifest.xml`

```
<?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>
```

Шаг 3. Инициируйте Kit во время выполнения кода, например:

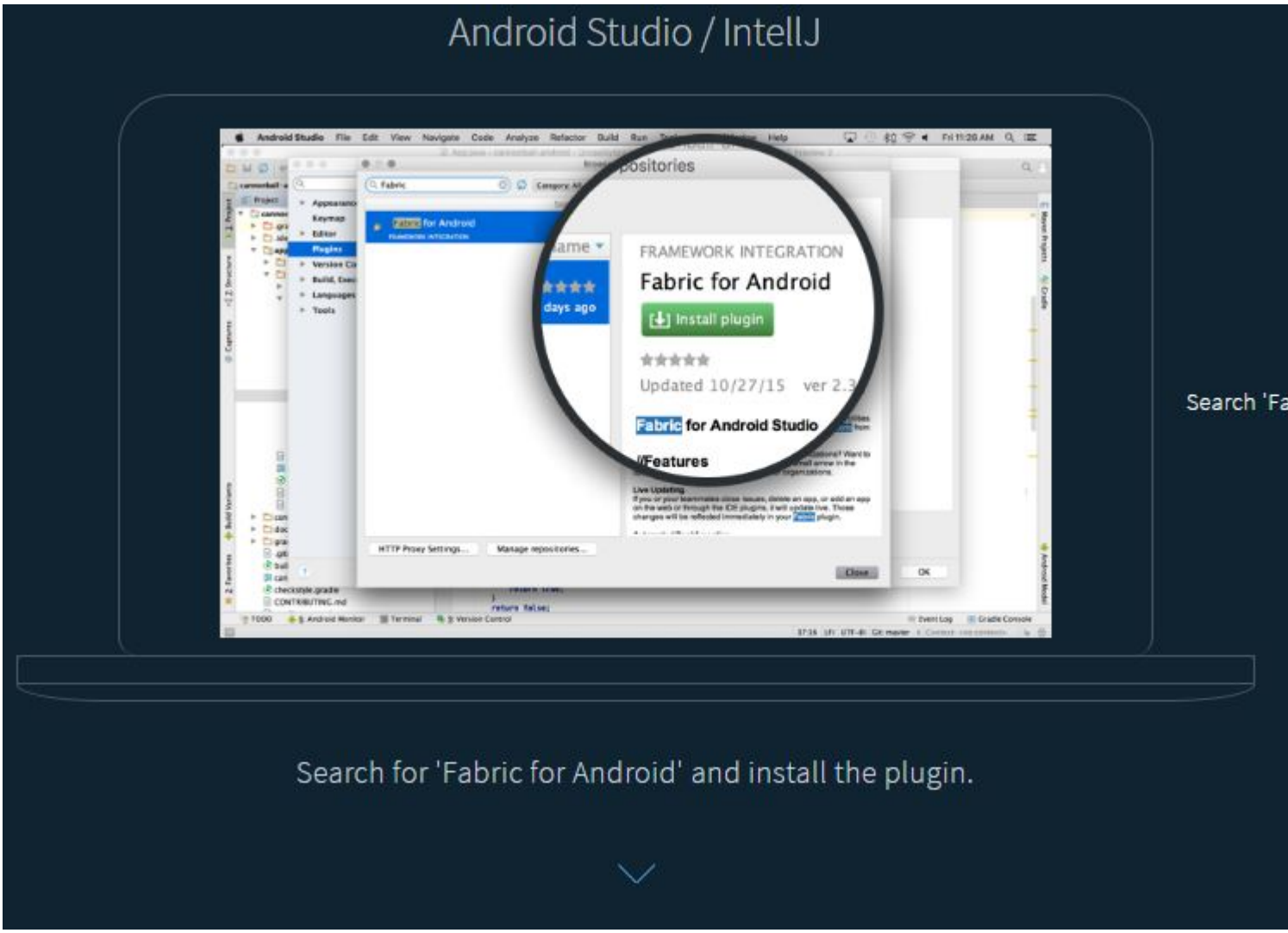
```
public class MainActivity extends ActionBarActivity {

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

      //Init the KIT
      Fabric.with(this, new Crashlytics());

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

Шаг 4: Создайте проект. Чтобы создать и запустить:



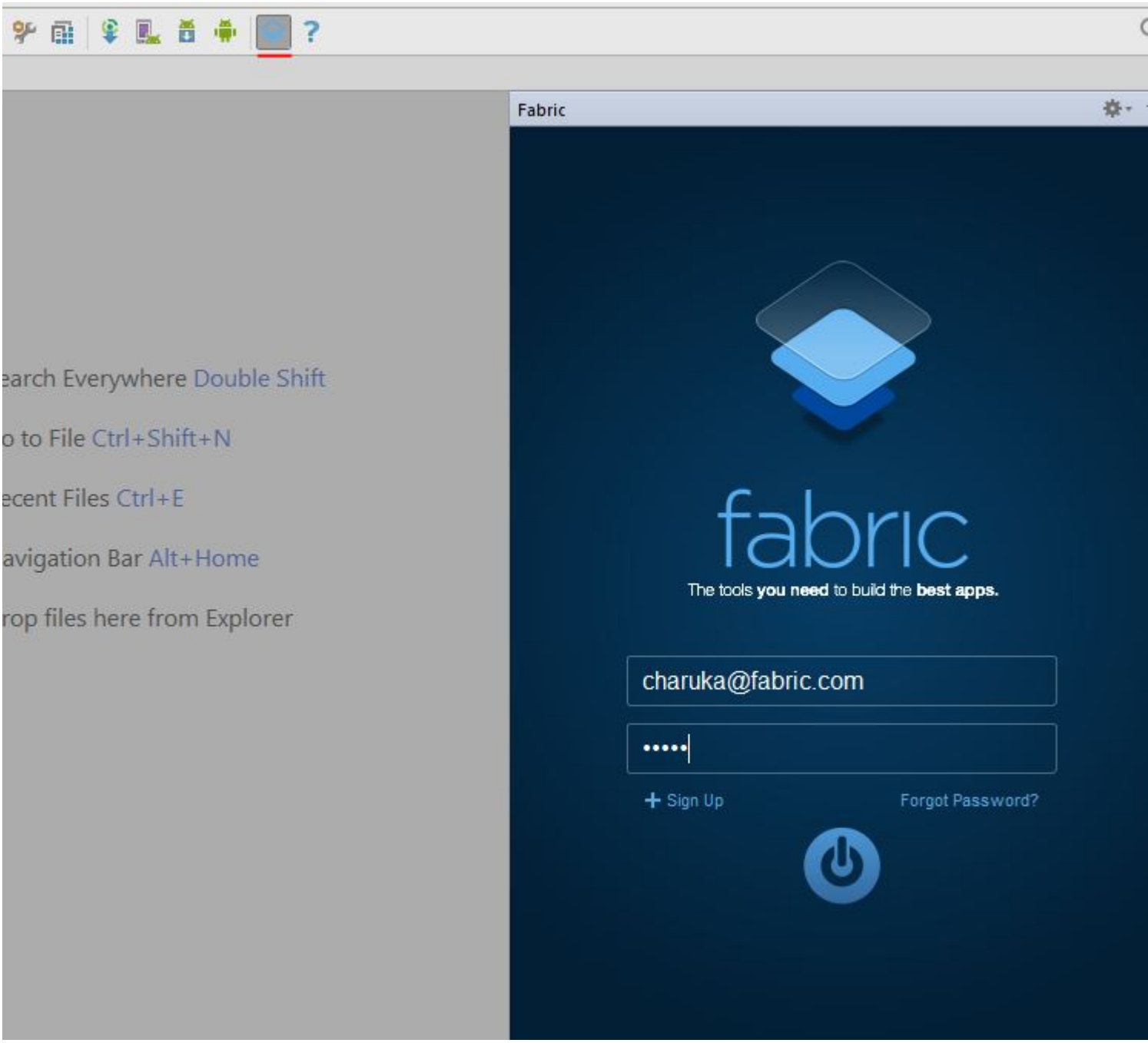
Использование плагина IDE Fabric

Наборы можно установить с помощью плагина Fabric IDE для Android Studio или IntelliJ по этой ссылке.



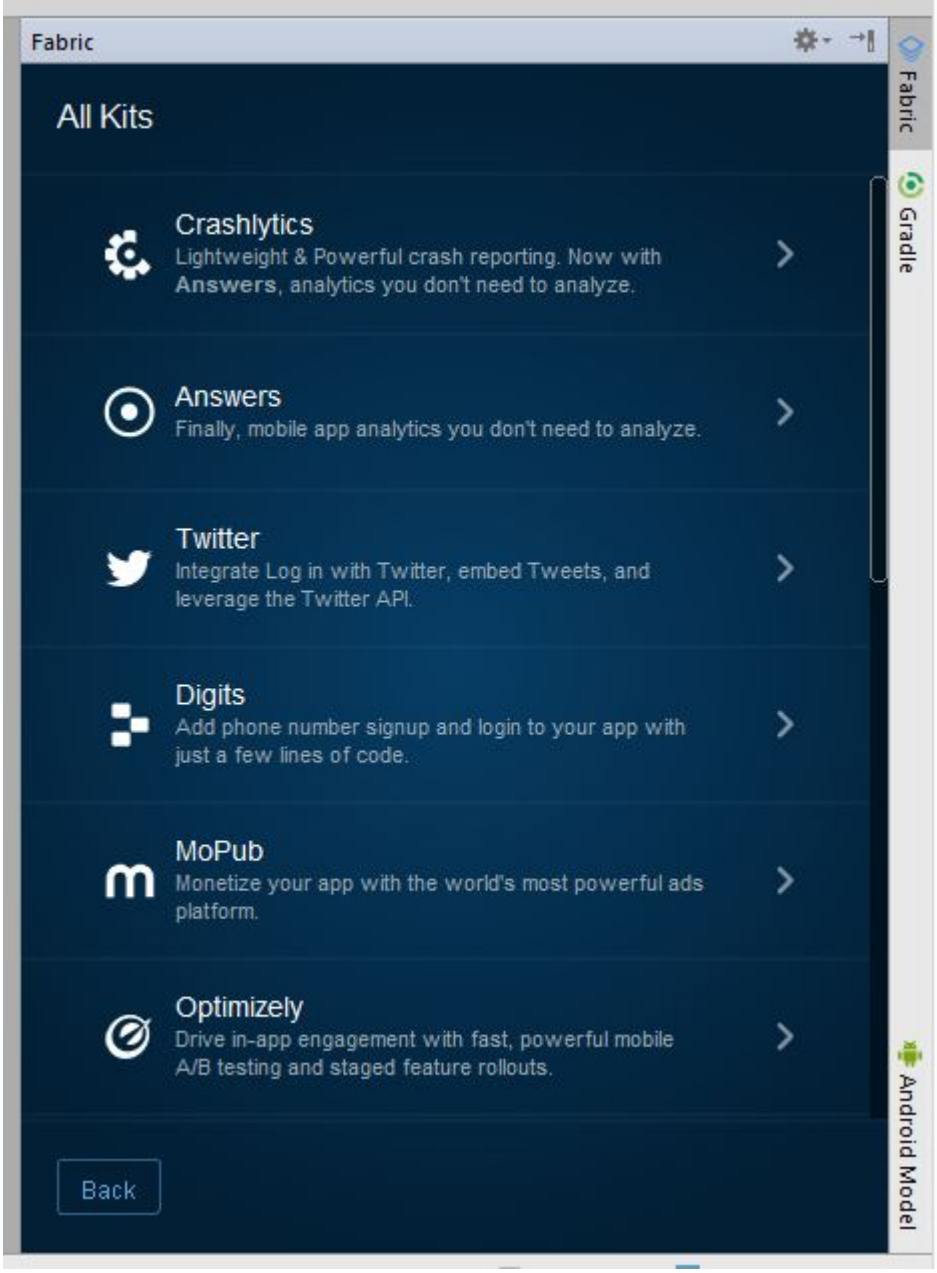
После установки плагина **перезапустите** Android Studio и **войдите в** свою учетную запись с помощью **Android Studio** .

(короткий ключ> `CTRL + L`)

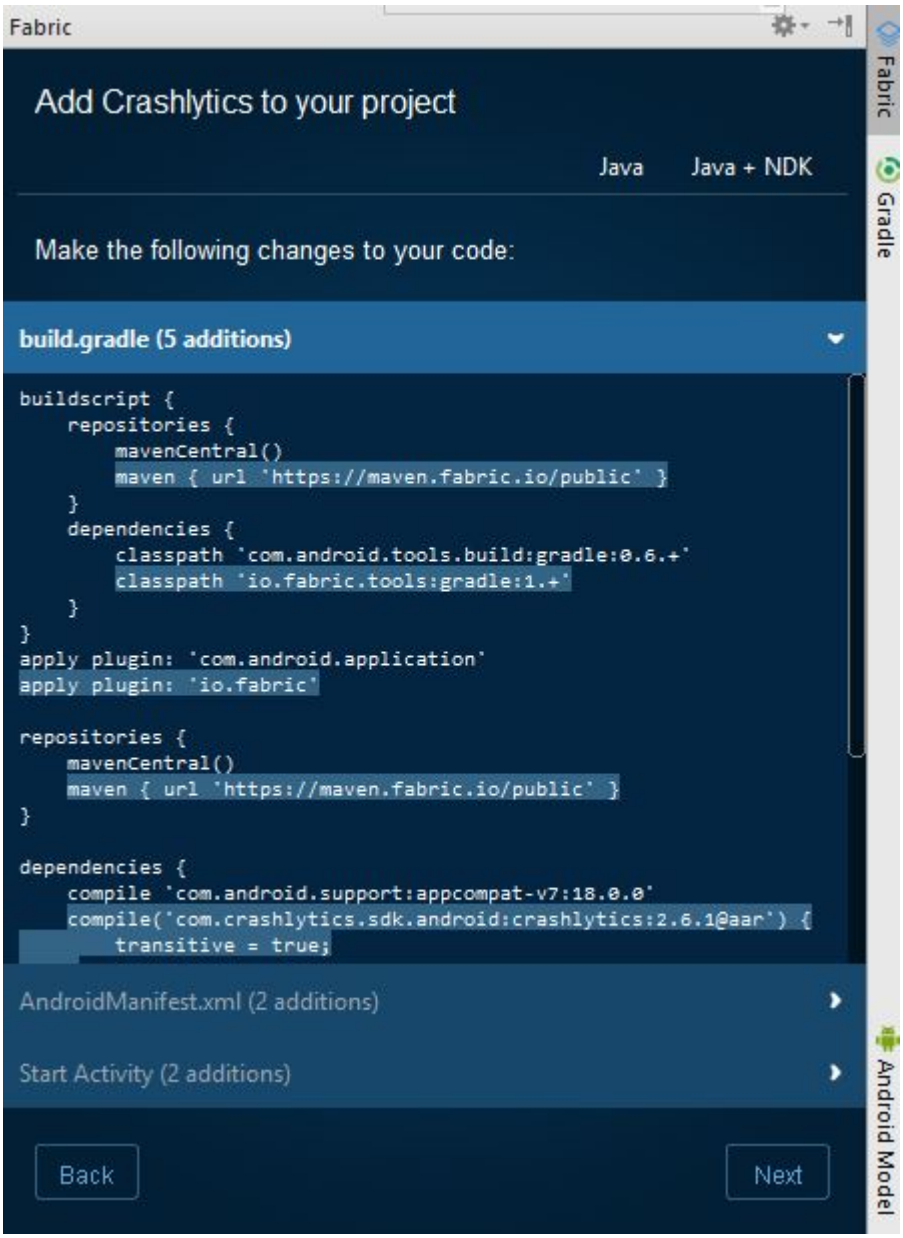


Затем он покажет проекты, которые у вас есть / проект, который вы открыли, выберите тот, который вам нужен, и нажмите «Далее» далее.

Выберите набор, который вы хотите добавить, для его примера это **Crashlytics** :



Затем нажмите « `Install` . Вам не нужно добавлять его вручную на этот раз, как над **плагином gradle** , вместо этого он будет **создан** для вас.



Готово!

Отчет о сбоях с ACRA

Шаг 1: Добавьте зависимость последнего ACRA AAR к вашему градиенту приложения (build.gradle).

Шаг 2. В вашем классе приложения (класс, который расширяет приложение, если он не создается). Добавьте аннотацию `@ReportsCrashes` и переопределите метод `attachBaseContext()` .

Шаг 3. Инициализация класса ACRA в вашем классе приложения.

```
@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);
            }
    }
```

Если AUTH_USERNAME и AUTH_PASSWORD являются учетными данными ваших желаемых бэкэндов .

Шаг 4: Определите класс приложения в AndroidManifest.xml

```
<application
       android:name=".MyApplication">
       <service></service>
       <activity></activity>
       <receiver></receiver>
</application>
```

Шаг 5: Убедитесь, что у вас есть разрешение на `internet` к `internet` для получения отчета из разбитого приложения

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

Если вы хотите отправить бесшумный отчет в бэкэнд, просто используйте метод ниже для его достижения.

```
ACRA.getErrorReporter().handleSilentException(e);
```

Принудительное испытание с повреждением

Добавьте кнопку, которую вы можете нажать, чтобы вызвать сбой. Вставьте этот код в свой макет, где вы хотите, чтобы кнопка появилась.

```
<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" />
```

Выбросить исключение RuntimeException

```
public void forceCrash(View view) {
    throw new RuntimeException("This is a crash");
}
```

Запустите приложение и нажмите новую кнопку, чтобы вызвать сбой. Через минуту или две вы сможете увидеть крах на панели управления Crashlytics, а также получите почту.

Сбой при сбое с использованием Шерлока

Шерлок фиксирует все ваши аварии и сообщает о них как уведомление. Когда вы нажимаете на уведомление, он открывает активность со всеми деталями сбоя вместе с информацией об устройстве и приложении

Как интегрировать Шерлок с вашим приложением?

Вам просто нужно добавить Шерлока в зависимость от градиента в вашем проекте.

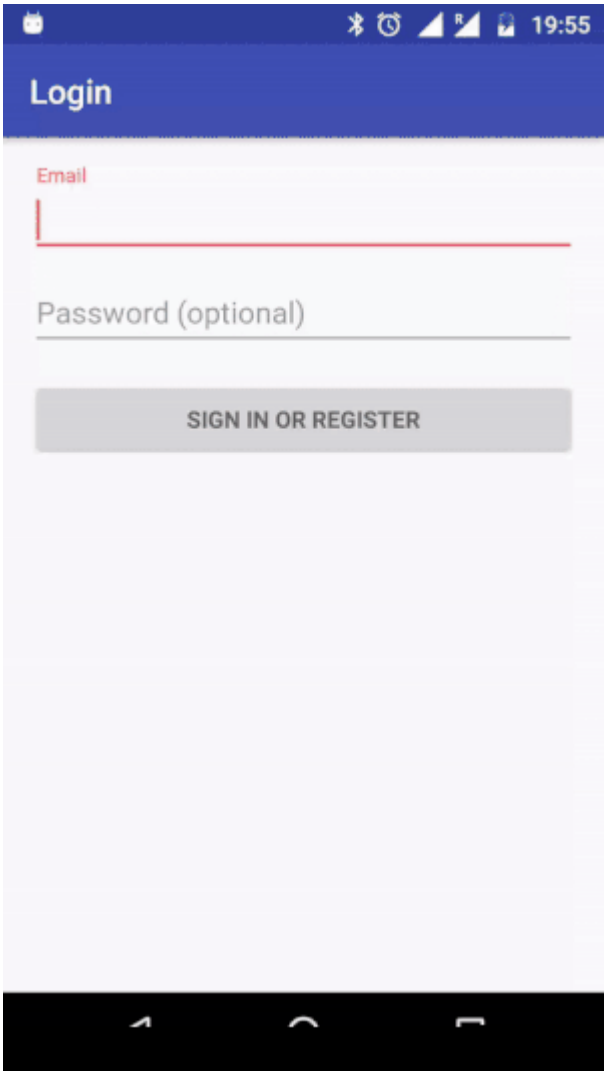
```
dependencies {
    compile('com.github.ajitsing:sherlock:1.0.1@aar') {
        transitive = true
    }
}
```

После синхронизации вашей студии Android инициализируйте Шерлок в своем классе 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);
  }
}
```

Это все, что вам нужно. Кроме того, Шерлок делает гораздо больше, чем просто сообщение об аварии. Чтобы проверить все его функции, взгляните на эту статью .

демонстрация



Прочитайте Инструменты отчетов о сбоях онлайн: https://riptutorial.com/ru/android/topic/3871/инструменты-отчетов-о-сбоях

глава 138: Интеграция OpenCV в Android Studio

замечания

Библиотеки Open CV можно найти в Интернете с помощью поисковой системы.

Готча :

- Если вы снижаете целевую платформу ниже KitKat, некоторые из библиотек OpenCV больше не будут работать, в частности классы, связанные с *org.opencv.android.Camera2Renderer* и другие связанные классы. Вы можете обойти это, просто удалив соответствующие файлы OpenCV .java.
- Если вы поднимете свою целевую платформу на Lollipop или выше моего примера загрузки файла, возможно, не сработает, потому что использование абсолютных путей к файлу неодобрительно. Поэтому вам может потребоваться изменить пример загрузки файла из галереи или где-то еще. Есть множество примеров, плавающих вокруг.

Examples

инструкции

Протестировано с AS v1.4.1, но должно работать и с более новыми версиями.

1. Создайте новый проект Android Studio с помощью мастера проекта (Меню: / Файл / Новый проект):

   - Назовите это « **cvtest1** »
   - Форм-фактор: **API 19, Android 4.4 (KitKat)**
   - **Пустое действие под** названием **MainActivity**

   У вас должен быть каталог *cvtest1, в* котором хранится этот проект. (строка заголовка студии Android показывает, где cvtest1 при открытии проекта)

2. Убедитесь, что приложение работает правильно. Попробуйте изменить что-то вроде текста «Hello World», чтобы подтвердить, что цикл сборки / тестирования подходит для вас. (Я тестирую эмулятор устройства API 19).

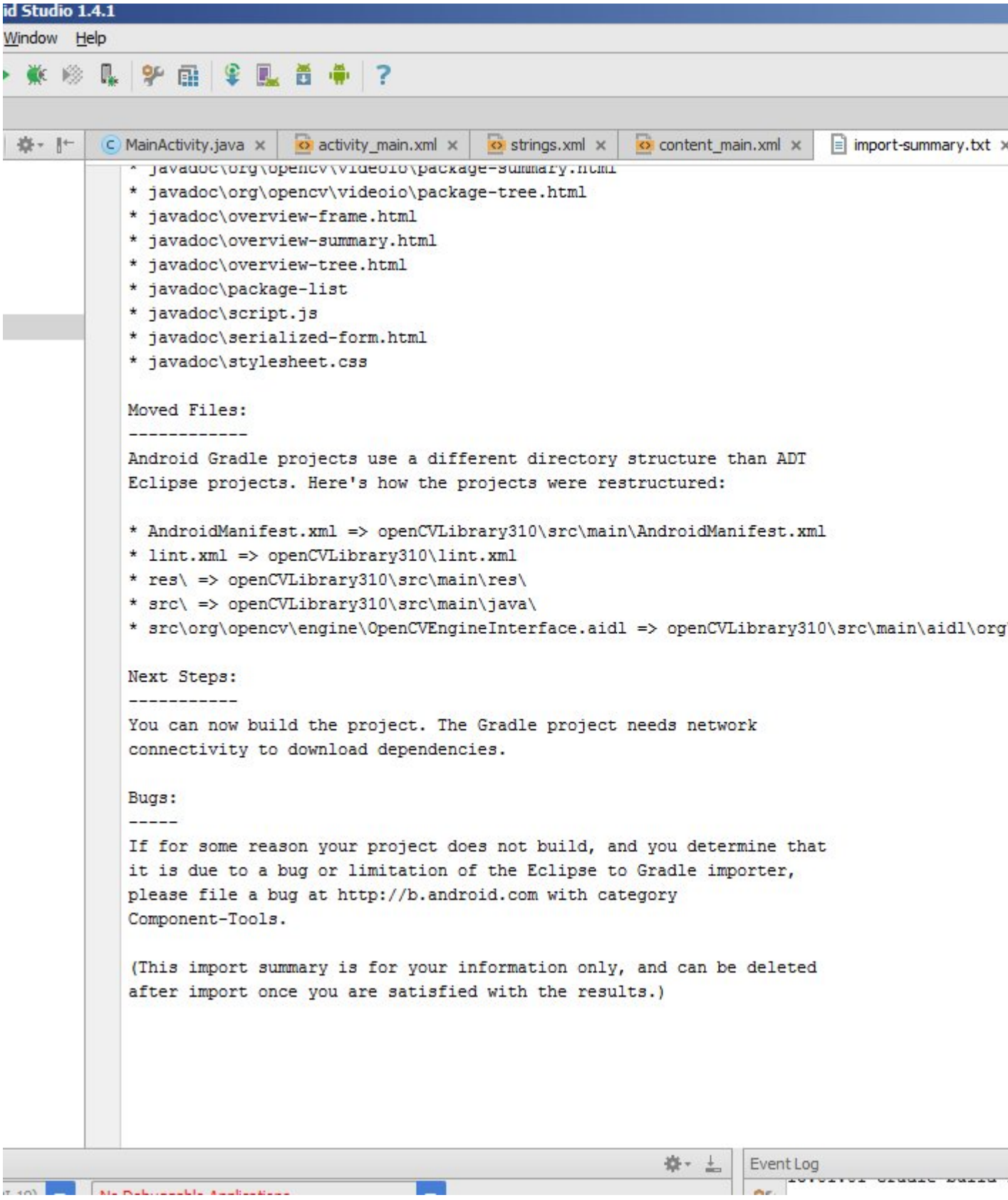
3. Загрузите пакет OpenCV для Android v3.1.0 и разархивируйте его в каком-нибудь временном каталоге. (Убедитесь, что это пакет специально для Android, а не только для пакета OpenCV для Java.) Я назову этот каталог « **unzip-dir** ». Ниже **unzip-dir** вы

должны иметь каталог **sdk / native / libs** с подкаталогами, которые начинаются с такие вещи, как **рука** ..., **mips** ... и **x86** ... (по одному для каждого типа «архитектуры» Android работает)

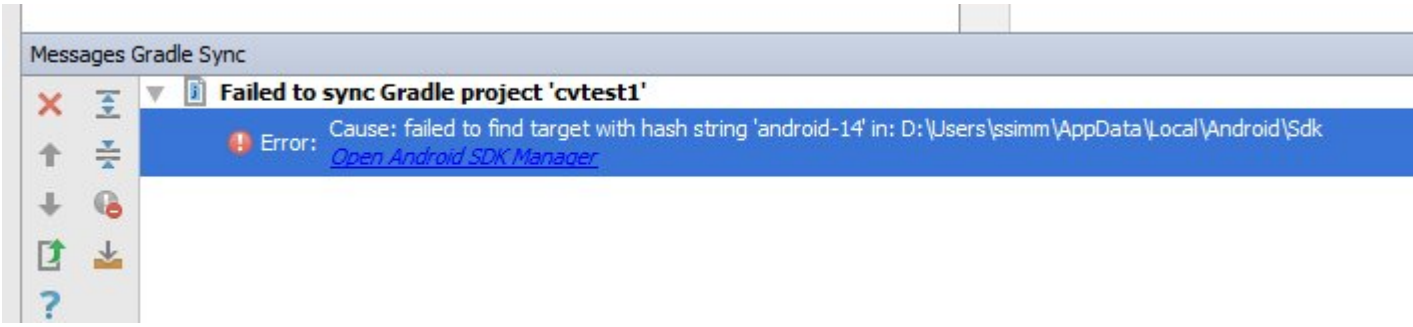
4. Из Android Studio импортируйте OpenCV в свой проект как модуль: **Menu: / File / New / Import_Module** :

    - Исходный каталог: **{unzip-dir} / sdk / java**
    - Имя модуля: Студия Android автоматически заполняет это поле с помощью **openCVLibrary310** (точное имя, вероятно, не имеет значения, но мы пойдем с этим).
    - Нажмите **дальше** . Вы получаете экран с тремя флажками и вопросами о банках, библиотеках и опциях импорта. Все три должны быть проверены. Нажмите « **Готово».**

    Android Studio начинает импортировать модуль, и вам будет показан файл **import-summary.txt, в** котором есть список того, что не было импортировано (в основном файлы javadoc) и другие фрагменты информации.



Но вы также получаете сообщение об ошибке, в котором **не удалось найти цель с хеш-строкой «андроид-14» ....** Это происходит из-за того, что файл build.gradle в загруженном zip-файле OpenCV говорит, что он компилируется с использованием API-интерфейсов Android версии 14, который по умолчанию у вас нет в Android Studio v1.4.1.

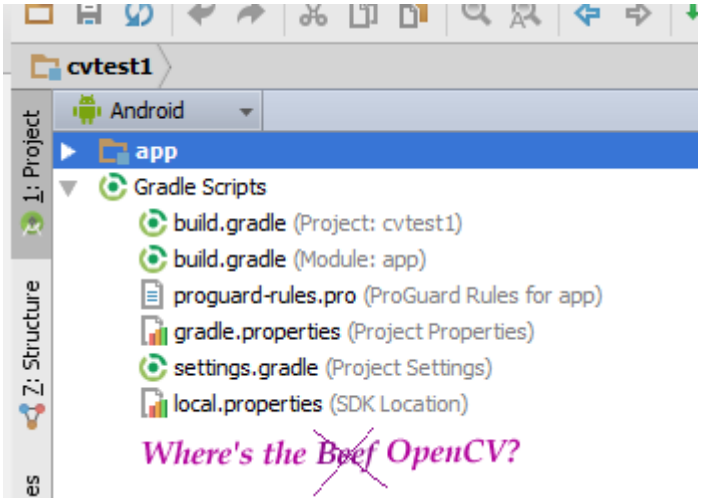


5. Откройте диалог структуры проекта (**меню:** / **File / Project_Structure**). Выберите модуль «приложение», перейдите на вкладку « **Зависимости** » и добавьте **: openCVLibrary310** в качестве зависимости от модуля. Когда вы выбираете **Add / Module_Dependency,** он должен появиться в списке модулей, которые вы можете добавить. Теперь он будет отображаться как зависимость, но вы получите еще несколько ошибок **can not-find-android-14** в журнале событий.

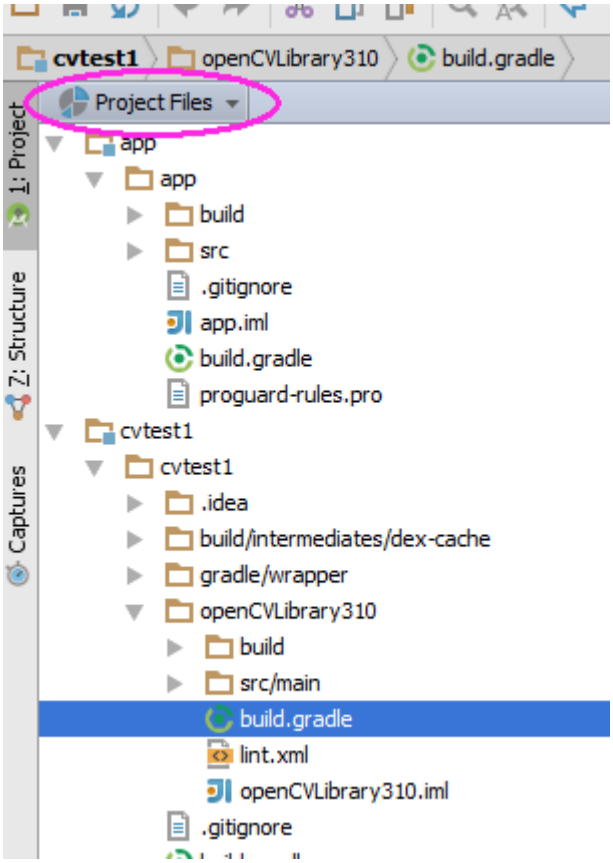
6. Посмотрите файл **build.gradle** для своего модуля приложения. В проекте Android есть несколько файлов build.gradle. Тот, который вы хотите, находится в **каталоге cvtest1 / app,** и из представления проекта он выглядит как **build.gradle (Module: app)** . Обратите внимание на значения этих четырех полей:

   - compileSDKVersion (мой говорит 23)
   - buildToolsVersion (мой говорит 23.0.2)
   - minSdkVersion (мой говорит 19)
   - targetSdkVersion (мой говорит 23)

7. Теперь у вашего проекта есть **каталог cvtest1 / OpenCVLibrary310,** но он не отображается в представлении проекта:

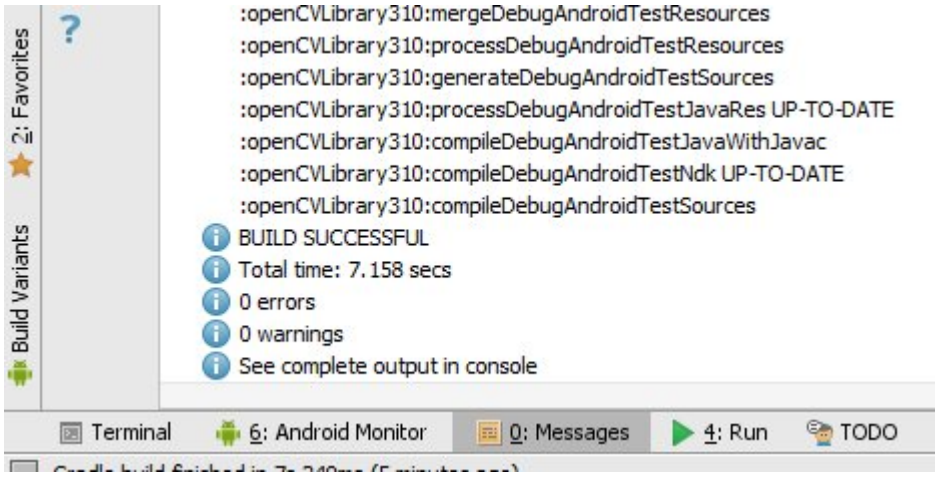


Используйте другой инструмент, например, любой файловый менеджер, и перейдите в этот каталог. Вы также можете переключить представление проекта с **Android** на **файлы проекта,** и вы можете найти этот каталог, как показано на этом скриншоте:



Внутри есть еще **один** файл **build.gradle** (он показан в приведенном выше **снимке** экрана). Обновите этот файл четырьмя значениями с шага 6.

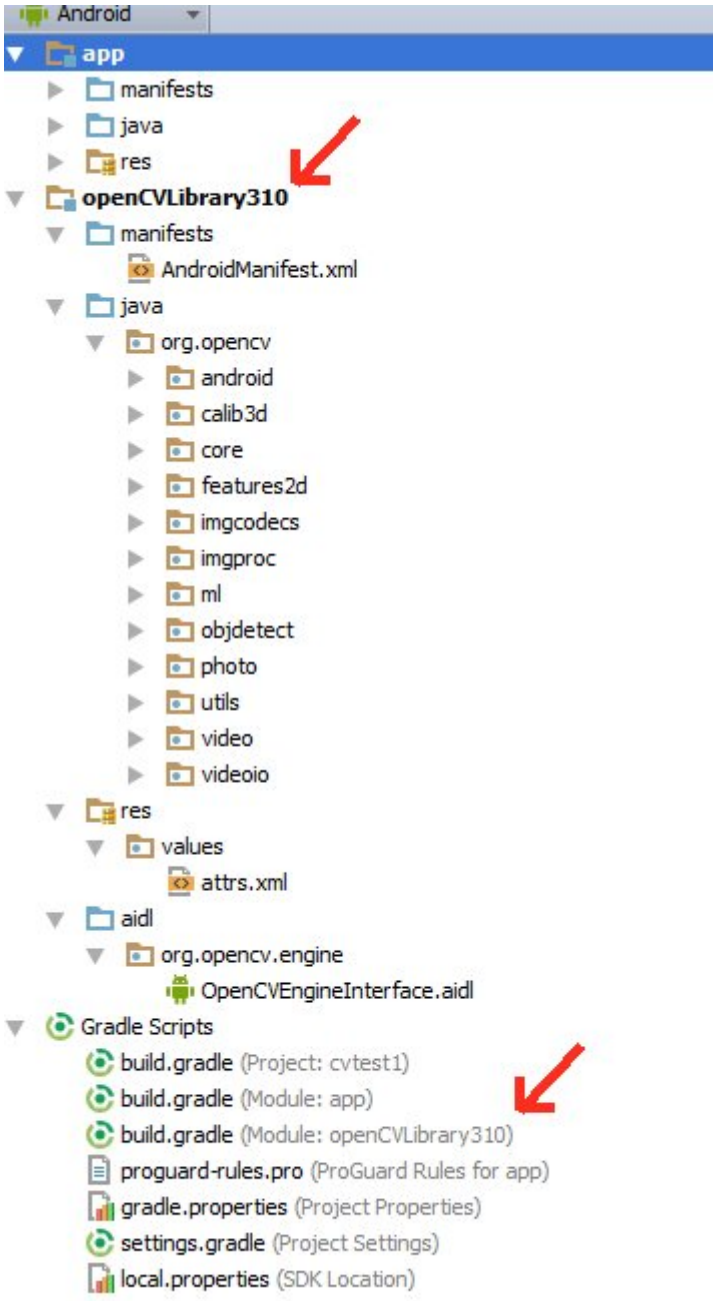
8. Повторно протестируйте свой проект, а затем очистите его / восстановите. (**Меню:** / **Build / Clean_Project**) Он должен очищать и строить без ошибок, и вы должны увидеть много ссылок на **: openCVLibrary310** на экране **0: Сообщения** .



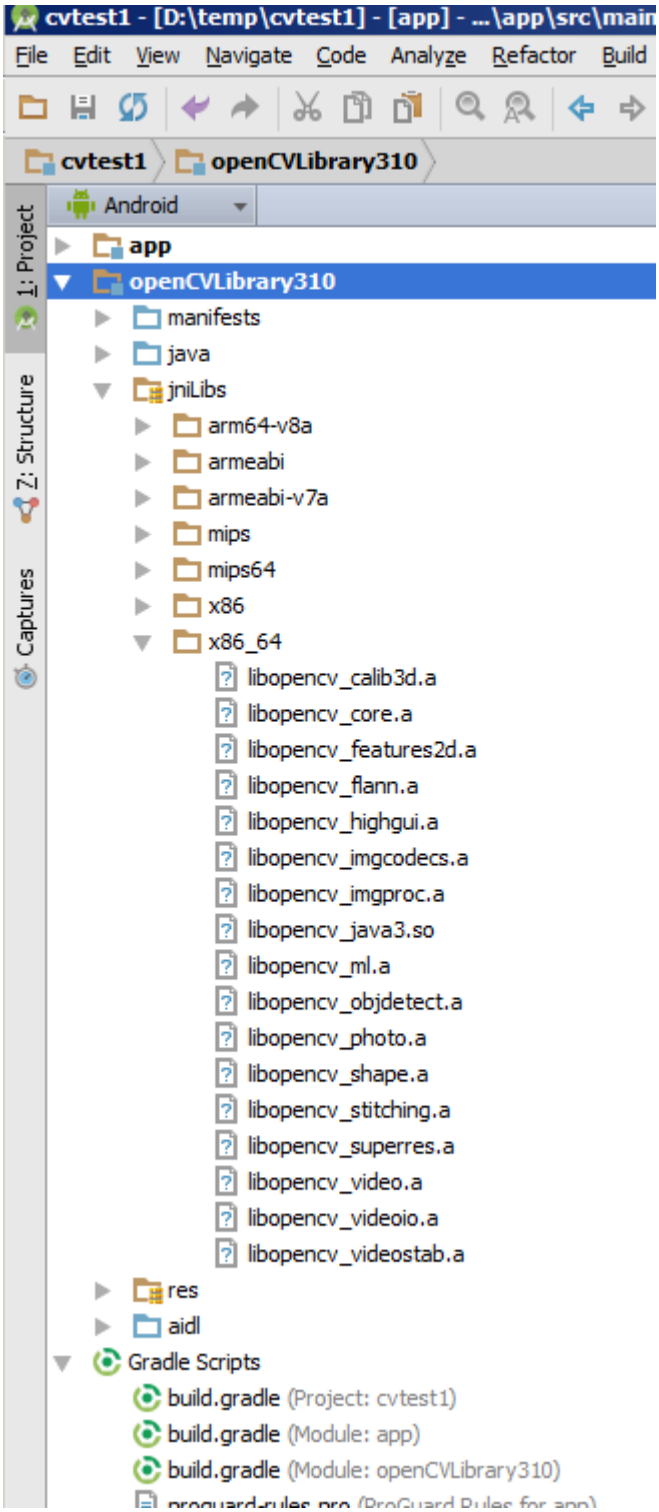
На данный момент модуль должен отображаться в иерархии проекта как **openCVLibrary310** , как и **приложение** . (Обратите внимание, что в этом маленьком раскрывающемся меню я переключился с **Project View** на **Android View**). Вы также должны увидеть дополнительный файл **build.gradle в** разделе «Gradle Scripts», но я обнаружил, что интерфейс Android Studio немного глючит, и иногда он не делает этого сразу. Поэтому попробуйте выполнить повторную синхронизацию, очистку и

даже перезагрузку Android Studio.

Вы должны увидеть модуль openCVLibrary310 со всеми функциями OpenCV под java, как на этом скриншоте:



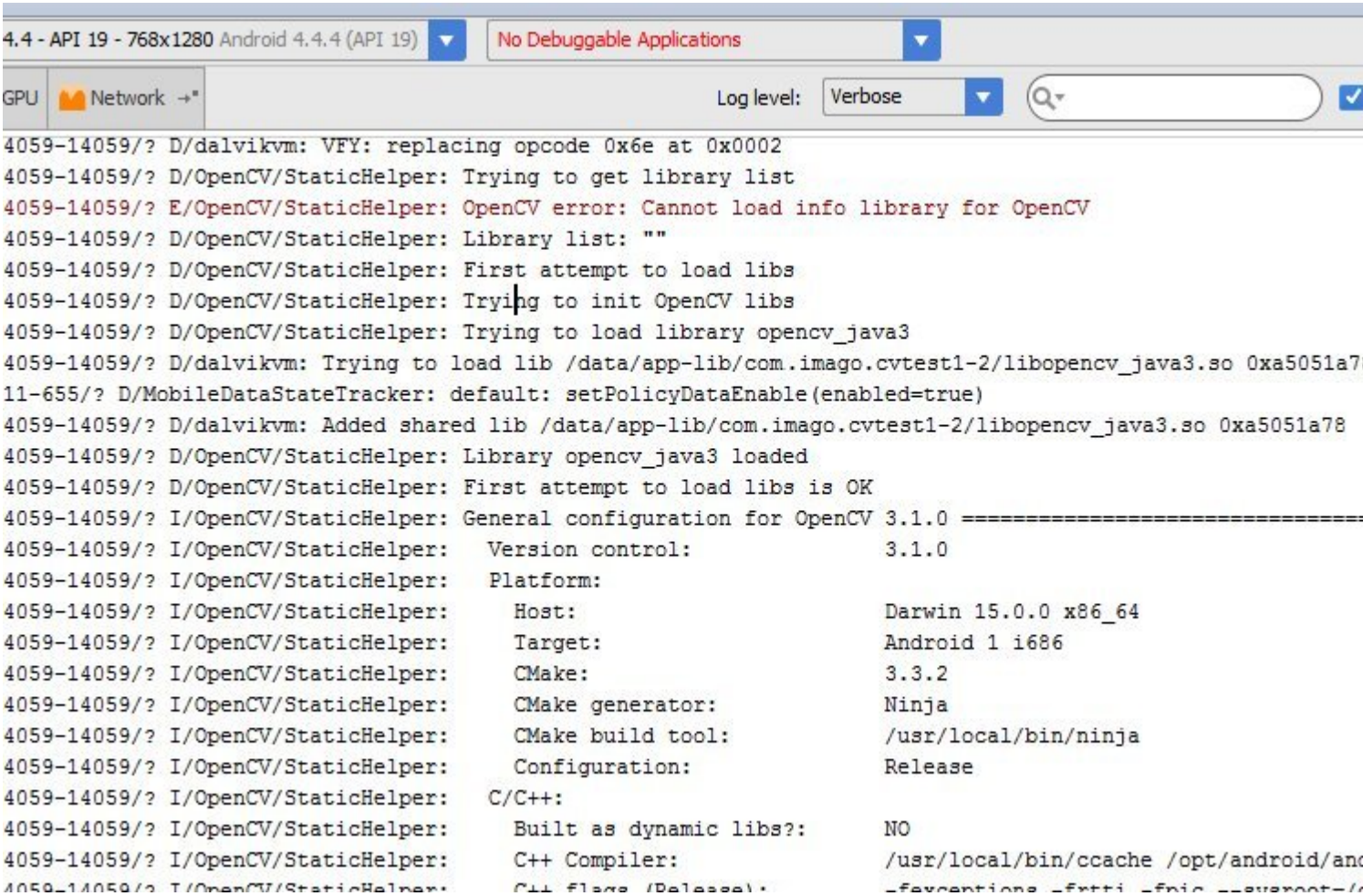
9. Скопируйте каталог **{unzip-dir} / sdk / native / libs** (и все под ним) в проект Android, в **cvtest1 / OpenCVLibrary310 / src / main /** , а затем переименуйте свою копию из **libs** в **jniLibs** . Теперь у вас должен быть **каталог cvtest1 / OpenCVLibrary310 / src / main / jniLibs** . Повторно протестируйте проект, и этот каталог теперь должен появиться в представлении проекта в **openCVLibrary310** .



10. Перейдите в метод *onCreate MainActivity.java* и добавьте этот код:

```
if (!OpenCVLoader.initDebug()) {
        Log.e(this.getClass().getSimpleName(), "  OpenCVLoader.initDebug(), not working.");
    } else {
        Log.d(this.getClass().getSimpleName(), "  OpenCVLoader.initDebug(), working.");
    }
```

Затем запустите приложение. В Android Monitor вы должны увидеть такие строки:



(Я не знаю, почему эта строка с сообщением об ошибке)

11. Теперь попробуйте использовать некоторый код openCV. В приведенном ниже примере я скопировал файл .jpg в каталог кэша приложения cvtest1 на эмуляторе Android. Приведенный ниже код загружает это изображение, запускает алгоритм обнаружения байтового края и затем записывает результаты обратно в .png-файл в том же каталоге.

```
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. Запустите приложение. Ваш эмулятор должен создать черно-белое «краевое» изображение. Вы можете использовать Android Device Monitor для получения вывода или записи активности для его отображения.

Прочитайте Интеграция OpenCV в Android Studio онлайн:
https://riptutorial.com/ru/android/topic/7068/интеграция-opencv-в-android-studio

глава 139: Интеграция входа в Google

Синтаксис

- newInstance () - создание одного экземпляра Google Helper
- initGoogleSignIn () - для инициализации входа в Google
- getGoogleAccountDetails () - Чтобы войти в учетную запись
- signOut () - Выйти из системы
- getGoogleClient () - для использования GoogleApiClient

параметры

параметр	подробность
ТЕГ	Строка, используемая при регистрации
GoogleSignInHelper	Статическая ссылка для помощника
AppCompatActivity	Ссылка на активность
GoogleApiClient	Ссылка на GoogleAPIClient
RC_SIGN_IN	Целое число представляет собой константу результата деятельности
isLoggingOut	Логическое значение для проверки выполнения задачи выхода из системы

Examples

Вход в Google со вспомогательным классом

Добавьте ниже в свой `build.gradle` из тега `android` :

```
// Apply plug-in to app.
apply plugin: 'com.google.gms.google-services'
```

Добавьте ниже вспомогательный класс в ваш пакет утилиты:

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

Добавьте ниже код в свой `OnActivityResult` в файле активности:

```
 // [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]
```

Прочитайте Интеграция входа в Google онлайн: https://riptutorial.com/ru/android/topic/2837/интеграция-входа-в-google

глава 140: Интеграция с подписью Google на Android

Вступление

Этот раздел основан на том, как интегрировать учетную запись google, On android apps

Examples

Интеграция google Auth в ваш проект. (Получить файл конфигурации)

Сначала получите конфигурационный файл для входа с

Открыть ссылку ниже

[https://developers.google.com/identity/sign-in/android/start-integrating][1]

нажмите, чтобы получить файл конфигурации

- Введите имя приложения и имя пакета и нажмите «Выбрать» и настроить службы
- предоставить SHA1 Включить Google SIGNIN и сгенерировать файлы конфигурации

Загрузите файл конфигурации и поместите файл в приложение / папку вашего проекта

1. Добавьте зависимость к вашему проекту build.gradle:

   classpath 'com.google.gms: google-services: 3.0.0'

2. Добавьте плагин на уровень сборки: (внизу)

   применить плагин: 'com.google.gms.google-services'

3. добавьте эту зависимость в файл приложения gradle

   dependties {compile 'com.google.android.gms: play-services-auth: 9.8.0'}

Внедрение кода Google SignIn

- В методе onCreate вашей учетной записи настройте Google Sign-In, чтобы запросить пользовательские данные, необходимые вашему приложению.

```
 GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestEmail()
        .build();
```

- создать объект GoogleApiClient с доступом к API входа в Google и указанными вами параметрами.

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
        .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
        .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
        .build();
```

- Теперь, когда пользователь нажимает кнопку входа в систему Google, вызывайте эту функцию.

```
 private void signIn() {
 Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
 startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

- реализуйте OnActivityResult, чтобы получить ответ.

```
 @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);
    }
}
```

- Последний шаг Обработать Результат и получить данные пользователя

```
 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);
 }
}
```

Прочитайте Интеграция с подписью Google на Android онлайн:
https://riptutorial.com/ru/android/topic/9960/интеграция-с-подписью-google-на-android

глава 141: Интеграция шлюза Android Paypal

замечания

Paypal предоставляет нам собственную библиотеку для оплаты, поэтому теперь она очень надежна и проста в применении в нашем приложении. Ниже приведен важный шаг.

Examples

Настройка PayPal в вашем коде Android

1) Сначала зайдите на веб-сайт разработчика Paypal Developer и создайте приложение.

2) Теперь откройте файл манифеста и дайте ниже разрешения

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

3) И некоторые необходимые действия и услуги -

```
 <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) Откройте класс Activity и настройте Configuration для вашего приложения,

```
//set the environment for production/sandbox/no netowrk
 private static final String CONFIG_ENVIRONMENT = PayPalConfiguration.ENVIRONMENT_PRODUCTION;
```

5) Теперь установите идентификатор клиента из учетной записи разработчика Paypal - закрытый статический окончательный String CONFIG_CLIENT_ID = «НАПРАВЛЯТЬ ИДЕНТИФИКАТОР КЛИЕНТА»; 6) Внутри метода onCreate вызов службы Paypal - намерение намерения = новый намерение (это, PayPalService.class); intent.putExtra (PayPalService.EXTRA_PAYPAL_CONFIGURATION, config); StartService (намерение);

7) Теперь вы готовы сделать платеж только при нажатии кнопки на платежную платежную операцию -

```
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) И, наконец, из onActivityResult получите ответ на платеж -

```
 @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.");
                   }
               }


           }
```

Прочитайте Интеграция шлюза Android Paypal онлайн:
https://riptutorial.com/ru/android/topic/5895/интеграция-шлюза-android-paypal

глава 142: Интеллектуальная карточка

Examples

Смарт-карта отправляет и принимает

Для соединения, вот фрагмент, который поможет вам понять:

```
//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);
}
```

Теперь вам нужно понять, что в java сообщение происходит с использованием пакета javax.smarcard, который недоступен для Android, поэтому взгляните сюда, чтобы получить представление о том, как вы можете общаться или отправлять / получать APDU (команда смарт-карты).

Теперь, как сказано в ответе, упомянутом выше

Вы не можете просто отправить команду APDU (команда смарт-карты) поверх конечной точки массового вывода и ожидать получения APDU ответа в конечной точке массового ввода. Для получения конечных точек см. Фрагмент кода ниже:

```
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;
                }
            }
        }
    }
```

Теперь у вас есть конечные точки для массового ввода и массового вывода для отправки и приема блоков APDU и блоков ответа APDU:

Для отправки команд см. Фрагмент кода ниже:

```
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));
    }
}
```

И для получения / чтения ответа см. Фрагмент кода ниже:

```
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;
}
```

Теперь, если вы видите этот ответ, первая команда, которую нужно отправить, это:

PC_to_RDR_IccPowerOn для активации карты.

которую вы можете создать, прочитав раздел 6.1.1 документа Doc для устройств USB Device.

Теперь давайте возьмем пример этой команды, такой как здесь: `62000000000000000000` Как вы можете отправить это:

```
write(connection, epOut, "62000000000000000000");
```

Теперь, после того как вы успешно отправили команду APDU, вы можете прочитать ответ, используя:

```
read(connection, epIn);
```

И получить что-то вроде

```
80 18000000 00 00 00 00 00 3BBF11008131FE45455041000000000000000000000000F1
```

Теперь ответ, полученный в коде здесь, будет в переменной `result` метода `read()` из кода

Прочитайте Интеллектуальная карточка онлайн: https://riptutorial.com/ru/android/topic/10945/интеллектуальная-карточка

глава 143: Интерактивный пользовательский интерфейс с UIAutomator

Синтаксис

- Инструментарий getInstrumentation ()
- UIDevice UiDevice.getInstance (контрольно-измерительная аппаратура)
- boolean UIDevice.pressHome ()
- boolean UIDevice.pressBack ()
- boolean UIDevice.pressRecentApps ()
- void UIDevice.wakeUp ()
- boolean UIDevice.swipe (int startX, int startY, int endX, int endY, int steps)
- boolean UIDevice.drag (int startX, int startY, int endX, int endY, int steps)
- UIObject2 UIDevice.findObject (By.desc (String contentDesc))
- boolean UIObject2.click ()

замечания

UIAutomator особенно хороши для тестирования пользовательских историй. У вас возникают проблемы, если элементы представления не имеют ни уникального *идентификатора ресурса,* ни *контента-desc* . В большинстве случаев есть способ закончить тест в любом случае, что занимает много времени. Если вы можете влиять на код своего приложения, UIAutomator может быть вашим инструментом тестирования.

Examples

Подготовьте свой проект и напишите первый тест UIAutomator

Добавьте необходимые библиотеки в раздел зависимостей build.gradle вашего 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'
}
```

Обратите внимание, что, конечно, версии могут отличаться в среднем времени.

После этой синхронизации с изменениями.

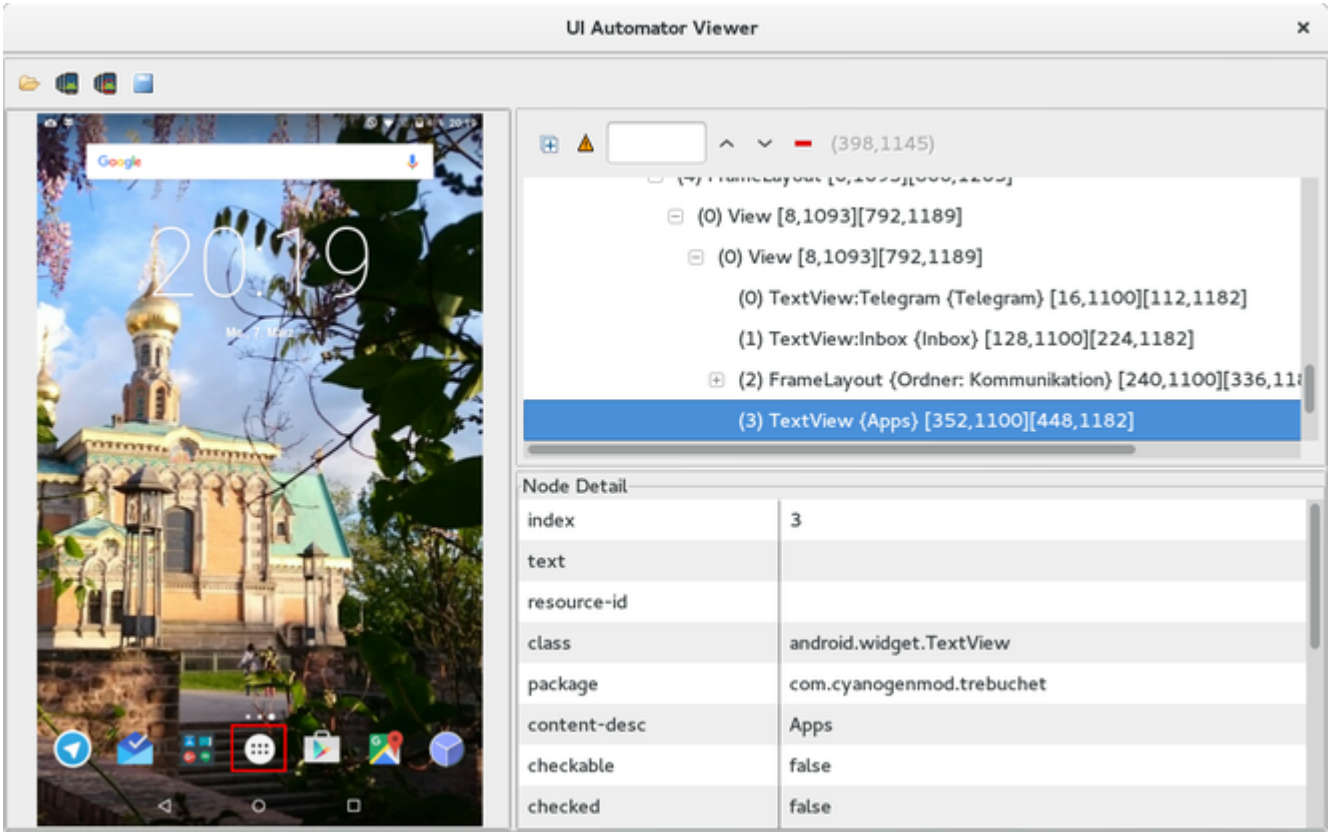
Затем добавьте новый класс Java внутри папки androidTest:

```
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();
  }
}
```

Сделав правый щелчок на вкладке класса и «Запустить« InterAppTest », выполняется этот тест.

Написание более сложных тестов с использованием UIAutomatorViewer

Чтобы разрешить писать более сложные UI-тесты, необходим *UIAutomatorViewer* . Инструмент, расположенный в */ tools /,* делает полноэкранный снимок экрана, включая макеты отображаемых в настоящее время видов. См. Последующую картинку, чтобы получить представление о том, что показано:



Для тестов UI мы ищем *идентификатор ресурса* , *контент-desc* или что-то еще, чтобы идентифицировать представление и использовать его в наших тестах.

Uiautomatorviewer выполняется через терминал.

Если мы сейчас, например, хотим нажать кнопку «Приложения», а затем открыть какое-то приложение и проведите по экрану, вот как выглядит метод тестирования:

```
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)
}
```

Создание тестового набора тестов UIAutomator

Сведение тестов UIAutomator вместе с набором тестов - это быстро:

```
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 {}
```

Выполните аналогичную процедуру, щелкнув правой кнопкой мыши и запустите пакет.

Прочитайте Интерактивный пользовательский интерфейс с UIAutomator онлайн: https://riptutorial.com/ru/android/topic/6249/интерактивный-пользовательский-интерфейс-с-uiautomator

глава 144: Интернационализация и локализация (I18N и L10N)

Вступление

Интернационализация (i18n) и локализация (L10n) используются для адаптации программного обеспечения в соответствии с различиями в языках, региональных различиях и целевой аудитории.

Интернационализация: процесс планирования будущей локализации, т. Е. Гибкость разработки программного обеспечения в той мере, в какой он может корректировать и адаптироваться к будущим усилиям по локализации.

Локализация: процесс адаптации программного обеспечения к определенному региону / стране / рынку (локали).

замечания

Чтобы протестировать устройство для локализации, устройство или эмулятор можно перезагрузить в определенной локали, используя `adb` следующим образом:

1. Запустите adb с помощью команды: `adb shell`
2. Выполните следующую команду в командной строке adb: `setprop persist.sys.locale [BCP-47 language tag];stop;sleep 5;start` где [тег языка BCP-47] - это код, специфичный для языка, как описано здесь: коды BCP47

например, для проверки японской локализации в приложении, используйте команду:
`setprop persist.sys.locale ja-JP;stop;sleep 5;start`

Examples

Планирование локализации: включить поддержку RTL в манифесте

Поддержка RTL (справа налево) является важной частью планирования для i18n и L10n. В отличие от английского языка, который написан слева направо, многие языки, такие как арабский, японский, иврит и т. Д., Написаны справа налево. Чтобы обратиться к более глобальной аудитории, рекомендуется планировать свои макеты для поддержки этого языка с самого начала проекта, так что добавление локализации стало проще позже.

Поддержка RTL может быть включена в Android-приложении, добавив тег `supportsRtl` в `AndroidManifest` , например:

```
<application
    ...
    android:supportsRtl="true"
    ...>
...
</application>
```

Планирование локализации: добавление поддержки RTL в макетах

Начиная SDK 17 (Android 4.2), поддержка RTL была добавлена в макеты Android и является важной частью локализации. Вперёд, `left/right` нотация в макетах должна быть заменена нотой `start/end` . Если, однако, ваш проект имеет значение `minSdk` меньше `17` , то в макетах следует использовать как `left/right` и `start/end` нотация.

Для относительных макетов следует использовать `alignParentStart` и `alignParentEnd` , например:

```
<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>
```

Для определения гравитации и макета гравитации следует использовать аналогичные обозначения, например:

```
    <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"/>
```

Также должны быть указаны прокладки и поля, например:

```
<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"/>
```

Планирование локализации: тестовые макеты для RTL

Чтобы проверить, совместимы ли макеты, RTL, выполните следующие действия:

> Перейдите в Настройки -> Параметры разработчика -> Рисование -> Форматирование форматирования RTL

Включение этой опции заставит устройство использовать RTL-локаторы, и вы можете легко проверить все части приложения для поддержки RTL. Обратите внимание: вам не нужно добавлять новую локальную / языковую поддержку до этого момента.

Кодирование для локализации: создание строк и ресурсов по умолчанию

Первым шагом для кодирования для локализации является создание ресурсов по умолчанию. Этот шаг настолько скрыт, что многие разработчики даже не думают об этом. Однако создание ресурсов по умолчанию важно, потому что, если устройство работает в неподдерживаемой локали, оно будет загружать все свои ресурсы из папок по умолчанию. Если даже один из ресурсов отсутствует в папках по умолчанию, приложение просто выйдет из строя.

По умолчанию набор строк должен быть помещен в следующую папку в указанном месте:

```
res/values/strings.xml
```

Этот файл должен содержать строки на языке, на котором ожидается, что большинство пользователей приложения будут говорить.

Кроме того, ресурсы по умолчанию для приложения должны быть размещены в следующих папках и местоположениях:

```
res/drawable/
res/layout/
```

Если вашему приложению нужны папки, такие как `anim` или `xml` , ресурсы по умолчанию должны быть добавлены в следующие папки и местоположения:

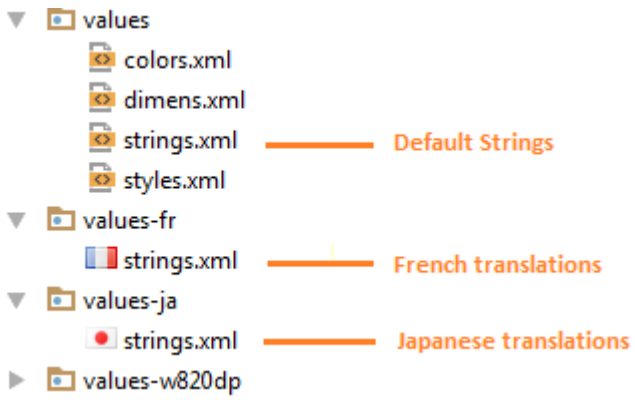
```
res/anim/
res/xml/
res/raw/
```

Кодирование для локализации: предоставление альтернативных строк

Чтобы обеспечить переводы на других языках (локалях), нам необходимо создать `strings.xml` в отдельной папке по следующему соглашению:

```
res/values-<locale>/strings.xml
```

Пример для этого приведен ниже:



В этом примере у нас есть английские строки по умолчанию в файле `res/values/strings.xml` , французские переводы представлены в папке `res/values-fr/strings.xml` а переводы японского языка указаны в папке `res/values-ja/strings.xml`

Другие переводы для других мест можно также добавить в приложение.

> Полный список кодов локали можно найти здесь: коды ISO 639

Непереводимые строки:

У вашего проекта могут быть определенные строки, которые не должны быть переведены. Строки, которые используются как ключи для SharedPreferences или строк, которые используются в качестве символов, попадают в эту категорию. Эти строки должны храниться только в файле `strings.xml` по умолчанию и должны быть помечены атрибутом `translatable="false"` . например

```
<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>
```

Этот атрибут важен, потому что переводы часто выполняются специалистами, которые

двуязычны. Это позволило бы этим лицам участвовать в переводах для определения строк, которые не должны быть переведены, что экономит время и деньги.

Кодирование для локализации: предоставление альтернативных макетов

Создание языковых макетов часто не требуется, если вы указали правильную нотацию `start/end` , как описано в предыдущем примере. Однако могут быть ситуации, когда макеты по умолчанию могут работать некорректно для определенных языков. Иногда макеты слева направо не могут переводить для языков RTL. В таких случаях необходимо обеспечить правильные раскладки.

Чтобы обеспечить полную оптимизацию для макетов RTL, мы можем использовать полностью отдельные файлы макета, используя `ldrtl` ресурса `ldrtl` (`ldrtl` означает layout-direction-right-to-left}). Например, мы можем сохранить ваши файлы макета по умолчанию в `res/layout/` и наши оптимизированные RTL макеты в `res/layout-ldrtl/` .

`ldrtl` подходит для ресурса с возможностью `ldrtl` , так что вы можете предоставить графику, ориентированную в направлении, соответствующем направлению чтения.

Вот отличный пост, который описывает приоритет макетов `ldrtl` : языковые макеты

Прочитайте Интернационализация и локализация (I18N и L10N) онлайн: https://riptutorial.com/ru/android/topic/8796/интернационализация-и-локализация--i18n-и-l10n-

глава 145: Интерфейс Java Java Native (JNI)

Вступление

JNI (Java Native Interface) - это мощный инструмент, который позволяет разработчикам Android использовать NDK и использовать собственный код C ++ в своих приложениях. В этом разделе описывается использование интерфейса Java <-> C ++.

Examples

Как вызвать функции в собственной библиотеке через интерфейс JNI

Интерфейс Java Native (JNI) позволяет вам вызывать собственные функции из кода Java и наоборот. В этом примере показано, как загружать и вызывать собственную функцию через JNI, она не переходит к доступу к методам и полям Java из собственного кода с использованием функций JNI .

Предположим, у вас есть родная библиотека с именем `libjniexample.so` в `libjniexample.so project/libs/<architecture>` , и вы хотите вызвать функцию из класса `JNITest` Java внутри пакета `com.example.jniexample` .

В классе JNITest объявите функцию следующим образом:

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

В вашем собственном коде определите функцию следующим образом:

```
#include <jni.h>

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

Аргумент `pEnv` является указателем на среду JNI, которую вы можете передать функциям JNI для доступа к методам и полям объектов и классов Java. `thiz` указатель является `jobject` ссылки на объект Java , который нативный метод был вызван (или класс , если это статический метод).

В вашем Java-коде в `JNITest` загрузите библиотеку следующим образом:

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

Обратите внимание на `lib` в начале, а `.so` в конце имени файла опущены.

Вызовите функцию native из Java следующим образом:

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

Как вызвать метод Java из собственного кода

Интерфейс Java Native Interface (JNI) позволяет вам вызывать функции Java из собственного кода. Вот простой пример того, как это сделать:

Код Java:

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

Родной код:

```
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);
}
```

Подпись JNI-метода к типу Java:

Подпись JNI	Тип Java
Z	логический
B	байт
C	голец
S	короткая
я	ИНТ

Подпись JNI	Тип Java
J	долго
F	поплавок
D	двойной
L полностью квалифицированный класс;	полностью квалифицирован-класс
[тип	тип[]

Итак, для нашего примера мы использовали (Z) I - что означает, что функция получает логическое значение и возвращает int.

Утилитный метод в слое JNI

Этот метод поможет получить строку Java из строки 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;
    }
```

Этот метод поможет вам преобразовать jbyteArray в 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;
```

```
}
```

Прочитайте Интерфейс Java Java Native (JNI) онлайн:
https://riptutorial.com/ru/android/topic/8674/интерфейс-java-java-native--jni-

глава 146: Интерфейсы

Examples

Пользовательский прослушиватель

Определить интерфейс

```
//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);
}
```

Создать слушателя

На следующем шаге нам нужно определить переменную экземпляра в объекте, который будет отправлять обратный вызов через `MyCustomListener` . И добавьте сеттер для нашего слушателя.

```
public class SampleClassB {
    private MyCustomListener listener;

    public void setMyCustomListener(MyCustomListener listener) {
        this.listener = listener;
    }
}
```

Выполнить прослушиватель

Теперь, в другом классе, мы можем создать экземпляр `SampleClassB` .

```
public class SomeActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
    }
}
```

Затем мы можем установить наш слушатель на `sampleClass` двумя способами:

реализует `MyCustomListener` в нашем классе:

```
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) {

    }
}
```

или просто создайте анонимный внутренний класс:

```
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) {

            }
        });
    }
}
```

Триггер-слушатель

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

Основной слушатель

Шаблон «слушатель» или «наблюдатель» является наиболее распространенной стратегией для создания асинхронных обратных вызовов в Android-разработке.

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

Теперь о вашей деятельности:

```
public class MyActivity extends Activity {
```

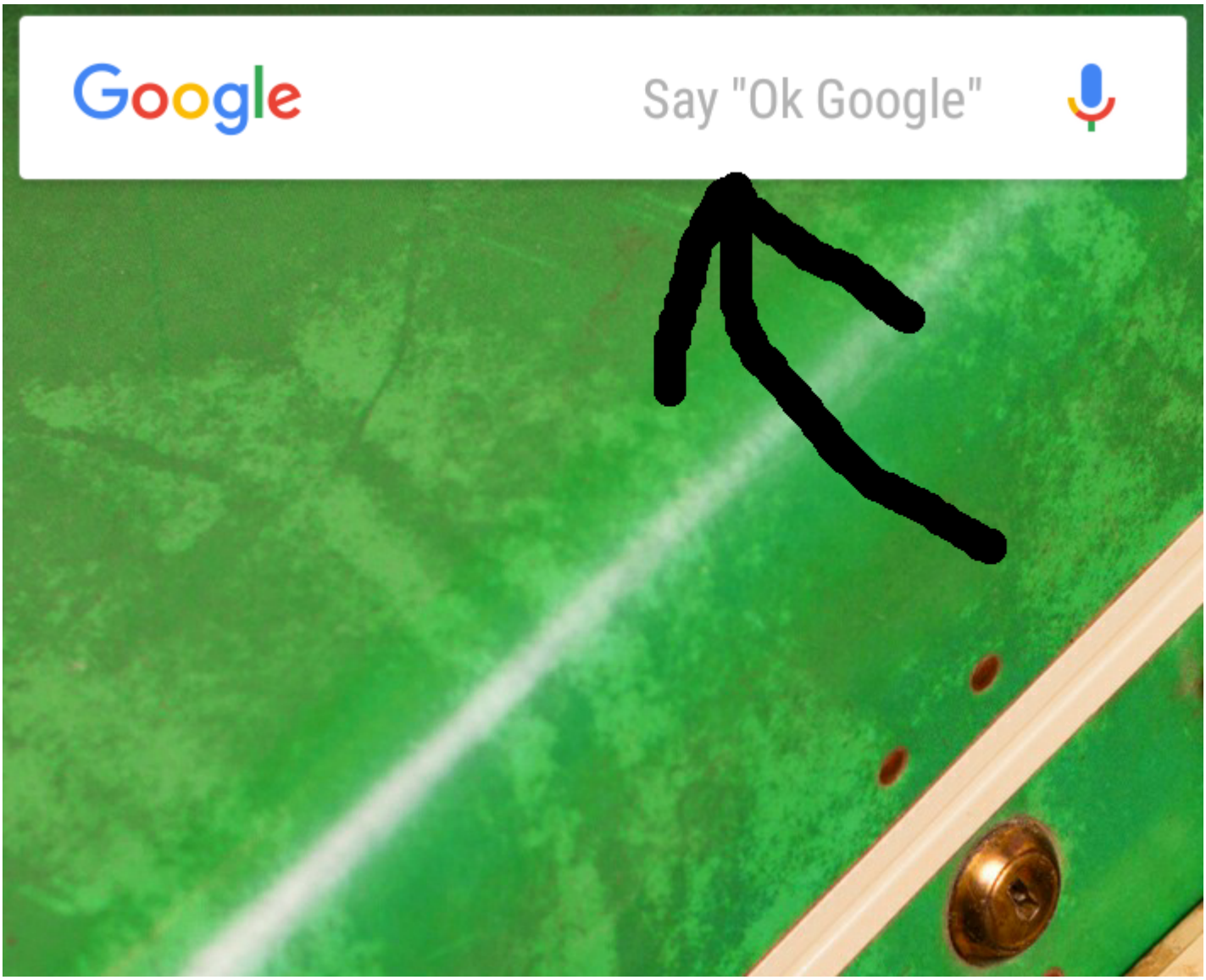
```
   public final String TAG = "MyActivity";

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

Прочитайте Интерфейсы онлайн: https://riptutorial.com/ru/android/topic/1785/интерфейсы

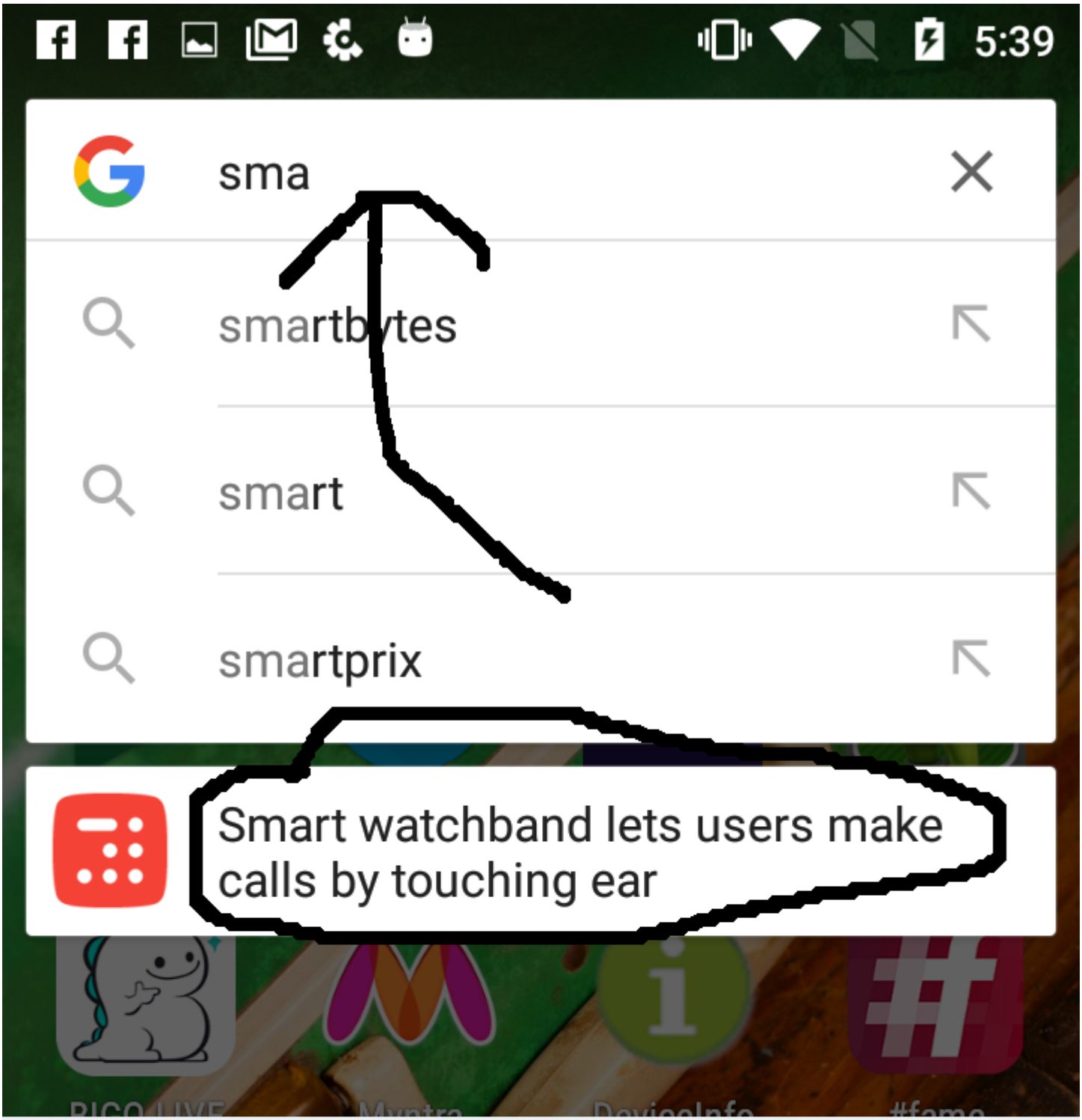
глава 147: Инфраструктура приложений Firebase

замечания

- Когда вы решите внедрить индексирование приложений, вы можете найти много блогов, документацию, которые могут вас смутить, в этом случае я предлагаю вам придерживаться официальных документов, предоставленных Firebase-Google. Даже если вы хотите использовать третью сторону для этого, сначала попробуйте следовать этой документации, потому что это даст вам четкое представление о том, как все работает.
- Google проведет около 24 часов, чтобы проиндексировать ваш контент. Будьте терпеливы. Вы можете сделать тестирование, чтобы все было хорошо на вашей стороне.
- Первый пример позволяет вам поддерживать HTTP-адрес вашего веб-сайта для перенаправления в вашем приложении. Это будет работать, например, вы искали запрос в поиске Google, результаты показывают один из URL вашего сайта, ссылки на приложения которого присутствуют в вашем уже установленном приложении. При щелчке по этому URL-адресу он перенаправляет вас прямо на ваш экран приложения, соответствующий этому результату поиска. Вот что я открыл для этого.
- Добавление API AppIndexing индексирует ваш контент и используется в Auto completions в панели поиска Google. Давайте возьмем пример приложения inShorts для каждой страницы, есть заголовок и небольшое описание. После чтения 2 или 3 заголовков закройте приложение и перейдите в google searchBar.



Попробуйте ввести заголовок, который вы только что прошли, вы получите предложение страницы страницы с заголовком в качестве заголовка. Это отличается от предложений приложений, которые вы получаете во время поиска приложений. Это происходит из-за того, что вы написали код API AppIndexing для этой конкретной страницы, и заголовок такой же, как вы инициализировали в `onCreate()` .


5:39
sma
smartbytes
smart
smartprix
Smart watchband lets users make calls by touching ear

вашего содержимого. Извлеките сервер robot.txt. Вы можете контролировать сканирование Google для вашего контента, отредактировав этот файл, вы можете обратиться к этой ссылке для получения более подробной информации.

Шаг 2: - Свяжите свое приложение с вашим сайтом. Включите ресурс links.json. Загрузите его в известную директорию вашего веб-сервера. Контент вашего ресурса links.json -

```
       [{
"relation": ["delegate_permission/common.handle_all_urls"],
"target" :
{ "namespace": "android_app",
"package_name": "<your_package_name>",
"sha256_cert_fingerprints": ["<hash_of_app_certificate>"] }
```

}]

Шаг 3: - Включите ссылки приложения в файл манифеста, чтобы перенаправить URL-адреса в ваше приложение, как показано ниже,

```
<activity
       android:name=".activity.SampleActivity"
       android:label="@string/app_name"
       android:windowSoftInputMode="adjustResize|stateAlwaysHidden">
       <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:host="example.live"
             android:pathPrefix="/vod"
             android:scheme="https"/>
       <data
             android:host="example.live"
             android:pathPrefix="/vod"
             android:scheme="http"/>
     </intent-filter>
   </activity>
```

Обратитесь к этому, если вы хотите узнать о каждом теге здесь.

<действие> Укажите **действие действия** ACTION_VIEW, чтобы можно было установить фильтр намерений из Google Search.

<data> Добавьте один или несколько тегов, где каждый тег представляет собой формат URI, который разрешает эту активность. Как минимум, тег должен включать атрибут android: schem. Вы можете добавить дополнительные атрибуты для дальнейшего уточнения типа URI, который принимает действие. Например, у вас может быть несколько действий, которые принимают аналогичные URI, но которые отличаются просто по имени пути. В этом случае используйте атрибут android: path или его варианты (pathPattern или pathPrefix), чтобы различать, какую деятельность система должна открывать для разных путей URI.

<category> Включить категорию BROWSABLE. Категория BROWSABLE требуется для того, чтобы фильтр намерения был доступен из веб-браузера. Без этого нажатие ссылки в браузере не может быть разрешено для вашего приложения. Категория DEFAULT является необязательной, но рекомендуется. Без этой категории активность может быть запущена только с явным намерением, используя ваше имя компонента приложения.

Шаг 4: - Обработка входящих URL-адресов

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

}

Шаг 5: - Вы можете проверить это, используя команду Android Debug Bridge или студийную конфигурацию. Команда Adb: - Запустите приложение и выполните следующую команду:

```
adb shell am start -a android.intent.action.VIEW -d "{URL}" < package name >
```

Android Studio Configurations: - **Студия Android> Сборка> Изменить конфигурацию> Параметры запуска> выберите URL> затем введите URL-адрес здесь> Применить** и проверить. Запустите приложение, если в окне «Выполнить» отображается ошибка, тогда вам нужно проверить свой формат URL с помощью приложения, упомянутые в манифесте, в противном случае он будет успешно запущен и перенаправлен на страницу, указанную вашим URL, если это указано.

Добавить API AppIndexing

Для добавления этого в проект вы можете легко найти официальный документ, но в этом примере я остановлюсь на некоторых ключевых областях, о которых нужно позаботиться.

Шаг 1: - Добавить службу google

```
dependencies {
      ...
      compile 'com.google.android.gms:play-services-appindexing:9.4.0'
      ...
    }
```

Шаг 2: - Импорт классов

```
import com.google.android.gms.appindexing.Action;
import com.google.android.gms.appindexing.AppIndex;
import com.google.android.gms.common.api.GoogleApiClient;
```

Шаг 3: - добавление API-интерфейсов индексирования приложений

```
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();
}
```

Чтобы проверить это, просто следуйте шагу 4 в примечаниях, приведенных ниже.

Прочитайте Инфраструктура приложений Firebase онлайн:
https://riptutorial.com/ru/android/topic/5957/инфраструктура-приложений-firebase

глава 148: Исключения

Examples

NetworkOnMainThreadException

Из документации :

> Исключение, которое возникает, когда приложение пытается выполнить сетевую операцию в своем основном потоке.
>
> Это делается только для приложений, ориентированных на SDK Honeycomb или выше. Приложениям, ориентированным на более ранние версии SDK, разрешено создавать сети по своим основным потокам цикла событий, но это сильно обескураживает.

Вот пример фрагмента кода, который может вызвать это исключение:

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

Над кодом будет `NetworkOnMainThreadException` для приложений, ориентированных на

Honeycomb SDK (Android v3.0) или выше, поскольку приложение пытается выполнить сетевую операцию в основном потоке.

Чтобы избежать этого исключения, ваши сетевые операции всегда должны выполняться в фоновом задании через `AsyncTask` , `Thread` , `IntentService` и т. Д.

```
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

Это очень распространенное `Exception` . Это заставляет ваше приложение останавливаться во время запуска или выполнения вашего приложения. В `LogCat` вы увидите сообщение:

```
android.content.ActivityNotFoundException : Unable to find explicit activity class;
have you declared this activity in your AndroidManifest.xml?
```

В этом случае проверьте, заявили ли вы свою активность в файле `AndroidManifest.xml` .

Самый простой способ объявить свою `Activity` в `AndroidManifest.xml` :

```
<activity  android:name="com.yourdomain.YourStoppedActivity" />
```

OutOfMemoryError

Это ошибка времени выполнения, которая возникает, когда вы запрашиваете большой объем памяти в куче. Это часто встречается при загрузке Bitmap в ImageView.

У вас есть несколько вариантов:

1. Используйте большую кучу приложений

Добавьте параметр «largeHeap» в тег приложения в вашем AndroidManifest.xml. Это сделает больше памяти доступной для вашего приложения, но, скорее всего, не устранит проблему с корнем.

```
<application largeHeap="true" ... >
```

2. Переработайте растровые изображения

После загрузки растрового изображения обязательно переработайте его и освободите память:

```
    if (bitmap != null && !bitmap.isRecycled())
       bitmap.recycle();
```

3. Загрузка образцовых растровых изображений в память

Избегайте загрузки всего битового массива в память сразу путем выборки уменьшенного размера, используя BitmapOptions и inSampleSize.

См., Например, документацию на Android

DexException

```
com.android.dex.DexException: Multiple dex files define Lcom/example/lib/Class;
```

Эта ошибка возникает из-за того, что приложение при упаковке находит два файла `.dex` которые определяют один и тот же набор методов.

Обычно это происходит из-за того, что приложение случайно приобрело две отдельные зависимости в одной и той же библиотеке.

Например, скажем, у вас есть проект, и вы хотите полагаться на две библиотеки: `A` и `B` , каждая из которых имеет свои собственные зависимости. Если библиотека `B` уже имеет зависимость от библиотеки `A` , эта ошибка будет выбрана, если библиотека `A` будет добавлена в проект сама по себе. Компиляция библиотеки `B` уже дала набор кода из `A` , поэтому, когда компилятор отправляется в библиотеку `A` , он находит уже обработанные методы библиотеки `A`

Чтобы решить проблему, убедитесь, что ни одна из ваших зависимостей не может быть

случайно добавлена дважды таким образом

UncaughtException

Если вы хотите обрабатывать исключенные снимки, попробуйте поймать их всех в методе onCreate для вас Класс приложения:

```
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();
    }
}
```

Регистрация собственного обработчика для неожиданных исключений

Вот как вы можете реагировать на исключения, которые не были обнаружены, подобно стандарту системы «Приложение XYZ разбилось»,

```
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
    }
}
```

Затем зарегистрируйте этот класс приложения в вашем AndroidManifest.xml:

```
<application android:name="de.ioxp.arkmobile.MyApplication" >
```

Прочитайте Исключения онлайн: https://riptutorial.com/ru/android/topic/112/исключения

глава 149: Как безопасно хранить пароли

Examples

Использование AES для шифрования соленого пароля

В этих примерах используется алгоритм AES для шифрования паролей. Длина соли может составлять до 128 бит.

Мы используем класс `SecureRandom` для генерации соли, которая объединяется с паролем для генерации секретного ключа. Используемые классы уже существуют в пакетах Android `javax.crypto` и `java.security` .

Когда генерируется ключ, мы должны сохранить этот ключ в переменной или сохранить его. Мы сохраняем его среди общих настроек в значении `S_KEY` . Затем пароль зашифровывается с использованием метода `doFinal` класса `Cipher` после его инициализации в `ENCRYPT_MODE` . Затем зашифрованный пароль преобразуется из массива байтов в строку и сохраняется среди общих настроек. Ключ, используемый для генерации зашифрованного пароля, может использоваться для дешифрования пароля аналогичным образом:

```
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;
    }
}
```

Прочитайте Как безопасно хранить пароли онлайн:
https://riptutorial.com/ru/android/topic/9093/как-безопасно-хранить-пароли

глава 150: Как использовать SparseArray

Вступление

`SparseArray` - альтернатива для `Map` . `Map` требует, чтобы ее ключи были объектами. Явление автобоксинга происходит, когда мы хотим использовать в качестве ключа примитивное значение `int` . Компилятор автоматически преобразует примитивные значения в свои бокс-типы (например, `int` в `Integer`). Разница в области памяти заметна: `int` использует 4 байта, `Integer` использует 16 байт. `SparseArray` использует `int` как ключевое значение.

замечания

Преимущество:

- Меньше использования памяти (из-за примитивных ключей).
- Нет автоматического бокса.

Недостаток :

- SparseArray использует двоичный поиск для значения find (O (log n)), поэтому его может быть не лучшим решением, если нужно работать с большим количеством элементов (используйте HashMap).

Существует несколько вариантов семейства, таких как: -SparseArray <Integer, Object> - SparseBooleanArray <Integer, Boolean> -SparseIntArray <Integer, Integer> -SparseLongArray <Integer, Long> -LongSparseArray <Long, Object> -LongSparseLongArray <Long, Long >

Операции SparseArray

- добавление элемента - put (int, x): добавляет сопоставление от указанного ключа к указанному значению, заменяя предыдущее сопоставление с указанным ключом, если оно есть. - append (int, x): помещает пару ключ / значение в массив, оптимизируя для случая, когда ключ больше всех существующих ключей в массиве. Вы должны использовать append () в случае последовательных ключей для оптимизации производительности. В противном случае put () в порядке.

- remove element - delete (int): Удаляет отображение из указанного ключа, если он есть. - removeAt (int): Удаляет отображение по данному индексу. - removeAtRange (int, int): удалить диапазон отображений в виде пакета.

- accessing element - get (int): Получает int, отображаемый из указанного ключа, или 0, если такое отображение не было сделано. - get (int, E): Получает int, отображаемый из указанного ключа, или указанное значение, если такое отображение не было

сделано. - valueAt (int): Учитывая индекс в диапазоне 0 ... size () - 1, возвращает значение из сопоставления значений ключа indexth, которое хранится в этом SparseIntArray. Индексы упорядочены в порядке возрастания.

- index / key search - keyAt (int): Учитывая индекс в диапазоне 0 ... size () - 1, возвращает ключ из сопоставления значений ключа indexth, который хранится в этом SparseIntArray. Индексы упорядочены в порядке возрастания. - valueAt (int): Учитывая индекс в диапазоне 0 ... size () - 1, возвращает значение из сопоставления значений ключа indexth, которое хранится в этом SparseIntArray. Индексы упорядочены в порядке возрастания. - indexOfKey (int): возвращает индекс, для которого keyAt (int) вернет указанный ключ или отрицательное число, если указанный ключ не отображается. - indexOfValue (E): возвращает индекс, для которого valueAt (int) вернет указанный ключ или отрицательное число, если никакие клавиши не сопоставляются с указанным значением. Остерегайтесь, что это линейный поиск, в отличие от поиска по ключу, и что несколько ключей могут сопоставляться с одним и тем же значением, и это найдет только один из них. Разница в области памяти заметна: int использует 4 байта, Integer использует 16 байт. SparseArray использует int как значение ключа.

Examples

Основной пример использования 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);
```

Учебник на YouTube

Прочитайте Как использовать SparseArray онлайн:
https://riptutorial.com/ru/android/topic/8824/как-использовать-sparsearray

глава 151: Камера и галерея

Examples

Выполнение полноразмерной фотографии с камеры

Чтобы сделать снимок, сначала нам нужно объявить необходимые разрешения в `AndroidManifest.xml` . Нам нужны два разрешения:

- `Camera` - открыть приложение для камеры. Если для атрибута `required` значение `true` вы не сможете установить это приложение, если у вас нет аппаратной камеры.
- `WRITE_EXTERNAL_STORAGE` - это разрешение требуется для создания нового файла, в котором сохраненная фотография будет сохранена.

AndroidManifest.xml

```
<uses-feature android:name="android.hardware.camera"
          android:required="true" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Основная идея при съемке полноразмерной фотографии с камеры заключается в том, что нам нужно создать новый файл для фото, прежде чем мы откроем приложение для камеры и сделаем снимок.

```
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();
    }
}
```

Сфотографировать

Добавьте разрешение на доступ к камере в файле AndroidManifest:

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

Файл 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>
```

Деятельность

```
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;
    }
}
```

Как запустить камеру или галерею и сохранить результат камеры в хранилище

Прежде всего вам нужны `Uri` и temp Folders и коды запросов:

```
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;
```

Затем init 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();
    }
```

```
}
```

Открытие `Camera` и `Gallery` намерения:

```
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);
    }

}
```

Затем в методе `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);
}
```

Вам нужны эти разрешения в `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" />
```

И вам нужно обрабатывать разрешения во время выполнения, такие как чтение / запись внешнего хранилища и т. Д. ...

Я проверяю разрешение `READ_EXTERNAL_STORAGE` в моем методе `openGallery` :

Мой `requestPermission` метод:

```
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);
    }
}
```

Затем переопределите метод `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);
    }
}
```

`showAlertDialog` метод:

```
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();
}
```

Установить разрешение камеры

Установите программное обеспечение с высоким разрешением.

```
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);
```

Декодирование растрового изображения корректно повернуто из ури, полученного с намерением

```
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;
}
```

Прочитайте Камера и галерея онлайн: https://riptutorial.com/ru/android/topic/4789/камера-и-галерея

глава 152: Картина холста с использованием SurfaceView

замечания

Перед использованием важно понять основную концепцию поверхности:

- В основном это отверстие в текущем окне
- Собственный пользовательский интерфейс можно разместить поверх него
- Рисование выполняется с использованием выделенного, неинтерфейсного потока
- Рисование не аппаратно ускорено
- Использует два буфера: один в настоящее время отображается, один используется для рисования.
- `unlockCanvasAndPost()` буферы.

Тупики могут легко возникать, если `lockCanvas()` и `unlockCanvasAndPost()` не вызываются в правильном порядке.

Examples

SurfaceView с рисованной нитью

В этом примере описано, как создать SurfaceView с выделенным потоком чертежа. Эта реализация также обрабатывает граничные случаи, такие как специфические проблемы производства, а также запуск / остановка потока для экономии времени процессора.

```
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");
    }
}
```

Этот макет содержит только пользовательский SurfaceView и максимизирует его размер экрана.

```
<?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>
```

Активность, которая использует SurfaceView, отвечает за запуск и прекращение рисования. Этот подход экономит батарею, когда рисунок останавливается, как только активность попадает в фоновый режим.

```
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();
    }
}
```

Прочитайте Картина холста с использованием SurfaceView онлайн:
https://riptutorial.com/ru/android/topic/3754/картина-холста-с-использованием-surfaceview

глава 153: Кинжал 2

Синтаксис

- @Module
- @Component (dependencies = {OtherComponent.class}, modules = {ModuleA.class, ModuleB.class})
- DaggerMyComponent.create ()
- DaggerMyComponent.builder (). MyModule (newMyModule ()). Создать ()

замечания

Не путать с кинжалом по квадрату, предшественником кинжалом 2.

Examples

Настройка компонентов для инъекций приложений и активности

Базовый `AppComponent` который зависит от одного `AppModule` для предоставления общесистемных одноэлементных объектов.

```
@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {

    void inject(App app);

    Context provideContext();

    Gson provideGson();
}
```

Модуль для использования вместе с `AppComponent` который будет предоставлять свои одноэлементные объекты, например, экземпляр `Gson` для повторного использования во всем приложении.

```
@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;
    }
}
```

Подклассовое приложение для настройки кинжала и синглтон-компонента.

```
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;
    }
}
```

Теперь компонент, зависящий от действия, который зависит от `AppComponent` чтобы получить доступ к объектам singleton.

```
@ActivityScope
@Component(dependencies = AppComponent.class, modules = ActivityModule.class)
public interface MainActivityComponent {

    void inject(MainActivity activity);
}
```

И многократно используемый `ActivityModule` который будет обеспечивать основные зависимости, такие как `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();
    }
}
```

Собирая все вместе, мы настроены и можем внедрить нашу деятельность и обязательно использовать то же самое `Gson` всех приложений!

```
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);
    }
}
```

Пользовательские области

```
@Scope
@Documented
@Retention(RUNTIME)
public @interface ActivityScope {
}
```

Области - это просто аннотации, и вы можете создавать свои собственные, где это необходимо.

Инъекция конструктора

Занятия без зависимостей могут быть легко созданы кинжалом.

```
public class Engine {

    @Inject // <-- Annotate your constructor.
    public Engine() {
    }
}
```

Этот класс может быть предоставлен *любым* компонентом. Он не имеет *никаких зависимостей* и *не имеет ограничений* . Дополнительного кода не требуется.

Зависимости объявляются как параметры в конструкторе. Кинжал вызовет конструктор и предоставит зависимости, если эти зависимости могут быть предоставлены.

```
public class Car {

    private Engine engine;

    @Inject
    public Car(Engine engine) {
        this.engine = engine;
    }
}
```

Этот класс может быть предоставлен каждым компонентом, *если* этот компонент может также обеспечить все его зависимости - `Engine` в этом случае. Так как `Engine` также может быть инжектирован конструктором, *любой* компонент может обеспечить `Car` .

Вы можете использовать инъекцию конструктора, когда все зависимости могут быть предоставлены компонентом. Компонент может обеспечить зависимость, если

- он может создать его, используя инъекцию конструктора
- модуль компонента может обеспечить его
- он может быть предоставлен родительским компонентом (если он является `@Subcomponent`)
- он может использовать объект, подвергаемый компоненту, зависящему от (зависимостей компонентов)

Использование @Subcomponent вместо @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);
    }
}
```

Как добавить кинжал 2 в build.gradle

Начиная с выпуска Gradle 2.2, использование плагина android-apt больше не используется. Должен использоваться следующий способ настройки кинжала 2. Для более старой версии Gradle используйте предыдущий метод, показанный ниже.

Для 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'
}
```

Для Gradle <2.2

Чтобы использовать Dagger 2, необходимо добавить плагин `android-apt` , добавьте его в root build.gradle:

```
buildscript {
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.0'
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}
```

Затем build.gradle модуля приложения должен содержать:

```
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}"
}
```

Ссылка: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Создание компонента из нескольких модулей

Dagger 2 поддерживает создание компонента из нескольких модулей. Вы можете создать свой компонент таким образом:

```
@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);
}
```

Затем два модуля ссылок `GeneralPurposeModule` и `SpecificModule` могут быть реализованы следующим образом:

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;
    }
}
```

Во время фазы ввода зависимостей компонент будет брать объекты из обоих модулей в соответствии с потребностями.

Этот подход очень полезен с точки зрения *модульности* . В этом примере используется модуль общего назначения, используемый для создания таких компонентов, как объект `Retrofit` (используемый для обработки сетевой связи) и `PropertiesReader` (отвечающий за обработку файлов конфигурации). Существует также специальный модуль, который обрабатывает создание конкретных контроллеров и классов обслуживания в отношении этого конкретного компонента приложения.

Прочитайте Кинжал 2 онлайн: https://riptutorial.com/ru/android/topic/3088/кинжал-2

глава 154: Кинжал библиотеки 2: Инъекция зависимостей в приложениях

Вступление

Кинжал 2, как объяснено в GitHub , является методом эволюции во время компиляции для инъекций зависимостей. Принимая подход, начатый в Dagger 1.x до его окончательного вывода, Dagger 2.x устраняет все отражения и улучшает четкость кода, удаляя традиционный `ObjectGraph` / `Injector` в пользу пользовательских интерфейсов `@Component` .

замечания

1. Настройка библиотеки в приложении (для проектов maven, gradle, java)
2. Преимущества использования Dragger
3. Важные ссылки (для документации и демонстраций)
4. Как интегрировать и использовать компоненты Dragger

Dagger 2 API:

Кинжал 2 предоставляет ряд специальных аннотаций:

@Module для классов, методы которых обеспечивают зависимости

@Provides для методов в классах @Module

@Inject для запроса зависимости (конструктор, поле или метод)

@Component - это интерфейс моста между модулями и инъекцией

Важные ссылки:

GitHub: https://github.com/google/dagger

UserGuide (Google): https://google.github.io/dagger/users-guide.html

Видео: https://google.github.io/dagger/resources.html

Vogella Tutorial: http://www.vogella.com/tutorials/Dagger/article.html

Codepath Tutorial: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Examples

Создайте аннотацию @Module и @Singleton для объекта

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

Каждый поставщик (или метод) должен иметь аннотацию `@Provides` а класс должен иметь аннотацию `@Module` . Аннотация `@Singleton` указывает, что будет только один экземпляр объекта.

Зависимость запросов от зависимых объектов

Теперь, когда у вас есть поставщики для разных моделей, вам необходимо их запросить. Так же, как `Vehicle` нуждается в `Motor` , вы должны добавить аннотацию `@Inject` в конструктор `Vehicle` следующим образом:

```
@Inject
public Vehicle(Motor motor){
    this.motor = motor;
}
```

Вы можете использовать аннотацию `@Inject` для запроса зависимостей в конструкторе, полях или методах. В этом примере я сохраняю инъекцию в конструкторе.

Подключение @Modules с @Inject

Соединение между поставщиком зависимостей, `@Module` и классами, запрашивающими их через `@Inject` , выполняется с помощью `@Component` , который является интерфейсом:

```
import javax.inject.Singleton;
import dagger.Component;

@Singleton
@Component(modules = {VehicleModule.class})
```

```
public interface VehicleComponent {
    Vehicle provideVehicle();
}
```

Для аннотации `@Component` вы должны указать, какие модули будут использоваться. В этом примере используется `VehicleModule` , который определен в этом примере . Если вам нужно использовать больше модулей, просто добавьте их, используя запятую в качестве разделителя.

Использование интерфейса @Component для получения объектов

Теперь, когда у вас есть все готовые соединения, вы должны получить экземпляр этого интерфейса и вызвать его методы для получения требуемого объекта:

```
VehicleComponent component = Dagger_VehicleComponent.builder().vehicleModule(new
VehicleModule()).build();
vehicle = component.provideVehicle();
Toast.makeText(this, String.valueOf(vehicle.getSpeed()), Toast.LENGTH_SHORT).show();
```

Когда вы пытаетесь создать новый объект интерфейса с `@Component` аннотации `@Component` , вы должны сделать это, используя префикс `Dagger_<NameOfTheComponentInterface>` , в данном случае `Dagger_VehicleComponent` , а затем используйте метод builder для вызова каждого модуля внутри.

Прочитайте Кинжал библиотеки 2: Инъекция зависимостей в приложениях онлайн: https://riptutorial.com/ru/android/topic/9079/кинжал-библиотеки-2--инъекция-зависимостей-в-приложениях

глава 155: клавиатура

Examples

Скрыть клавиатуру, когда пользователь нажимает на другое место на экране

Добавьте код в свою **деятельность** .

Это будет работать и для **Fragment** , **поэтому нет необходимости** добавлять этот код в **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);
}
```

Регистрация обратного вызова для открытия и закрытия клавиатуры

Идея состоит в том, чтобы измерить макет до и после каждого изменения, и если произошли существенные изменения, вы можете быть уверены, что его программная клавиатура.

```
// 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;
```

```
            }
        }
};
```

то в нашем `onCreate` задайте начальное значение для `mLastContentHeight`

```
mLastContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();
```

и добавить слушателя

```
rootView.getViewTreeObserver().addOnGlobalLayoutListener(keyboardLayoutListener);
```

не забудьте удалить слушателя при `destroy`

```
rootView.getViewTreeObserver().removeOnGlobalLayoutListener(keyboardLayoutListener);
```

Прочитайте клавиатура онлайн: https://riptutorial.com/ru/android/topic/5606/клавиатура

глава 156: кнопка

Синтаксис

- <Кнопка ... />
- андроид: OnClick = "MethodName"
- button.setOnClickListener (новый OnClickListener () {...});
- public class classname реализует View.OnLongClickListener

Examples

inline onClickListener

Скажем, у нас есть кнопка (мы можем создать ее программно или связать ее с представлением с помощью findViewbyId () и т. Д.)

```
Button btnOK = (...)
```

Теперь создайте анонимный класс и установите его в строке.

```
btnOk.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Do stuff here...
    }
});
```

Использование макета для определения действия клика

Когда мы создаем кнопку в макете, мы можем использовать атрибут `android:onClick` для ссылки на метод в коде для обработки кликов.

кнопка

```
<Button
    android:width="120dp"
    android:height="wrap_content"
    android:text="Click me"
    android:onClick="handleClick" />
```

Затем в вашей деятельности создайте метод `handleClick` .

```
public void handleClick(View v) {
    // Do whatever.
}
```

Использование одного и того же события click для одного или нескольких представлений в XML

Когда мы создаем любой вид в макете, мы можем использовать атрибут android: onClick для ссылки на метод в связанной деятельности или фрагменте для обработки событий щелчка.

XML-макет

```
<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" />
```

Код операции / фрагмента

В вашем **коде** создайте метод, который вы назвали, где v будет рассмотренным, и сделать что-то для каждого вида, вызывающего этот метод.

```
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;
    }
}
```

Если вы хотите, вы также можете использовать разные методы для каждого вида (в этом случае, конечно, вам не нужно проверять идентификатор).

Прослушивание событий с длинным кликом

Чтобы поймать длинный клик и использовать его, вам необходимо предоставить соответствующую кнопку для прослушивания:

```
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);
```

Определение внешнего прослушивателя

Когда следует использовать его

- Когда код внутри встроенного прослушивателя слишком велик, и ваш метод / класс становится уродливым и трудно читаемым
- Вы хотите выполнить одно и то же действие в различных элементах (представлении) вашего приложения

Для этого вам необходимо создать класс, реализующий один из слушателей в API просмотра .

Например, дайте помощь, когда вы долго щелкните по любому элементу:

```
public class HelpLongClickListener implements View.OnLongClickListener
{
    public HelpLongClickListener() {
    }

    @Override
    public void onLongClick(View v) {
        // show help toast or popup
    }
}
```

Тогда вам просто нужно иметь атрибут или переменную в вашей `Activity` чтобы использовать ее:

```
HelpLongClickListener helpListener = new HelpLongClickListener(...);

button1.setOnClickListener(helpListener);
button2.setOnClickListener(helpListener);
label.setOnClickListener(helpListener);
button1.setOnClickListener(helpListener);
```

ПРИМЕЧАНИЕ. Определение слушателей в отдельном классе имеет один недостаток: он не может напрямую обращаться к полям класса, поэтому вам необходимо передать данные (контекст, представление) через конструктор, если вы не используете атрибуты public или не определяете geters.

Пользовательский кликер для предотвращения нескольких быстрых кликов

Чтобы **предотвратить многократное срабатывание кнопки за короткий промежуток**

времени (скажем, 2 клика в течение 1 секунды, что может вызвать серьезные проблемы, если поток не контролируется), можно реализовать пользовательский ***SingleClickListener*** .

Этот ClickListener устанавливает определенный временной интервал как порог (например, 1000 мс).
Когда кнопка нажата, будет проверяться, будет ли запускаться триггер за прошедшее время, которое вы определили, и если это не вызвало его.

```
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);

}
```

И в классе, SingleClickListener связан с Button на карту

```
myButton = (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(new SingleClickListener() {
    @Override
    public void performClick(View view) {
        // do stuff
    }
});
```

Пользовательский стиль кнопки

Существует много возможных способов настройки внешнего вида кнопки. В этом примере представлены несколько вариантов:

Вариант 0: используйте ThemeOverlay (в настоящее время самый простой / быстрый способ)

Создайте новый стиль в файле стилей:

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>
```

Затем в макете, где вы размещаете свою кнопку (например, 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>
```

Вариант 1. Создайте свой собственный стиль кнопки

В значениях / styles.xml создайте новый стиль для своей кнопки:

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>
```

Затем в макете, где вы помещаете свою кнопку (например, в 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>
```

Вариант 2: назначьте для каждого из состояний кнопки возможность рисования

Создайте xml-файл в выпадающей папке с именем «mybuttondrawable.xml», чтобы определить ресурс, который можно использовать для каждого из состояний ваших кнопок:

рисуем / 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>
```

Каждый из этих чертежей может представлять собой изображения (например, mybutton_disabled.png) или xml-файлы, определенные вами и сохраненные в папке drawables. Например:

рисуем / 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>
```

Затем в макете, где вы размещаете свою кнопку (например, 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>
```

Вариант 3: добавьте стиль вашей кнопки в тему приложения

Вы можете переопределить стиль кнопки Android по умолчанию в определении темы

вашего приложения (в значениях / 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>
```

Вариант 4. Наложение цвета на стиль кнопки по умолчанию программно

Просто найдите кнопку в своей деятельности и примените цветной фильтр:

```
Button mybutton = (Button) findViewById(R.id.mybutton);
mybutton.getBackground().setColorFilter(anycolor, PorterDuff.Mode.MULTIPLY)
```

Вы можете проверить различные режимы смешивания здесь и хорошие примеры здесь .

Прочитайте кнопка онлайн: https://riptutorial.com/ru/android/topic/5607/кнопка

глава 157: Компоненты архитектуры Android

Вступление

Компоненты Android Architecture - это новая коллекция библиотек, которые помогут вам создавать надежные, тестируемые и поддерживаемые приложения. Основные части: Lifecycles, ViewModel, LiveData, Room.

Examples

Добавление компонентов архитектуры

Проект 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'
}
```

График построения приложения

```
// 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"
```

Использование жизненного цикла в AppCompatActivity

Расширьте свою активность от этой деятельности

```
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 с преобразованиями 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;
    }
}
```

Где-то в пользовательском интерфейсе:

```
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);
        }
}
```

Пространство помещения

Комната требует четыре части: класс базы данных, классы DAO, классы сущностей и классы миграции (теперь вы можете использовать **только методы DDL**):

Сущности

```
// 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;
}
```

Классы 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);
}
```

Класс базы данных

```
// register your entities and DAOs
@Database(entities = {VideoItem.class, VideoTag.class}, version = 2)
public abstract class ContentDatabase extends RoomDatabase {
    public abstract VideoDao videoDao();
}
```

Миграции

```
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() {
    }
```

```
}
```

Использовать в классе приложения или предоставлять через кинжал

```
ContentDatabase provideContentDatabase() {
    return Room.databaseBuilder(context, ContentDatabase.class, "data.db")
            .addMigrations(Migrations.ALL).build();
}
```

Напишите свой репозиторий:

```
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);
    }
}
```

Использование в ViewModel:

```
ContentRepository contentRepository = ...;
contentRepository.getVideoByTag(tag, limit);
```

Пользовательские LiveData

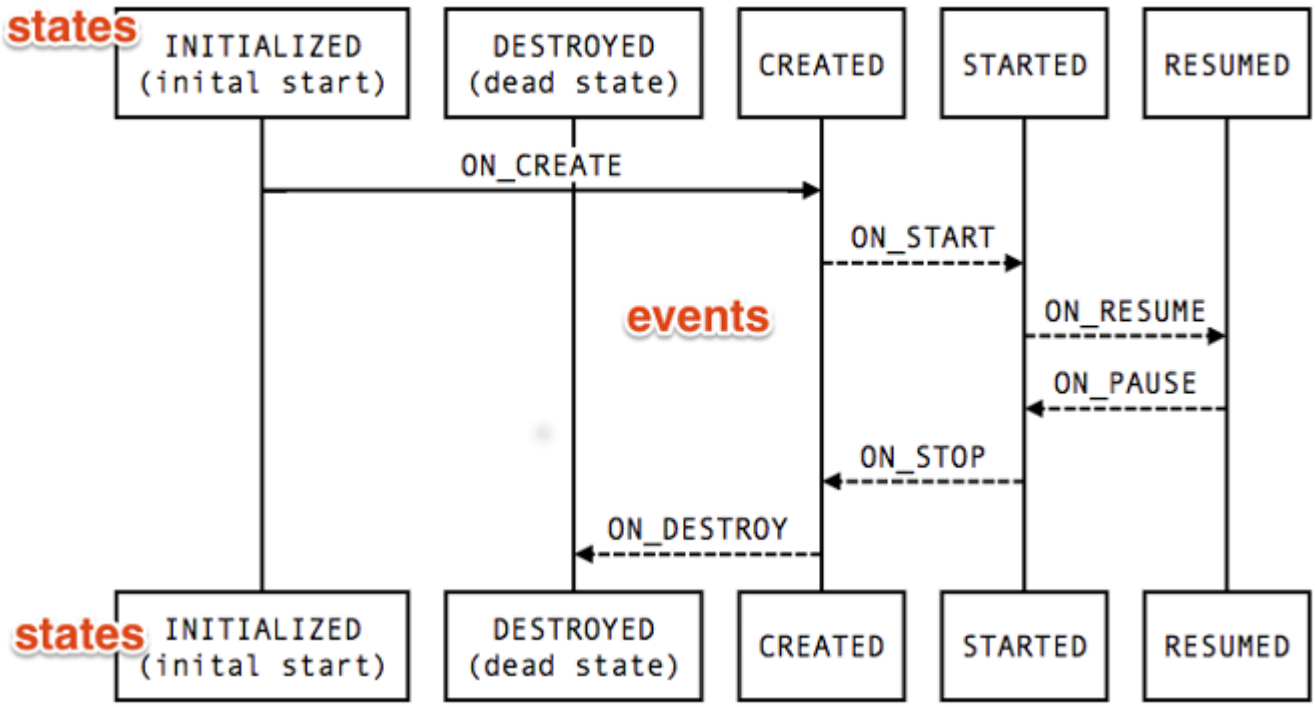
Вы можете написать пользовательские LiveData, если вам нужна специальная логика. Не пишите пользовательский класс, если вам нужно только преобразовать данные (используйте класс Transformations)

```
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);
    }
}
```

Пользовательский компонент, совместимый с жизненным циклом

Каждый жизненный цикл компонента пользовательского интерфейса изменился, как показано на изображении.



Вы можете создать компонент, который будет уведомлен об изменении состояния жизненного цикла:

```
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
    }
}
```

Прочитайте Компоненты архитектуры Android онлайн:
https://riptutorial.com/ru/android/topic/10872/компоненты-архитектуры-android

глава 158: контекст

Вступление

В документации Google: «Интерфейс к глобальной информации о среде приложения. Он позволяет получать доступ к ресурсам и классам приложений, а также переадресации для операций на уровне приложений, таких как запуск, трансляция и получение намерений и т. Д.».

Проще говоря, Context - это текущее состояние вашего приложения. Это позволяет вам предоставлять информацию объектам, чтобы они могли знать о том, что происходит в других частях вашего приложения.

Синтаксис

- `getApplicationContext()`
- `getBaseContext()`
- `getContext()`
- `this`

замечания

Эта страница StackOverflow содержит несколько полных и хорошо написанных объяснений концепции Context:

Что такое контекст?

Examples

Основные примеры

Стандартное использование в действии:

```
Context context = getApplicationContext();
```

Стандартное использование в фрагменте:

```
Context context = getActivity().getApplicationContext();
```

`this` (когда в классе, который распространяется из контекста, например, в классах Application, Activity, Service и IntentService)

```
TextView textView = new TextView(this);
```

другой `this` пример:

```
Intent intent = new Intent(this, MainActivity.class);
startActivity(intent);
```

Прочитайте контекст онлайн: https://riptutorial.com/ru/android/topic/9774/контекст

глава 159: КоординаторLayout и поведение

Вступление

КоординаторLayout является супермощным FrameLayout, и цель этой ViewGroup - координировать представления, которые внутри него.

Основной привлекательностью CoordinatorLayout является его способность координировать анимацию и переходы представлений в самом файле XML.

КоординаторLayout предназначен для двух основных случаев использования:

: Как декор приложения верхнего уровня или хром

: Как контейнер для конкретного взаимодействия с одним или несколькими дочерними представлениями

замечания

`CoordinatorLayout` - это контейнер, который расширяет `FrameLayout` .
Присоединив `CoordinatorLayout.Behavior` к прямому ребенку `CoordinatorLayout` , вы сможете перехватывать события касания, вставки окна, измерение, макет и вложенную прокрутку.

Указав `Behaviors` для дочерних представлений `CoordinatorLayout` вы можете обеспечить множество разных взаимодействий внутри одного родителя, и эти представления также могут взаимодействовать друг с другом. В классах просмотра могут указываться поведение по умолчанию при использовании в качестве дочернего элемента `CoordinatorLayout` с использованием аннотации `DefaultBehavior` .

Examples

Создание простого поведения

Чтобы создать `Behavior` просто расширьте класс `CoordinatorLayout.Behavior` .

Расширьте координаторLayout.Behavior

Пример:

```
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);
      }
}
```

Это поведение должно быть привязано к ребенку. Вид `CoordinatorLayout` для вызова.

Привязать поведение программно

```
MyBehavior myBehavior = new MyBehavior();
CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams)
view.getLayoutParams();
params.setBehavior(myBehavior);
```

Привязать поведение в XML

Вы можете использовать атрибут `layout_behavior` для привязки поведения в XML:

```
<View
  android:layout_height="...."
  android:layout_width="...."
  app:layout_behavior=".MyBehavior" />
```

Прикрепите поведение автоматически

Если вы работаете с настраиваемым представлением, вы можете прикрепить поведение, используя аннотацию `@CoordinatorLayout.DefaultBehavior` :

```
@CoordinatorLayout.DefaultBehavior(MyBehavior.class)
public class MyView extends ..... {

}
```

Использование SwipeDismissBehavior

`SwipeDismissBehavior` работает над любым представлением и реализует функциональные возможности салфетки для удаления в наших макетах с помощью `CoordinatorLayout` .

Просто используйте:

```
        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);
```

Создание зависимостей между представлениями

Вы можете использовать `CoordinatorLayout.Behavior` для создания зависимостей между представлениями. Вы можете якорь `View` на другой `View` по:

- используя атрибут `layout_anchor` .
- создание пользовательского `Behavior` и реализация метода `layoutDependsOn` возвращающего `true` .

Например, чтобы создать `Behavior` для перемещения `ImageView` при перемещении другого (пример панели инструментов), выполните следующие действия:

- Создайте пользовательское поведение :

```
public class MyBehavior extends CoordinatorLayout.Behavior<ImageView> {...}
```

- Переопределите метод `layoutDependsOn` возвращающий `true` . Этот метод вызывается каждый раз, когда происходит изменение в макете:

```
@Override
```

```
public boolean layoutDependsOn(CoordinatorLayout parent,
        ImageView child, View dependency) {
    // Returns true to add a dependency.
    return dependency instanceof Toolbar;
}
```

- Всякий раз, когда метод `layoutDependsOn` возвращает `true` `onDependentViewChanged` метод `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);
}
```

Прочитайте КоординаторLayout и поведение онлайн:
https://riptutorial.com/ru/android/topic/5714/координаторlayout-и-поведение

глава 160: Коснитесь событий

Examples

Как варьироваться между дочерними и родительскими группами событий касания

1. `onTouchEvents()` для вложенных групп представлений может управляться `boolean` onInterceptTouchEvent .

Значение по умолчанию для `OnInterceptTouchEvent` равно false.

Родительский `onTouchEvent` принимается до `onTouchEvent` ребенка. Если `OnInterceptTouchEvent` возвращает значение false, он отправляет событие движения по цепочке в обработчик `OnTouchEvent` дочернего `OnTouchEvent` . Если он вернет true, родительский элемент обработает событие touch.

Однако могут быть случаи, когда мы хотим, чтобы некоторые дочерние элементы управляли `OnTouchEvent` s, а некоторые из них управлялись родительским представлением (или, возможно, родителем родителя).

Это можно управлять несколькими способами.

2. Одним из способов защиты дочернего элемента от родительского `OnInterceptTouchEvent` является реализация requestDisallowInterceptTouchEvent .

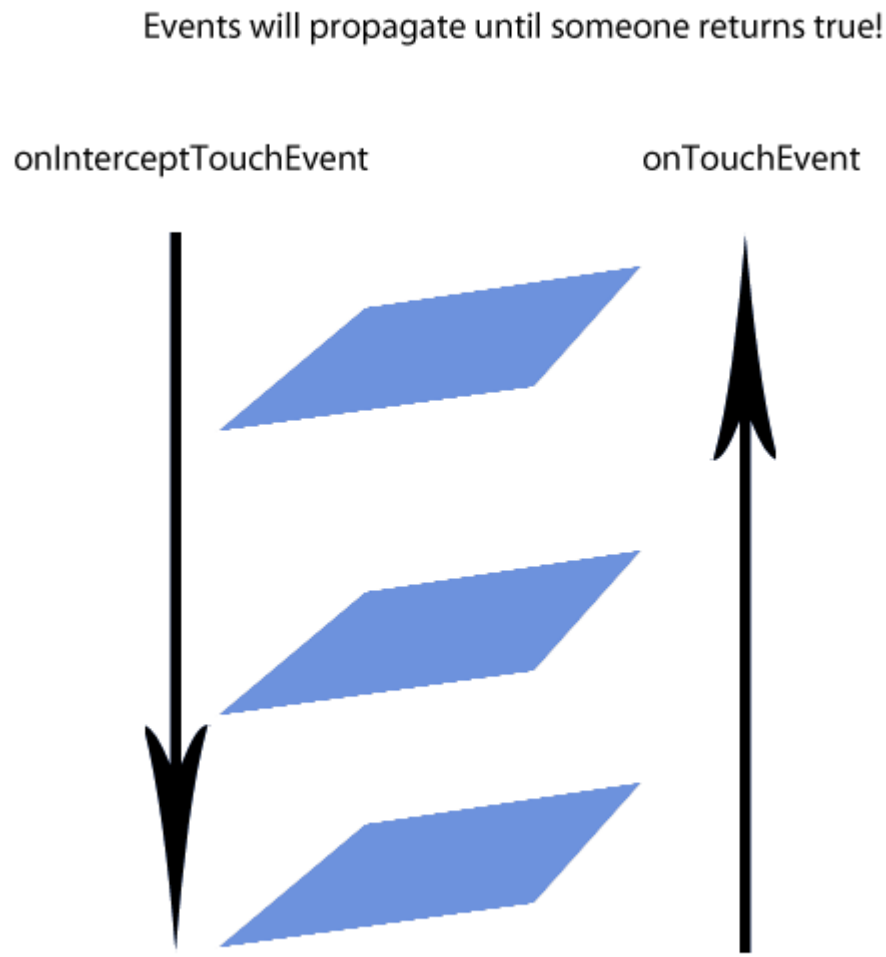
   public void requestDisallowInterceptTouchEvent (boolean disallowIntercept)

Это не позволяет любому из родительских представлений управлять `OnTouchEvent` для этого элемента, если элемент имеет обработчики событий.

3.

Если `OnInterceptTouchEvent` является ложным, будет оцениваться `OnTouchEvent` дочернего элемента. Если у вас есть методы в дочерних элементах, обрабатывающих различные события касания, любые связанные обработчики событий, которые отключены, возвращают OnTouchEvent родительскому элементу.

Этот ответ:
Визуализация того, как проходит распространение сенсорных событий:
`parent -> child|parent -> child|parent -> child views.`



Предоставлено здесь

4. Другой способ - вернуть переменные значения из `OnInterceptTouchEvent` для родителя.

Этот пример взят из Managing Touch Events в ViewGroup и демонстрирует, как перехватывать `OnTouchEvent` ребенка при прокрутке пользователя.

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;
}
```

Это код из той же ссылки, показывающий, как создать параметры прямоугольника вокруг вашего элемента:

4б.

```
// 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);
}
```

Прочитайте Коснитесь событий онлайн: https://riptutorial.com/ru/android/topic/7167/коснитесь-событий

глава 161: Кэш Bitmap

Вступление

Эффективное кэширование с использованием бит-памяти. Это особенно важно, если ваше приложение использует анимацию, поскольку они будут остановлены во время очистки GC и сделают ваше приложение менее вялым для пользователя. Кэш позволяет повторно использовать объекты, которые дорого создавать. Если вы загружаете объект в память, вы можете думать об этом как кэш для объекта. Работа с растровым рисунком в android сложна. Важно, чтобы кешировать бимап, если вы собираетесь использовать его повторно.

Синтаксис

- `LruCache<String, Bitmap> mMemoryCache;//declaration of LruCache object.`
- void addBitmapToMemoryCache (String key, Bitmap bitmap) {} // объявление общего метода, добавляющего растровое изображение в кэш-память
- Bitmap getBitmapFromMemCache (String key) {} // объявление универсального метода для получения бимака из кеша.

параметры

параметр	подробности
ключ	ключ для сохранения растрового изображения в кеше памяти
битовая карта	растровое значение, которое будет кэшироваться в память

Examples

Кэш растровых изображений с использованием кеша LRU

Ключ LRU

Следующий пример кода демонстрирует возможную реализацию класса LruCache для кэширования изображений.

```
private LruCache<String, Bitmap> mMemoryCache;
```

Здесь строковое значение является ключом для значения битмапа.

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

Для добавления растрового изображения в кэш памяти

```
public void addBitmapToMemoryCache(String key, Bitmap bitmap) {
if (getBitmapFromMemCache(key) == null) {
        mMemoryCache.put(key, bitmap);
    }
}
```

Для получения растрового изображения из кеша памяти

```
public Bitmap getBitmapFromMemCache(String key) {
    return mMemoryCache.get(key);
}
```

Для загрузки растрового изображения в изображение просто используйте **getBitmapFromMemCache («Pass key»).**

Прочитайте Кэш Bitmap онлайн: https://riptutorial.com/ru/android/topic/9901/кэш-bitmap

глава 162: Локализация ресурсов на Android

Examples

валюта

```
Currency currency = Currency.getInstance("USD");
NumberFormat format = NumberFormat.getCurrencyInstance();
format.setCurrency(currency);
format.format(10.00);
```

Добавление перевода в приложение для Android

Вам нужно создать другой файл `strings.xml` для каждого нового языка.

1. Щелкните правой кнопкой мыши папку *res*
2. Выберите *Новый файл ресурсов* → *Значения*
3. Выберите локаль из доступных квалификаторов
4. Нажмите кнопку « *Далее*» (>>)
5. Выберите язык
6. Назовите файл *strings.xml*

strings.xml

```
<resources>
    <string name="app_name">Testing Application</string>
    <string name="hello">Hello World</string>
</resources>
```

strings.xml (привет)

```
<resources>
    <string name="app_name">परीक्षण आवेदन</string>
    <string name="hello">नमस्ते दुनिया</string>
</resources>
```

Программирование программно:

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

Вышеуказанная функция изменит текстовые поля, на которые ссылаются *строки strings.xml* . Например, предположим, что у вас есть следующие два текстовых вида:

```
<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"/>
```

Затем, после изменения языкового стандарта, строки языка, имеющие идентификаторы `app_name` и `hello` будут соответственно изменены.

Тип каталогов ресурсов в папке «res»

Когда требуется локализация различных типов ресурсов, каждый из них имеет свой собственный дом в структуре проекта Android. Ниже приведены различные каталоги, которые мы можем разместить в каталоге `\res` . Типы ресурсов, размещенные в каждом из этих каталогов, описаны в следующей таблице:

каталог	Тип ресурса
аниматор /	XML-файлы, которые определяют анимацию свойств.
аним /	XML-файлы, которые определяют анимацию анимации. (Имущественная анимация также может быть сохранена в этом каталоге, но каталог аниматора / предпочтительнее для анимации свойств, чтобы различать два типа.)
цвет/	XML-файлы, которые определяют список состояний цветов. См. Раздел «Список состояний цвета»
рисуем /	«Растровые файлы (.png, .9.png, .jpg, .gif) или файлы XML, которые

каталог	Тип ресурса
	скомпилированы в следующие подтипы поддающихся ресурсам:: `Bitmap files - Nine-Patches (re-sizable bitmaps) - State lists - Shapes - Animation drawables - Other drawables - "`
мипмап /	Файлы с возможностью рисования для различных значений плотности значков пусковых установок. Дополнительные сведения об управлении значками запуска с помощью mipmap / folders см. В разделе «Управление проектами».
макет /	XML-файлы, которые определяют макет пользовательского интерфейса. См. «Ресурс компоновки».
меню/	XML-файлы, которые определяют меню приложений, например меню параметров, контекстное меню или подменю. См. «Ресурс меню».
сырье /	Произвольные файлы для сохранения в исходном виде. Чтобы открыть эти ресурсы с помощью сырого InputStream, вызовите Resource.openRawResource () с идентификатором ресурса, который является R.raw.filename.
	Однако, если вам нужен доступ к исходным именам файлов и иерархии файлов, вы можете рассмотреть возможность сохранения некоторых ресурсов в каталоге assets / resources (вместо ofres / raw /). Файлы в активах / не получают идентификатор ресурса, поэтому вы можете читать их только с помощью AssetManager.
ценности/	XML-файлы, содержащие простые значения, такие как строки, целые числа и цвета, а также стили и темы
XML /	Произвольные XML-файлы, которые можно прочитать во время выполнения, вызывая Resource.getXML (). Здесь должны быть сохранены различные файлы конфигурации XML, такие как конфигурация, доступная для поиска.

Типы конфигурации и имена квалификаторов для каждой папки в каталоге «res»

Каждый каталог ресурсов в папке `res` (перечисленный в примере выше) может иметь разные варианты содержащихся ресурсов в каталоге с одинаковым именем, суффикс с различными `qualifier-values` для каждого `configuration-type` .

Пример вариаций `` каталог с разными значениями классификатора, которые часто встречаются в наших проектах для Android:

- рисуем /
- рисуем-ан /
- вытяжка-FR-ПКА /
- вытяжка-ен-порт /
- вытяжка-ен-NoTouch-12key /
- вытяжка-порт-ldpi /
- вытяжка-порт-NoTouch-12key /

Исчерпывающий список всех разных типов конфигурации и их значений квалификатора для ресурсов android:

конфигурация	Квалификационные значения
MCC и MNC	Примеры:
	mcc310
	mcc310-mnc004
	mcc208-mnc00
	и т.п.
Язык и регион	Примеры:
	ан
	фр
	ан-РУСЬ
	фр-RFR
	фр-PKA
Направление макета	ldrtl
	ldltr
smallestWidth	swdp
	Примеры:
	sw320dp

конфигурация	Квалификационные значения
	sw600dp
	sw720dp
Доступная ширина	WDP
	w720dp
	w1024dp
Доступная высота	HDP
	h720dp
	h1024dp
Размер экрана	маленький
	нормальный
	большой
	XLarge
Аспект экрана	долго
	недолго
Круглый экран	круглый
	notround
Ориентация экрана	порт
	земельные участки
Режим пользовательского интерфейса	автомобиль
	стол письменный
	телевидение
	appliancewatch
Ночной режим	ночь
	notnight
Плотность пикселей экрана (dpi)	ldpi

конфигурация	Квалификационные значения
	MDPI
	ИПЧР
	xhdpi
	xxhdpi
	xxxhdpi
	nodpi
	tvdpi
	anydpi
Тип сенсорного экрана	без касаний
	Палец
Доступность клавиатуры	keysexposed
	keyshidden
	keyssoft
Метод первичного ввода текста	nokeys
	БУКВ
	12key
Доступ к навигационной клавише	navexposed
	navhidden
Первичный метод без касания	nonav
	DPad
	трекбол
	рулевое колесо
Версия платформы (уровень API)	Примеры:
	v3
	v4

конфигурация	Квалификационные значения
	v7

Измените язык приложения Android

В приведенных выше примерах вы понимаете, как локализовать ресурсы приложения. В следующем примере объясните, как изменить языковой стандарт приложения внутри приложения, а не на устройстве. Чтобы изменить только локализацию приложения, вы можете использовать утилиту ниже locale.

```
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();
    }
}
```

Инициализируйте язык, который пользователь предпочитает, из класса 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());
    }
}
```

Вам также необходимо создать базовый актив и распространить эту активность на все другие действия, чтобы вы могли изменить язык приложения только в одном месте:

```
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);
    }

}
```

Примечание. Всегда инициализируйте локаль в конструкторе.

Теперь вы можете использовать LocalizationActivity, как показано ниже.

```
public class MainActivity extends LocalizationActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

    }
}
```

Примечание. При изменении программного языка приложения необходимо перезапустить свою активность, чтобы изменить локальное изменение. Чтобы правильно работать для этого решения, вы и используйте локаль из общих настроек при запуске приложения, вы `android:name=".LocaleApp"` в вы Manifest.xml.

Иногда Lint checker запрашивает создание сборки релиза. Чтобы решить эту проблему, выполните нижеуказанные варианты.

Первый:

Если вы хотите отключить перевод для некоторых строк, то добавьте следующий атрибут в default string.xml

```
<string name="developer" translatable="false">Developer Name</string>
```

Во- **вторых:**

Игнорировать все отсутствующие переводы из файла ресурсов добавить следующий атрибут Это атрибут ignore пространства имен инструментов в файле строк, как показано ниже:

```
<?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>
```

В третьих:

Другой способ отключить непереводимую строку

http://tools.android.com/recent/non-translatablestrings

Если у вас много ресурсов, которые нельзя переводить, вы можете поместить их в файл с именем donottranslate.xml, и lint рассмотрит все из них непереводимые ресурсы.

В- **четвертых:**

Вы также можете добавить locale в файл ресурсов

```
<resources
xmlns:tools="http://schemas.android.com/tools"
    tools:locale="en" tools:ignore="MissingTranslation">
```

Вы также можете отключить проверку отсутствия перевода для lint из приложения / build.gradle

```
lintOptions {

        disable 'MissingTranslation'
    }
```

Прочитайте Локализация ресурсов на Android онлайн:
https://riptutorial.com/ru/android/topic/3345/локализация-ресурсов-на-android

глава 163: Локализованная дата / время в Android

замечания

Рекомендуется использовать методы класса DateUtils для форматирования дат, которые являются локальными, т. Е. Которые учитывают предпочтения пользователя (например, 12 h / 24h clock time formats). Эти методы наиболее подходят для дат, отображаемых пользователю.

Для полностью настроенных представлений даты рекомендуется использовать класс SimpleDateFormat , поскольку он позволяет полностью контролировать все элементы даты.

Examples

Пользовательский локализованный формат даты с DateUtils.formatDateTime ()

DateUtils.formatDateTime () позволяет вам указать время и на основе флагов, которые вы предоставляете, создает локализованную строку datetime. Флаги позволяют указать, включать ли определенные элементы (например, в будний день).

```
Date date = new Date();
String localizedDate = DateUtils.formatDateTime(context, date.getTime(),
DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_WEEKDAY);
```

formatDateTime () автоматически заботится о правильных форматах дат.

Стандартное форматирование даты и времени в Android

Формат даты:

```
Date date = new Date();
DateFormat df = DateFormat.getDateInstance(DateFormat.MEDIUM);
String localizedDate = df.format(date)
```

Отформатируйте дату и время. Дата в коротком формате, время в длинном формате:

```
Date date = new Date();
DateFormat df = DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.LONG);
String localizedDate = df.format(date)
```

Полностью настроенная дата / время

```
Date date = new Date();
df = new SimpleDateFormat("HH:mm", Locale.US);
String localizedDate = df.format(date)
```

Обычно используемые шаблоны:

- ЧЧ: час (0-23)
- чч: час (1-12)
- a: Маркер AM / PM
- мм: минута (0-59)
- ss: second
- dd: день в месяце (1-31)
- MM: месяц
- yyyy: year

Прочитайте Локализованная дата / время в Android онлайн:
https://riptutorial.com/ru/android/topic/6057/локализованная-дата---время-в-android

глава 164: Макеты

Вступление

Макет определяет визуальную структуру пользовательского интерфейса, такую как активность или виджет.

Макет объявляется в XML, включая элементы экрана, которые будут отображаться в нем. Код может быть добавлен в приложение для изменения состояния экранных объектов во время выполнения, включая те, которые объявлены в XML.

Синтаксис

- Android: гравитация = "сверху | снизу | влево | право | center_vertical | fill_vertical | center_horizontal | fill_horizontal | центр | заполнить | clip_vertical | clip_horizontal | начать | конец"
- андроид: layout_gravity = "сверху | снизу | влево | право | center_vertical | fill_vertical | center_horizontal | fill_horizontal | центр | заполнить | clip_vertical | clip_horizontal | начать | конец"

замечания

Атрибуты LayoutParams и Layout_

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

RelativeLayouts в верхней части иерархии представлений

Как объясняется в этой статье о производительности в Android , `RelativeLayout` требует, чтобы два макета пропуска отображались правильно. Для сложных иерархий представлений это может существенно повлиять на производительность. Вложение `RelativeLayouts` делает эту проблему еще хуже, потому что каждый `RelativeLayout` приводит к увеличению количества проходов макета.

Examples

LinearLayout

LinearLayout - это `ViewGroup` которая упорядочивает `ViewGroup` в одном столбце или в одной строке. Ориентацию можно задать, вызвав метод `setOrientation()` или используя атрибут xml `android:orientation` .

1. **Вертикальная ориентация** : `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>
```

Вот скриншот, как это будет выглядеть:

2. **Горизонтальная ориентация** : `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` также поддерживает назначение веса отдельным детям с атрибутом `android:layout_weight` .

RelativeLayout

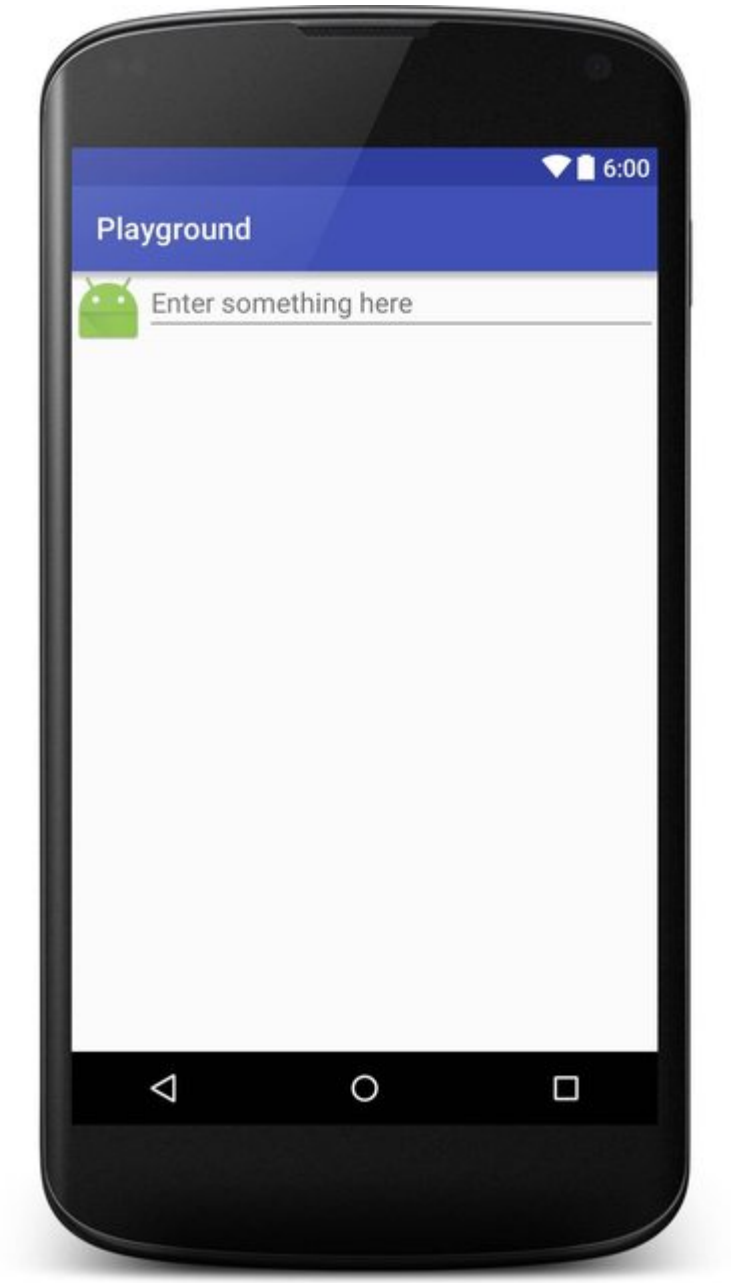
`RelativeLayout`

- это `ViewGroup` которая отображает дочерние представления в относительных положениях. По умолчанию все дочерние представления рисуются в верхнем левом углу макета, поэтому вы должны определить положение каждого вида, используя различные свойства макета, доступные из `RelativeLayout.LayoutParams` . Значение для каждого свойства макета является либо логическим, чтобы включить положение макета относительно родительского RelativeLayout, либо идентификатор, который ссылается на другое представление в макете, на которое должно быть расположено представление.

Пример:

```
<?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>
```

Вот скриншот, как это будет выглядеть:



Гравитация и гравитация макета

андроид: layout_gravity

- `android:layout_gravity` используется для установки положения элемента в его родительском элементе (например, дочерний `View` внутри `Layout`).
- Поддерживается LinearLayout и FrameLayout

Android: гравитация

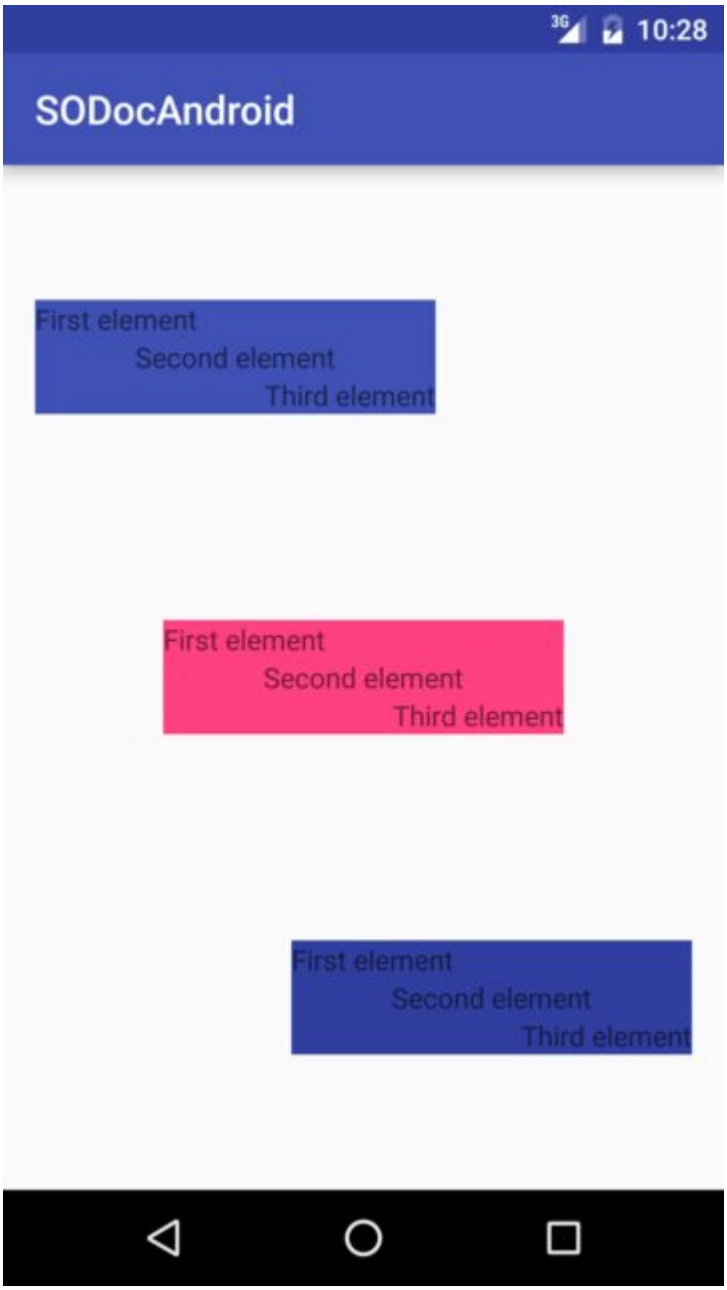
- `android:gravity` используется для установки положения содержимого внутри элемента (например, текста внутри `TextView`).

```
<?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>
```

Который получает визуализацию следующим образом:

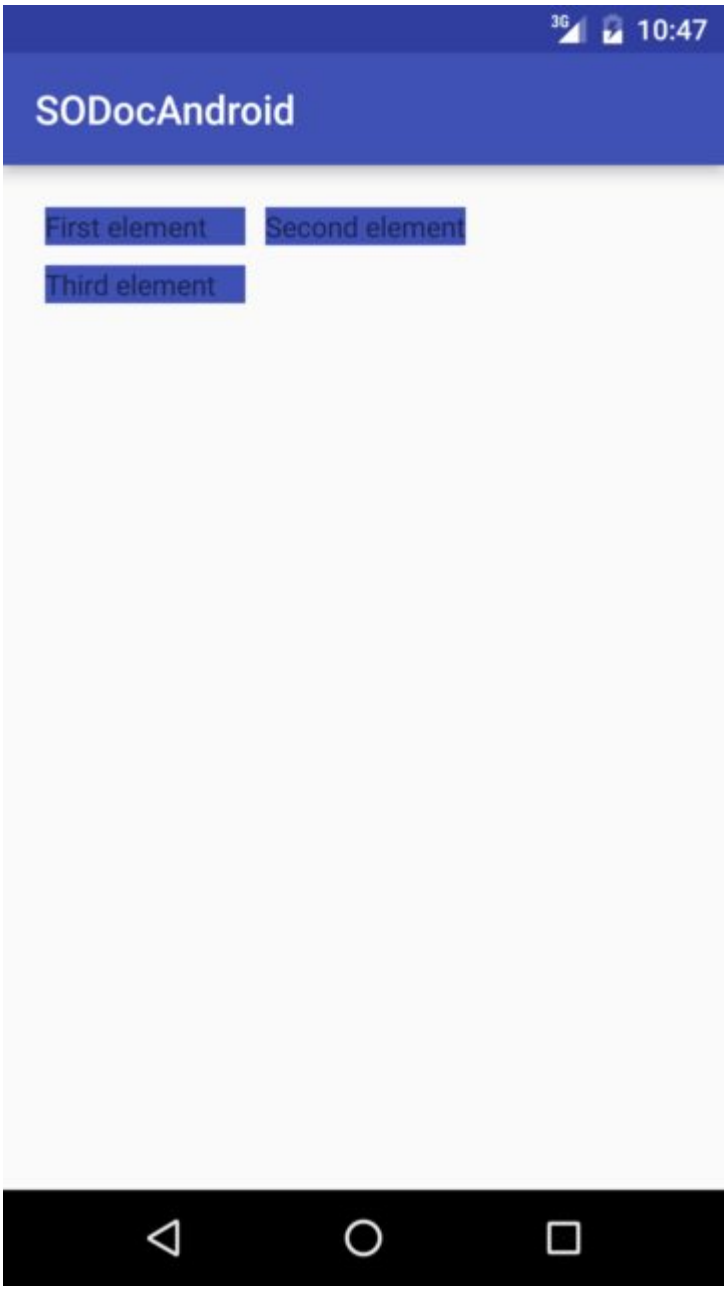


Макет сетки

GridLayout, как следует из названия, представляет собой макет, используемый для организации представлений в сетке. GridLayout делит себя на столбцы и строки. Как видно из приведенного ниже примера, количество столбцов и / или строк определяется свойствами `columnCount` и `rowCount` . Добавление представлений в этот макет добавит первое представление к первому столбцу, второе представление ко второму столбцу и третье представление к первому столбцу второй строки.

```
<?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>
```



Процентные макеты

2,3

Библиотека поддержки Percent предоставляет `PercentFrameLayout` и `PercentRelativeLayout` , две группы представлений, которые обеспечивают простой способ указания **размеров и полей** вида в **процентах** от общего размера.

Вы можете использовать библиотеку поддержки процентов, добавив следующее к вашим зависимостям.

```
compile 'com.android.support:percent:25.3.1'
```

Если вы хотите отобразить представление, которое заполняет экран горизонтально, но только половину экрана по вертикали, вы должны выполнить следующее.

```
<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>
```

Вы также можете определить проценты в отдельном файле XML с кодом, например:

```
<fraction name="margin_start_percent">25%</fraction>
```

И обратитесь к ним в своих макетах с помощью `@fraction/margin_start_percent` .

Они также содержат возможность установки пользовательского **формата изображения** через `app:layout_aspectRatio` .
Это позволяет вам установить только один размер, например, только ширину, и высота будет автоматически определяться на основе указанного вами формата, будь то 4: 3 или 16: 9 или даже квадрата 1: 1 соотношение сторон.

Например:

```
 <ImageView
    app:layout_widthPercent="100%"
    app:layout_aspectRatio="178%"
    android:scaleType="centerCrop"
    android:src="@drawable/header_background"/>
```

FrameLayout

`FrameLayout` предназначен для блокировки области на экране для отображения одного элемента. Тем не менее, вы можете добавить несколько детей в FrameLayout и управлять своей позицией в FrameLayout, назначив гравитацию каждому ребенку, используя атрибут android: layout_gravity .

Как правило, `FrameLayout` используется для хранения одного дочернего представления. Обычные случаи использования создают владельцы мест для раздувания `Fragments` в `Activity` , перекрытия представлений или применения переднего плана к представлениям.

Пример:

```
<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>
```

Это будет выглядеть так:

CoordinatorLayout

2,3

`CoordinatorLayout` представляет собой контейнер, несколько похожий на `FrameLayout` но с дополнительными возможностями, он называется сверхмощным `FrameLayout` в официальной документации.

Присоединив `CoordinatorLayout.Behavior` к прямому ребенку CoordinatorLayout, вы сможете перехватывать события касания, вставки окна, измерение, макет и вложенную прокрутку.

Чтобы использовать его, вам сначала нужно добавить зависимость для библиотеки поддержки в файле gradle:

```
compile 'com.android.support:design:25.3.1'
```

Номер последней версии библиотеки можно найти здесь.

Одним из практических примеров использования `CoordinatorLayout` является создание представления с помощью `FloatingActionButton` . В этом конкретном случае мы создадим `RecyclerView` с `SwipeRefreshLayout` и `FloatingActionButton` поверх этого. Вот как вы можете это сделать:

```
<?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>
```

Обратите внимание, как FloatingActionButton привязан к координаторуLayout с

`app:layout_anchor="@id/coord_layout"`

CoordinatorLayout Scrolling Behavior

2.3-2.3.2

Встроенный `CoordinatorLayout` может использоваться для достижения эффектов прокрутки материала при использовании внутренних макетов, поддерживающих вложенную прокрутку, таких как `NestedScrollView` или `RecyclerView` .

Для этого примера:

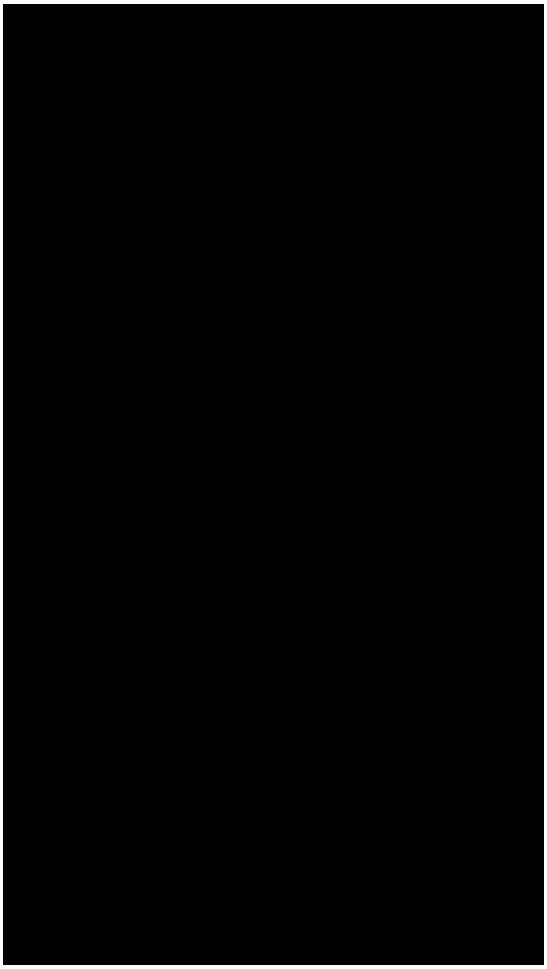
- `app:layout_scrollFlags="scroll|enterAlways"` используется в свойствах панели инструментов
- `app:layout_behavior="@string/appbar_scrolling_view_behavior"` используется в свойствах ViewPager
- В фрагментах ViewPager используется RecyclerView

Вот XML-файл макета, используемый в Activity:

```
<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>
```

Результат:



Посмотреть Вес

Одним из наиболее используемых атрибутов для LinearLayout является вес его дочерних представлений. Вес определяет объем пространства, который будет потребляться по сравнению с другими видами в LinearLayout.

Вес используется, когда вы хотите предоставить определенное пространство экрана одному компоненту по сравнению с другим.

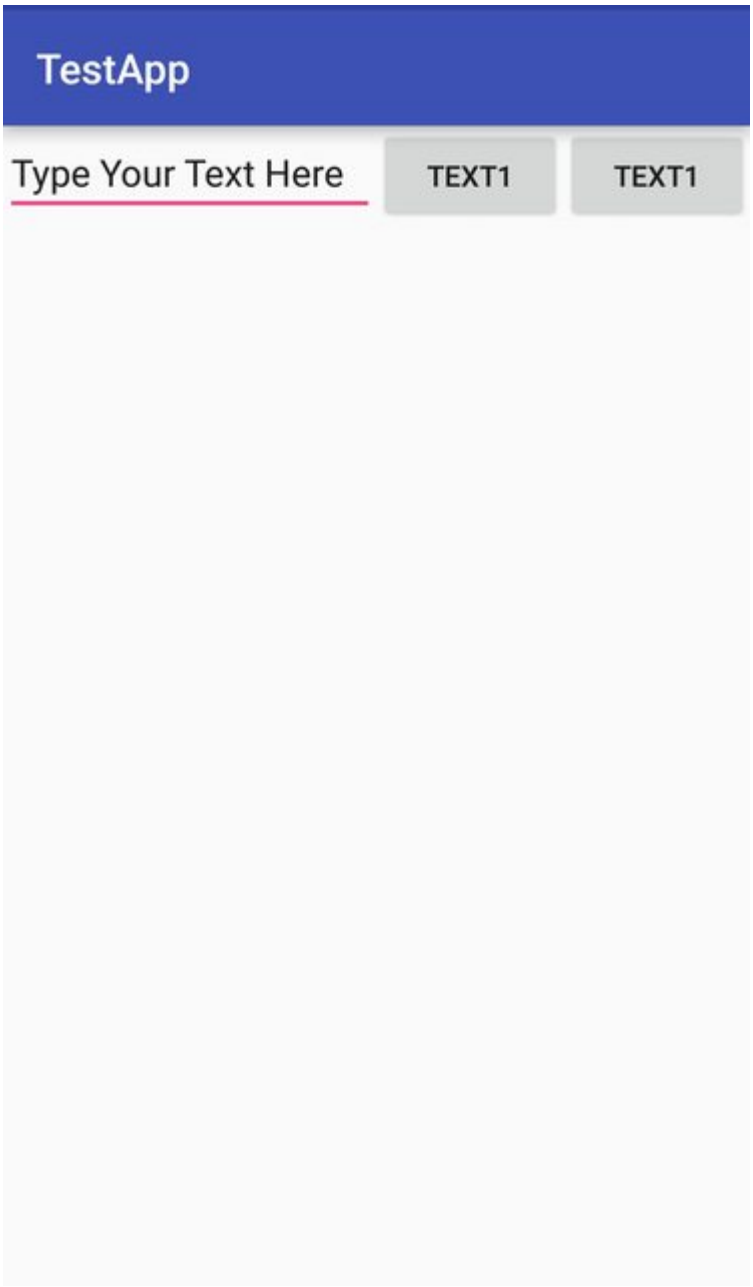
Ключевые свойства :

- `weightSum` - общая сумма весов всех детских просмотров. Если вы не укажете `weightSum` , система сама рассчитает сумму всех весов.
- `layout_weight` определяет объем пространства из общей суммы веса, которую займет виджет.

Код:

```
<?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>
```

Выход:



Теперь, даже если размер устройства больше, EditText займет 2/4 экрана. Следовательно, внешний вид вашего приложения воспринимается последовательно на всех экранах.

Примечание. Здесь `layout_width` поддерживается `0dp` поскольку пространство виджета разделяется по горизонтали. Если виджеты должны быть выровнены по вертикали, то `layout_height` будет установлен в `0dp` . Это делается для повышения эффективности кода, потому что во время выполнения система не будет пытаться вычислять ширину или высоту соответственно, так как это управляется весом. Если вы вместо этого используете `wrap_content` система `wrap_content` попытается вычислить ширину / высоту перед применением атрибута weight, который вызывает другой цикл вычисления.

Создание программного обеспечения LinearLayout

иерархия

```
- LinearLayout(horizontal)
   - ImageView
   - LinearLayout(vertical)
     - TextView
     - TextView
```

Код

```
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

Каждая отдельная группа `ViewGroup` (например, `LinearLayout` , `RelativeLayout` , `CoordinatorLayout` и т. Д.) `LinearLayout` хранить информацию о своих свойствах своих детей. О том, как его дети выкладываются в `ViewGroup` . Эта информация хранится в объектах класса-оболочки класса `ViewGroup.LayoutParams` .

Чтобы включить параметры, специфичные для конкретного типа макета, в `ViewGroups` используются подклассы класса `ViewGroup.LayoutParams` .

Например, для

- `LinearLayout` это `LinearLayout.LayoutParams`
- `RelativeLayout` это `RelativeLayout.LayoutParams`
- `CoordinatorLayout` это `CoordinatorLayout.LayoutParams`
- ...

Большинство `ViewGroups` повторно используют возможность установки `margins` для своих детей, поэтому они не подклассы `ViewGroup.LayoutParams` напрямую, но вместо этого они представляют собой подкласс `ViewGroup.MarginLayoutParams` (который сам является подклассом `ViewGroup.LayoutParams`).

`LayoutParams` в xml

Объекты `LayoutParams` создаются на основе раздутого `xml` файла макета.

```
<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>
```

Все параметры , которые начинаются с `layout_` указать , как макет **вмещающих** должен работать. Когда компоновка надуваются, эти параметры обернуты в надлежащем `LayoutParams` объект, который позже будет использоваться в `Layout` , чтобы правильно позиционировать конкретный `View` внутри `ViewGroup` . Другие атрибуты `View` непосредственно `View` информации о связанных и обрабатываются `View` самого.

Для `TextView` :

- `layout_width` , `layout_height` и `layout_gravity` будут храниться в объекте `LinearLayout.LayoutParams` и использоваться `LinearLayout`
- `gravity` , `text` и `textColor` будут использоваться самим `TextView`

Для `ImageView` :

- `layout_width` , `layout_height` и `layout_weight` будут сохранены в объекте `LinearLayout.LayoutParams` и использованы `LinearLayout`
- `background` , `scaleType` и `src` будут использоваться самим `ImageView`

Получение объекта `LayoutParams`

`getLayoutParams` - это метод `View's` который позволяет извлекать текущий объект `LayoutParams` .

Поскольку объект `LayoutParams` напрямую связан с **охватывающей** `ViewGroup` , этот метод

возвращает ненулевое значение только тогда, когда `View` подключен к `ViewGroup` . Вы должны иметь в виду, что этот объект может отсутствовать во все времена. Особенно вы не должны зависеть от наличия внутри конструктора `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();
        }
    }

    //...
}
```

Если вы хотите , чтобы зависеть от наличия `LayoutParams` объекта, вы должны использовать `onAttachedToWindow` метод.

```
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();
        }
    }

    //...
}
```

Объект Casting `LayoutParams`

Возможно, вам придется использовать функции, специфичные для конкретной группы `ViewGroup` (например, вы можете программно изменить правила `RelativeLayout`). Для этого вам нужно будет знать, как правильно `ViewGroup.LayoutParams` объект `ViewGroup.LayoutParams` .

Это может быть немного запутанным при получении объекта `LayoutParams` для дочернего `View` который фактически является другой `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>
```

ВНИМАНИЕ: тип `LayoutParams` объекта напрямую связан с типом **вмещающих** `ViewGroup` .

***Неправильное** литье* :

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
FrameLayout.LayoutParams par = (FrameLayout.LayoutParams) innerLayout.getLayoutParams();
                                        // INCORRECT! This will produce ClassCastException
```

***Правильное** литье* :

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
LinearLayout.LayoutParams par = (LinearLayout.LayoutParams) innerLayout.getLayoutParams();
                                       // CORRECT! the enclosing layout is a LinearLayout
```

Прочитайте Макеты онлайн: https://riptutorial.com/ru/android/topic/94/макеты

глава 165: Мгновенный запуск в Android Studio

замечания

Instant Run - это расширенное поведение для команд запуска и отладки, которые позволяют быстрее отлаживать, не требуя полной сборки и переустановки для изменения eevry, выполненного в коде вашего приложения.

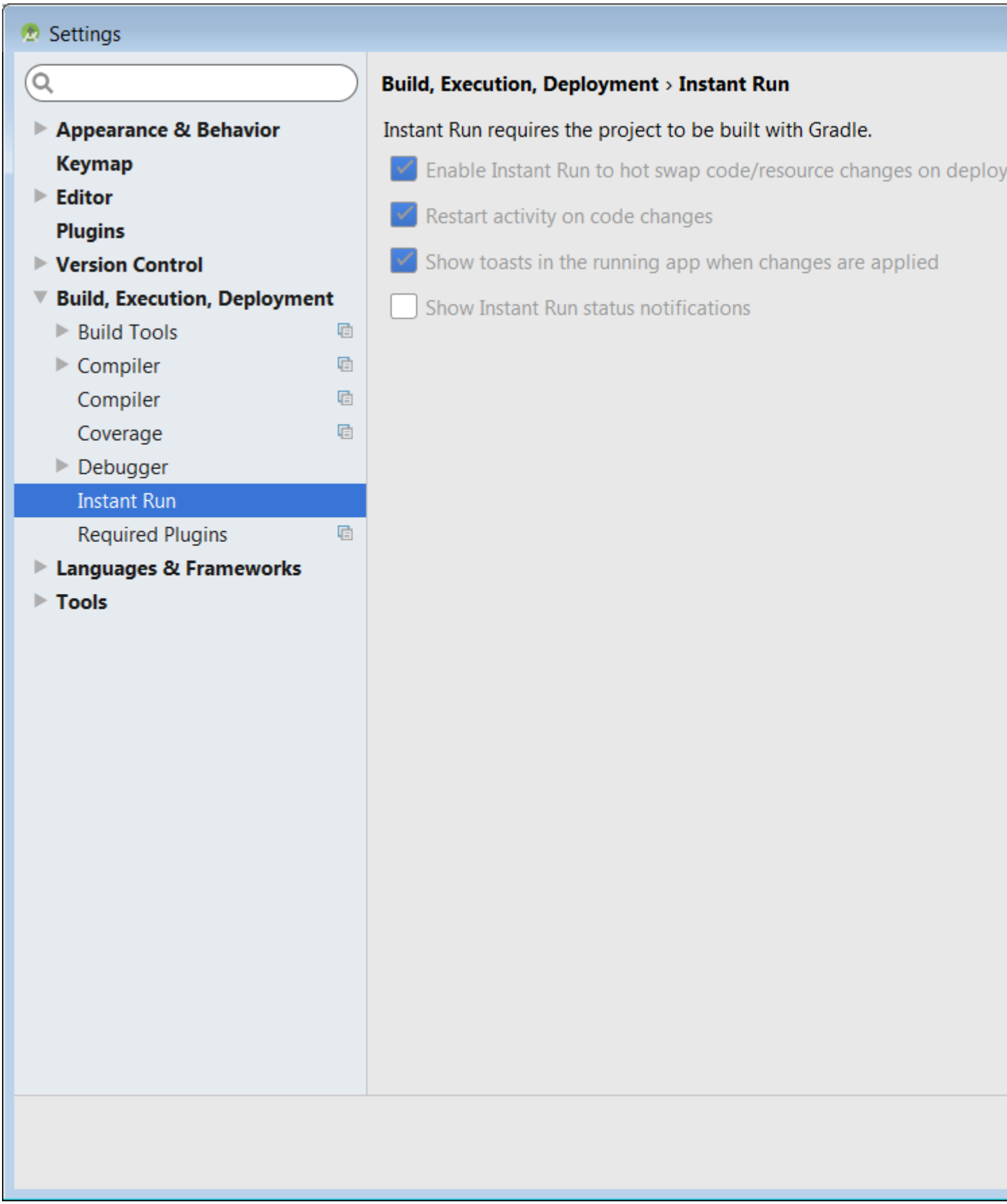
> Представленный в Android Studio 2.0, Instant Run - это поведение команд Run и Debug, которые значительно сокращают время между обновлениями вашего приложения. Хотя ваша первая сборка может занять больше времени, Instant Run подталкивает последующие обновления к вашему приложению без создания нового APK, поэтому изменения видны намного быстрее.
>
> Instant Run поддерживается только при развертывании варианта сборки отладки, используйте Android Plugin для Gradle версии 2.0.0 или выше и установите minSdkVersion на 15 или выше в файле build.gradle на уровне модуля вашего приложения. Для достижения максимальной производительности установите minSdkVersion на 21 или выше.
>
> После развертывания приложения в кнопке «Запуск» (или кнопку «Отладка») появляется маленькая желтая кнопка «Громовой удар», указывающая, что «Мгновенный запуск» готов нажать обновления при следующем нажатии кнопки. Вместо создания нового APK он подталкивает только эти новые изменения, и в некоторых случаях приложение даже не нужно перезапускать, но сразу же показывает эффект этих изменений кода.
>
> Instant Run подталкивает обновленный код и ресурсы к подключенному устройству или эмулятору, выполняя «горячую» замену, «теплую» или «холодную» замену. Он автоматически определяет тип свопа для выполнения в зависимости от типа изменения, которое вы сделали. Видео выше содержит интересную информацию о том, как все это работает под капотом. Однако для краткого описания того, как работает Instant Run, когда вы нажимаете определенные изменения кода на целевое устройство, обратитесь к следующей таблице.

Документация

Examples

Включение или отключение Instant Run

1. Откройте диалоговое окно «Настройки» или «Настройки»:
    - В Windows или Linux выберите « `File` > « `Settings` в главном меню.
    - В Mac OSX выберите « `Android Studio` > « `Preferences` в главном меню.
2. Перейдите к `Build, Execution, Deployment` > `Compiler` .
3. В текстовом поле рядом с параметрами командной строки введите параметры командной строки.
4. Нажмите «OK», чтобы сохранить и выйти.



Верхний вариант - мгновенный запуск. Установите / снимите этот флажок.

Документация

Типы кода свопы в мгновенном запуске

Существует три типа свопов кода, которые мгновенный запуск позволяет поддерживать более быструю отладку и запуск приложения из вашего кода в Android Studio.

- Горячая замена
- Теплый обмен
- Холодная замена

Когда срабатывает каждый из этих свопов?

HOT SWAP запускается при изменении существующего метода.

WARM SWAP запускается, когда существующий ресурс изменяется или удаляется (что-либо в папке res)

COLD SWAP при изменении структурного кода в коде приложения, например

1. Добавить, удалить или изменить:

- аннотация
- поле экземпляра
- статическое поле
- подпись статического метода
- подпись метода экземпляра

2. Измените, какой родительский класс наследует текущий класс
3. Измените список реализованных интерфейсов
4. Изменение статического инициализатора класса
5. Переупорядочить элементы макета, использующие динамические идентификаторы ресурсов

Что происходит, когда происходит обмен кода?

Изменения **HOT SWAP** видны мгновенно - как только будет сделан следующий вызов метода, реализация которого изменена.

WARM SWAP перезапускает текущую деятельность

COLD SWAP перезапускает все приложение (без переустановки)

Неподдерживаемый код изменяется при использовании Instant Run

Есть несколько изменений, когда мгновение не будет делать трюк, а полная сборка и повторная установка вашего приложения произойдет так, как это было раньше, до того, как появился Instant Run.

1. Измените манифест приложения
2. Изменение ресурсов, на которые ссылается манифест приложения
3. Изменение элемента пользовательского интерфейса виджета Android (требуется очистка и повтор)

Документация

Прочитайте Мгновенный запуск в Android Studio онлайн:
https://riptutorial.com/ru/android/topic/2119/мгновенный-запуск-в-android-studio

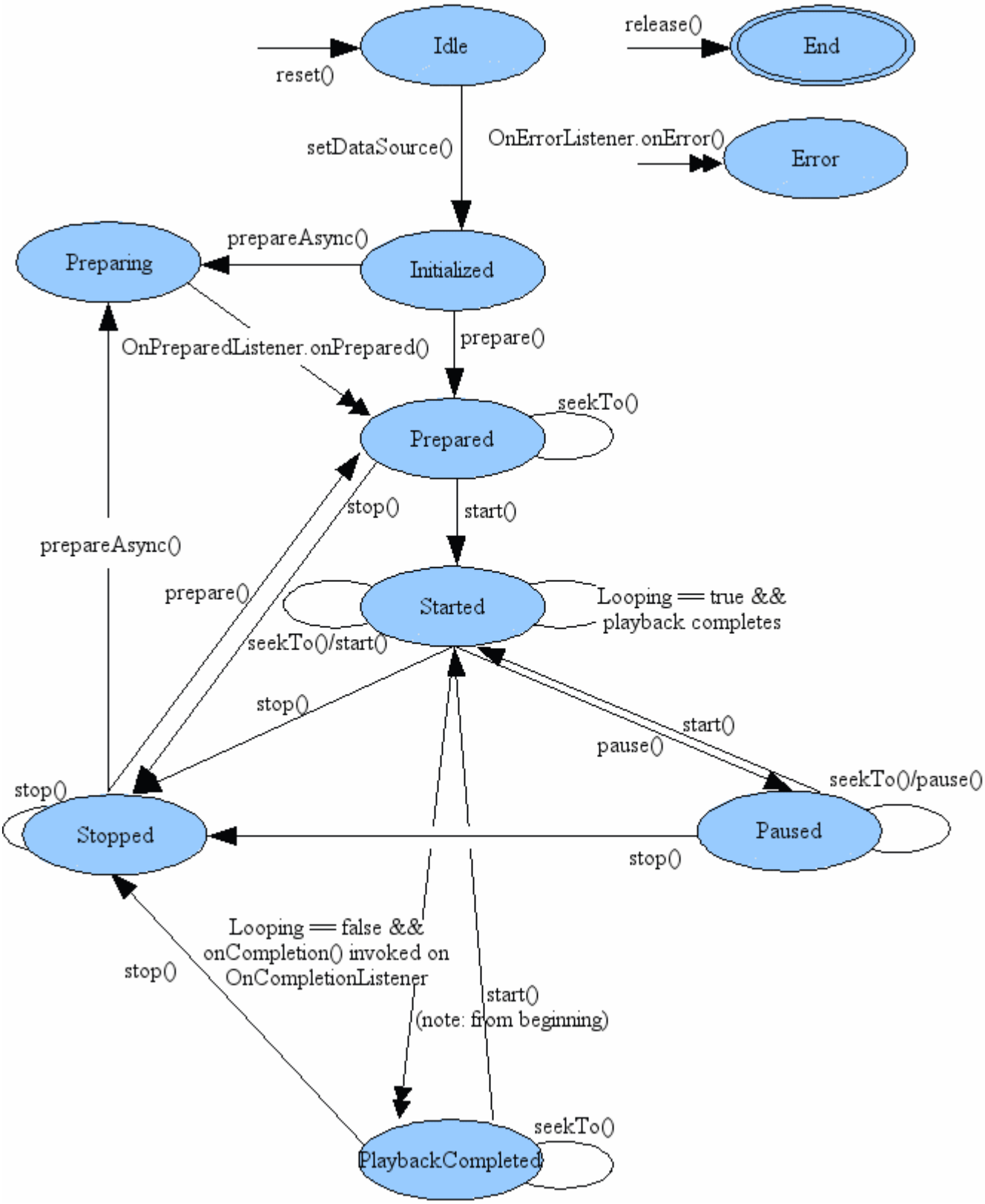
глава 166: Медиа-плейер

Синтаксис

- void setAudioStreamType (int streamtype)
- void setDataSource (контекст контекста, Uri uri)
- void prepare ()
- void prepareAsync ()
- void start ()
- void stop ()

замечания

Использование `MediaPlayer` в основном основано на диаграмме состояний:



Это означает, что для воспроизведения аудио / видео должна выполняться определенная последовательность действий, это конкретный порядок. В нем также указывается, какие действия можно предпринять в каком состоянии .

API MediaPlayer не обладает гибкостью (добавляет пользовательский декодер и визуализацию) и не имеет sSupport для динамической адаптивной потоковой передачи через HTTP (DASH) и SmoothStreaming. Для этого загляните в ExoPlayer .

Examples

Базовое создание и воспроизведение

Класс MediaPlayer может использоваться для управления воспроизведением аудио / видео файлов и потоков.

Создание объекта MediaPlayer может быть трех типов:

1. Средства массовой информации из местных ресурсов

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.resource);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

2. Из локального URI (полученного от ContentResolver)

```
Uri myUri = ....; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

3. Из внешнего URL-адреса

```
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();
```

Асинхронная подготовка

`MediaPlayer$prepare()` является блокирующим вызовом и блокирует пользовательский интерфейс до завершения выполнения. Для решения этой проблемы можно использовать `MediaPlayer$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
```

При синхронных операциях ошибки обычно сигнализируются с исключением или кодом ошибки, но всякий раз, когда вы используете асинхронные ресурсы, вы должны убедиться,

что ваше приложение уведомлено об ошибках соответствующим образом. Для 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
        }
});
```

Получение системных мелодий

В этом примере показано, как получить URI для системных рингтонов (`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;
    }
```

Список зависит от типа запрошенных мелодий звонка. Возможности:

- `RingtoneManager.TYPE_RINGTONE`
- `RingtoneManager.TYPE_NOTIFICATION`
- `RingtoneManager.TYPE_ALARM`
- `RingtoneManager.TYPE_ALL = TYPE_RINGTONE | TYPE_NOTIFICATION | TYPE_ALARM`

Чтобы получить Ringtones как `android.media.Ringtone` каждый `Uri` должен быть разрешен `RingtoneManager` :

```
android.media.Ringtone osRingtone = RingtoneManager.getRingtone(context, uri);
```

Чтобы воспроизвести звук, используйте метод:

```
public void setDataSource(Context context, Uri uri)
```

от `android.media.MediaPlayer` . `MediaPlayer` должен быть инициализирован и подготовлен в соответствии с диаграммой состояний

Получение и настройка объема системы

Типы аудиопотоков

Существуют различные профили потоков рингтонов. Каждый из них имеет разный объем.

Каждый пример здесь написан для `AudioManager.STREAM_RING` потока `AudioManager.STREAM_RING` . Однако это не единственный. Доступные типы потоков:

- `STREAM_ALARM`
- `STREAM_DTMF`
- `STREAM_MUSIC`
- `STREAM_NOTIFICATION`
- `STREAM_RING`
- `STREAM_SYSTEM`
- `STREAM_VOICE_CALL`

Настройка громкости

Чтобы получить объем определенного профиля, звоните:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int currentVolume = audioManager.getStreamVolume(AudioManager.STREAM_RING);
```

Это значение очень мало полезно, когда максимальное значение для потока неизвестно:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int streamMaxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM_RING);
```

Отношение этих двух значений даст относительный объем (0 <объем <1):

```
float volume = ((float) currentVolume) / streamMaxVolume
```

Регулировка громкости на один шаг

Чтобы увеличить громкость для потока выше на один шаг, вызовите:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_RAISE, 0);
```

Чтобы уменьшить громкость для потока на один шаг, вызовите:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_LOWER, 0);
```

Настройка MediaPlayer для использования определенного типа потока

Для этого есть вспомогательная функция класса `MediaPlayer` .
Просто вызовите `void setAudioStreamType(int streamtype)` :

```
MediaPlayer mMedia = new MediaPlayer();
mMedia.setAudioStreamType(AudioManager.STREAM_RING);
```

Медиаплеер с ходом и положением воспроизведения буфера

```
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);
                }
```

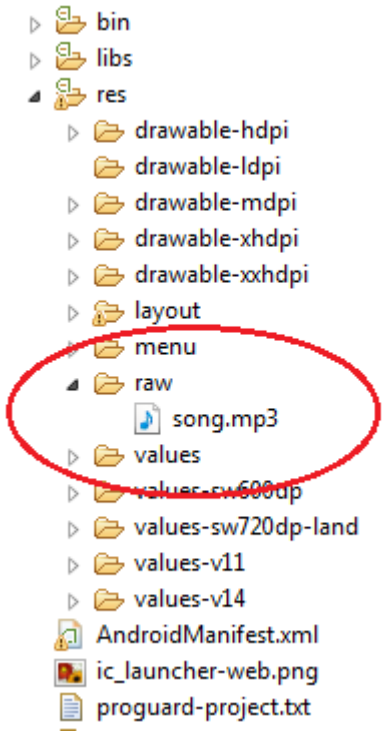
```
            }
        });

        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>
```

Импортируйте аудио в androidstudio и воспроизводите его

Это пример того, как получить воспроизведение аудиофайла, который у вас уже есть на вашем ПК / ноутбуке. Сначала создайте новый каталог под res и назовите его как raw



скопируйте аудио, которое вы хотите воспроизвести в эту папку. Возможно, это файл .mp3 или .wav.

Теперь, например, при нажатии кнопки вы хотите воспроизвести этот звук, вот как это делается

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

Это будет воспроизводить песню только один раз, когда нажимается кнопка, если вы хотите воспроизвести песню на каждой кнопке, нажмите код записи, подобный этому

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

Прочитайте Медиа-плейер онлайн: https://riptutorial.com/ru/android/topic/1851/медиа-плейер

глава 167: Меню

Синтаксис

- inflater.inflate (R.menu.your_xml_file, меню);

параметры

параметр	Описание
`inflate(int menuRes, Menu menu)`	Наполните иерархию меню из указанного ресурса XML.
`getMenuInflater ()`	Возвращает `MenuInflater` с этим контекстом.
`onCreateOptionsMenu (Menu menu)`	Инициализируйте содержимое меню стандартных параметров Activity. Вы должны поместить элементы меню в меню.
`onOptionsItemSelected (MenuItem item)`	Этот метод вызывается всякий раз, когда выбран пункт в меню параметров.

замечания

Чтобы узнать больше о **меню** , прочтите это . Надеюсь, поможет!

Examples

Меню опций с разделителями

В Android есть меню параметров по умолчанию, которое может принимать несколько параметров. Если необходимо отобразить большее количество параметров, то имеет смысл группировать эти параметры, чтобы сохранить ясность. Параметры можно группировать, помещая между ними разделители (то есть горизонтальные линии). Чтобы разрешить разделители, можно использовать следующую тему:

```
<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>
```

Изменив тему, в меню можно добавить разделители.

Применить пользовательский шрифт в меню

```
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);
}
```

а затем в Управлении:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    applyFontToMenu(menu,this);
    return true;
}
```

Создание меню в действии

Чтобы определить свое собственное меню, создайте XML-файл в каталоге `res/menu/` вашего проекта и создайте меню со следующими элементами:

- `<menu>` : Определяет Меню, в котором хранятся все пункты меню.
- `<item>` : Создает MenuItem, который представляет один элемент в меню. Мы также можем создать вложенный элемент для создания подменю.

Шаг 1:

Создайте свой собственный XML-файл следующим образом:

В `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>
```

Шаг 2:

Чтобы указать меню параметров, переопределите `onCreateOptionsMenu()` в своей *деятельности* .

В этом методе вы можете раздуть свой ресурс меню (определенный в вашем файле XML, то есть `res/menu/main_menu.xml`)

```
 @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }
```

Когда пользователь выбирает элемент из меню параметров, система вызывает переопределенный `onOptionsItemSelected()` вашей *активности* .

- Этот метод передает выбранный MenuItem.
- Вы можете идентифицировать элемент, вызвав `getItemId()` , который возвращает уникальный идентификатор элемента меню (определяемый `android:id attribute` в ресурсе меню - `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);
    }
}
```

Обертывание!

Код вашей `Activity` должен выглядеть следующим образом:

```
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);
        }
    }
}
```

Снимок экрана о том, как выглядит ваше собственное меню:

Прочитайте Меню онлайн: https://riptutorial.com/ru/android/topic/2028/меню

глава 168: Место нахождения

Вступление

API Android Location используются в самых разных приложениях для различных целей, таких как поиск местоположения пользователя, уведомление о том, когда пользователь покинул общую область (Geofencing), и помочь интерпретировать активность пользователя (ходьба, работа, вождение и т. Д.).

Однако API-интерфейсы Android не являются единственным средством приобретения местоположения пользователя. Ниже приводятся примеры использования Android `LocationManager` и других распространенных библиотек местоположений.

замечания

Для создания приложений с поддержкой местоположения на Android существует два пути:

- Исходный исходный код `LocationManager` Android
- Google `FusedLocationProviderApi` , входящий в состав Google Play Services

LocationManager

Pros

- Более гранулированный контроль
- Доступно на всех устройствах
- Часть платформы Android

Cons

- Слив батареи - проблема, если ее не управляют должным образом
- Требуется логика для переключения поставщиков местоположения, если устройство не может найти местоположение (например, плохой GPS внутри здания)

Характеристики

- Приемник NMEA
- Приемник состояния GPS
- Слушайте изменения статуса поставщика (например, GPS отключается пользователем)
- Список поставщиков для выбора источника местоположения

Провайдеры

GPS

- Необходимые разрешения:
  - `ACCESS_FINE_LOCATION`
- Точность: 10 м - 100 м
- Требования к питанию: HIGH
- Доступность: во всем мире (с ясным видом на небо)
- **ПРИМЕЧАНИЯ** :
  - Обновления местоположения обычно поступают один раз в секунду, но в ситуациях, когда GPS не используется в течение некоторого времени, и A-GPS недоступен, для получения местоположения требуется несколько минут.
  - В тех случаях, когда затруднено четкое представление о небе, точки GPS не будут группироваться очень хорошо (точки места «прыгать»), и точность может вводить в заблуждение в определенных областях из-за эффекта « Городской каньон ».

сеть

- Необходимые разрешения:
  - `ACCESS_COARSE_LOCATION` или `ACCESS_FINE_LOCATION`
- Точность: 100 м - 1000 м +
- Требования к питанию: LOW - MEDIUM
- Доступность: В пределах диапазона ячеек башни или сигнала Wi-Fi
- **ЗАМЕТКИ:**
  - Обновления местоположения происходят реже, чем GPS
  - Обновления местоположения обычно не кластеризуются (точки места «прыгать»), и точность может варьироваться в зависимости от количества различных факторов (количество сигналов Wi-Fi, уровень сигнала, тип ячейки башни и т. Д.),

пассивный

- Необходимые разрешения:
  - `ACCESS_FINE_LOCATION`
- Точность: 10 м - 1000 м +
- Требования к питанию: НЕТ
- Доступность: только когда другое приложение получает местоположение от GPS или сети
- **ЗАМЕТКИ:**
  - Не полагайтесь на это, чтобы постоянно обновлять его. Это пассивно слушает другие приложения, которые делают запросы на местоположение, и передает их обратно.
  - Не возвращает FagedLocationProviderApi сгенерированные точки, только базовые точки местоположения, используемые для их создания.

FusedLocationProviderApi

Pros

- Предлагает меньше разряда аккумулятора «из коробки»
- Обрабатывает плохой GPS-приемник
- Получает обновления более регулярно

Cons

- Менее детальный контроль над GPS
- Может быть недоступен на всех устройствах или в некоторых странах
- Требуется зависимость сторонней библиотеки

Характеристики

- Хорошо управляемое использование поставщиков местоположения для оптимальной экономии тесто
- Обычно генерирует более точные точки, а затем Network Location Provider
- Более частые обновления библиотеки, позволяющие улучшить
- Нет необходимости указывать, какой тип поставщика использовать

Приоритетные уровни местоположения

PRIORITY_HIGH_ACCURACY

- Необходимые разрешения:
  - `ACCESS_FINE_LOCATION` для более точного местоположения или `ACCESS_COARSE_LOCATION` для менее точного местоположения
- Точность: 10 м - 100 м
- Требования к питанию: HIGH
- Доступность: везде, где доступны службы Google Play.
- **ЗАМЕТКИ:**
  - Если `ACCESS_FINE_LOCATION` не используется, это не будет использовать GPS для генерации обновлений местоположения, но все равно найдет довольно точную точку в правильных условиях.
  - Если `ACCESS_FINE_LOCATION` используется, он может или не может использовать GPS для создания точек местоположения, в зависимости от того, насколько точно он может отслеживать устройство в соответствии с условиями окружающей среды.
  - Хотя это может сообщать о более точном обновлении местоположения, чем другие настройки, оно все еще подвержено эффекту « Городской каньон ».

PRIORITY_BALANCED_POWER_ACCURACY

- Необходимые разрешения:
  - `ACCESS_FINE_LOCATION` для более точного местоположения или `ACCESS_COARSE_LOCATION` для менее точного местоположения
- Точность: 100 м - 1000 м +
- Требования к питанию: СРЕДНЯЯ
- Доступность: везде, где доступны службы Google Play.
- **ЗАМЕТКИ:**
  - Такие же заметки, как `PRIORITY_HIGH_ACCURACY`
  - Хотя это маловероятно, этот параметр может по-прежнему использовать GPS для создания местоположения.

PRIORITY_LOW_POWER

- Необходимые разрешения:
  - `ACCESS_FINE_LOCATION` или `ACCESS_COARSE_LOCATION`
- Точность: 100 м - 1000 м +
- Требования к питанию: LOW
- Доступность: везде, где доступны службы Google Play.
- **ЗАМЕТКИ:**
  - Вероятно, он не использует GPS, но пока не проверен.
  - Обновления обычно не очень точны
  - Обычно используется для обнаружения значительных изменений в местоположении

PRIORITY_NO_POWER

- Необходимые разрешения:
  - `ACCESS_FINE_LOCATION` или `ACCESS_COARSE_LOCATION`
- Точность: 10 м - 1000 м +
- Требования к питанию: НЕТ
- Доступность: везде, где доступны службы Google Play.
- **ЗАМЕТКИ:**
  - Функции почти идентичны с `LocationManager PASSIVE_PROVIDER`
  - `PASSIVE_PROVIDER` отчеты об обновлениях Служб Google Play, когда `PASSIVE_PROVIDER` сообщает о возвращенных базовых обновлениях местоположения

Поиск проблемы

OnLocationChanged () никогда не вызывается

Поскольку это, похоже, является общей проблемой при получении Android Locations, я изложу быстрый контрольный список общих исправлений:

---

1. Проверьте свой манифест!

   Одна из наиболее распространенных проблем заключается в том, что правильные разрешения никогда не предоставлялись. Если вы используете GPS (с сетью или без нее), используйте `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>` use `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>` , иначе используйте `<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>` , Для FuseLocationApi от Google требуется `ACCESS_FINE_LOCATION` .

---

2. (Для Android 6+) Проверьте разрешения во время выполнения !

   Проверьте и запросите разрешения! Если вам никогда не дадут разрешения, вы столкнетесь с аварийными ситуациями или, что еще хуже (если вы поймаете все исключения), вы ничего не увидите! Не имеет значения, предоставит ли пользователь разрешение в начале приложения, всегда проверяйте, есть ли у вас разрешения для всех вызовов. Пользователь может легко перейти к своим настройкам и отозвать их.

---

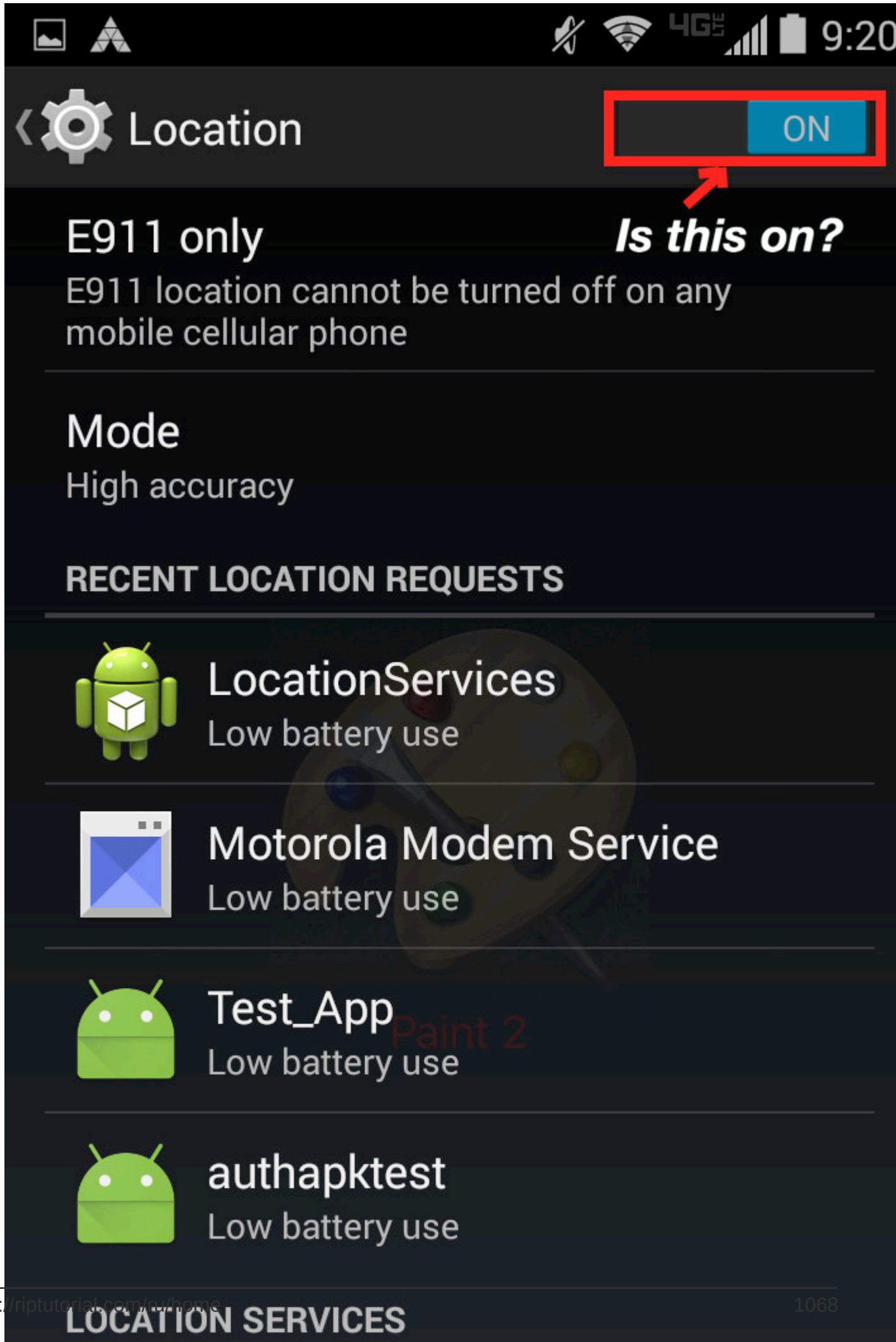
3. Дважды проверьте свой код!

   Вы уверены, что проходите в правом слушателе? `IntentService` ли вы, что `BroadcastReceiver` или `IntentService` к вашему манифесту? Используете ли вы `PendingIntent.getService()` в классе `BroadcastReceiver` или `getBroadcast()` в классе `IntentService` ? Вы уверены, что не регистрируете своего слушателя где-то еще в своем коде сразу после запроса?

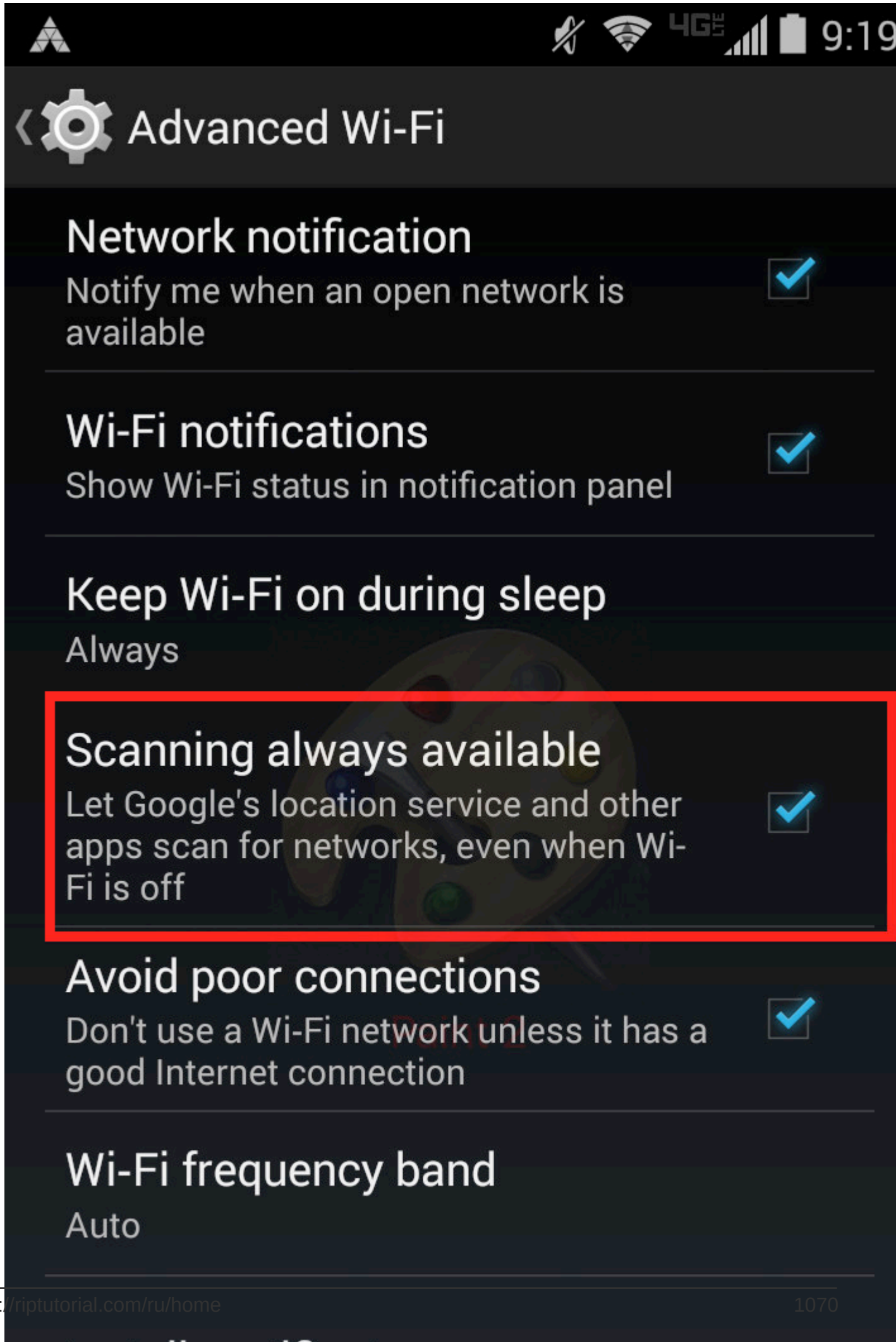
---

4. Проверьте настройки устройства!

   Очевидно, убедитесь, что вы включили службы определения местоположения.


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

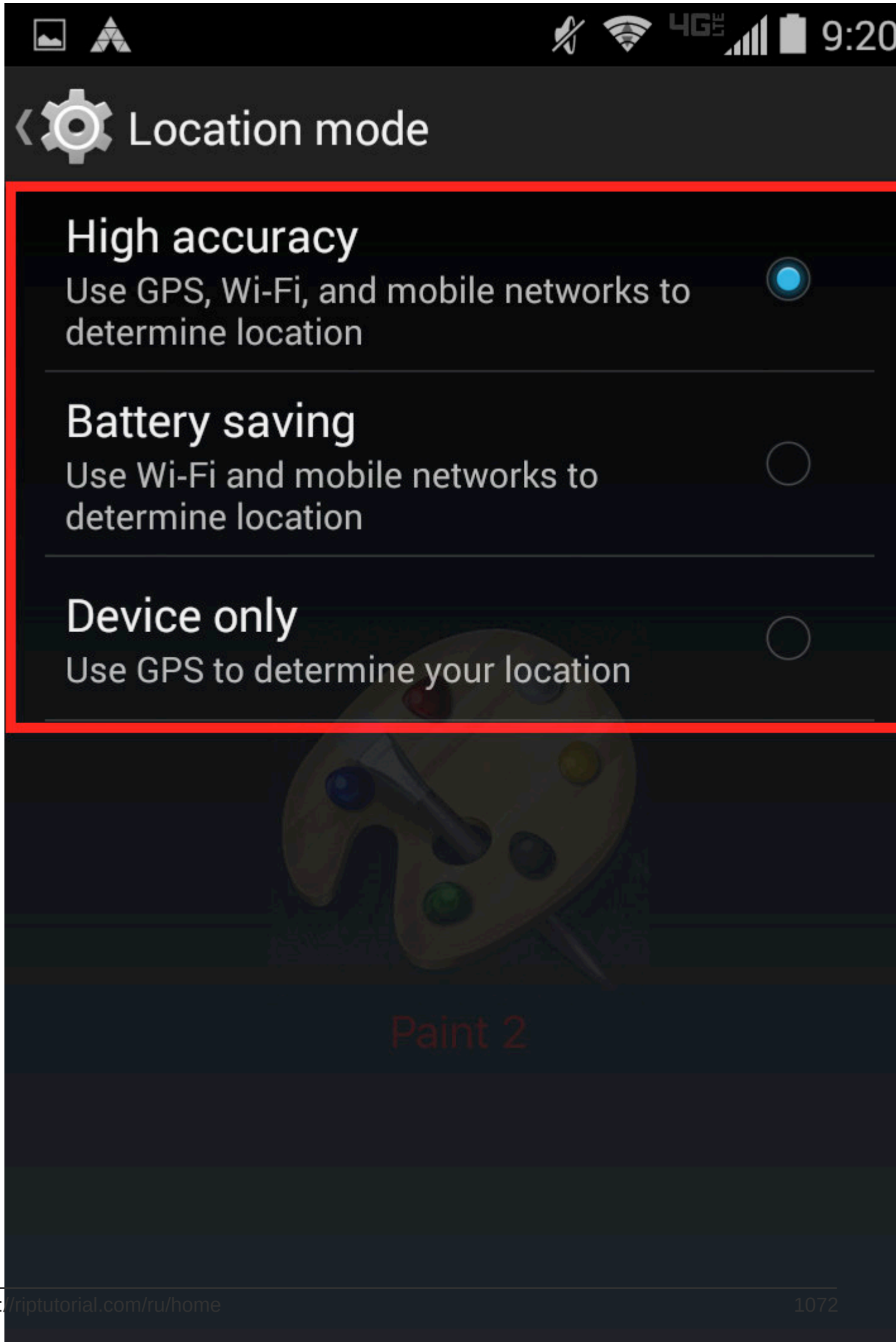
Если вы используете сетевые службы, включили ли вы «Сканирование всегда доступно»? У вас установлен режим «Лучшее» («Высокая точность») или «Аккумуляторная батарея» (только для сети)?


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

Если вы используете GPS, включили ли вы режим «Лучшее» («Высокая точность») или «Только устройство» в режиме местоположения?

включили ли вы режим «Лучшее» («Высокая точность») или «Только устройство» в режиме местоположения?

включили ли вы режим «Лучшее» («Высокая точность») или «Только устройство» в режиме местоположения?


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

Да, это здесь дважды. Вы пытались использовать `LocationListener` вместо `PendingIntent` или наоборот, чтобы убедиться, что вы действительно реализовали `LocationManager` правильно? Вы уверены, что запрос местоположения не удаляется в какой-либо части жизненного цикла Activity или Service, который вы не ожидали?

`PendingIntent` или наоборот, чтобы убедиться, что вы действительно реализовали `LocationManager` правильно? Вы уверены, что запрос местоположения не удаляется в какой-либо части жизненного цикла Activity или Service, который вы не ожидали?

---

6. Проверьте свое окружение!

   Вы тестируете GPS на первом этаже здания в центре Сан-Франциско? Проверяете ли вы сетевые местоположения в середине нигде? Вы работаете в секретном подземном бункере, лишенном всех радиосигналов, удивляясь, почему ваше устройство не получает место? Всегда проверяйте свое окружение, пытаясь устранить проблемы с местоположением!

Могло быть много других менее очевидных причин, по которым местоположение не работает, но прежде чем искать эти эзотерические исправления, просто запустите этот быстрый контрольный список.

Examples

API с плавающим местоположением

Пример использования активности w / 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
    }
}
```

Пример использования службы с ожиданием и трансляцией

ExampleActivity

Рекомендуемое чтение: 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
            }
        }
    }
}
```

Служба определения местоположения

ПРИМЕЧАНИЕ. Не забудьте зарегистрировать эту службу в манифесте!

```
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

ПРИМЕЧАНИЕ. Не забудьте зарегистрировать этот приемник в манифесте!

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

Запрос обновлений местоположения с помощью LocationManager

Как всегда, вы должны убедиться, что у вас есть необходимые разрешения.

```
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) {

    }
}
```

Запрос обновлений местоположения в отдельном потоке с помощью LocationManager

Как всегда, вы должны убедиться, что у вас есть необходимые разрешения.

```
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) {

    }
}
```

Зарегистрировать геозонность

Я создал `GeoFenceObserversationService` **одноэлементный** класс.

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);
        }
    }
```

Где я начал службу? Из класса приложения

- `startService(new Intent(getApplicationContext(),GeoFenceObserversationService.class));`

Как я зарегистрировал Geofences?

- `GeoFenceObserversationService.getInstant().addGeofences();`

Получить адрес из местоположения с помощью геокодирования

После того, как вы получили объект `Location` из `FusedAPI` , вы можете легко получить `Address` информацию от этого объекта.

```
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;
}
```

Получение обновлений местоположения в BroadcastReceiver

Сначала создайте класс BroadcastReceiver для обработки входящих обновлений местоположения:

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

Затем, когда вы подключаетесь к GoogleApiClient в обратном вызове 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);
}
```

Не забудьте удалить намерение обновления местоположения в соответствующем обратном вызове жизненного цикла:

```
@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();
}
```

Прочитайте Место нахождения онлайн: https://riptutorial.com/ru/android/topic/1837/место-нахождения

глава 169: Модульное тестирование в Android с помощью JUnit

замечания

- Vogella: Тестирование устройств с помощью JUnit
- Junit Аннотации: java2novice.com
- Класс утверждения : junit.org
- JUnit Api: tutorialspoint.com
- Анонимные сообщения

Examples

Создание локальных модульных тестов

Поместите свои тестовые классы здесь: `/src/test/<pkg_name>/`

Пример тестового класса

```
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
    }
}
```

Сломать

```
public class ExampleUnitTest {
  ...
}
```

В тестовом классе вы можете создать несколько тестовых классов и поместить их в тестовый пакет.

```
@Test
public void addition_isCorrect() {
   ...
}
```

Метод проверки, несколько тестовых методов могут быть созданы внутри тестового класса.

Обратите внимание на аннотацию `@Test` .

> Аннотирование теста сообщает JUnit, что общедоступный метод void, к которому он присоединен, может быть запущен в качестве тестового примера.

Есть несколько других полезных аннотаций, таких как `@Before` , `@After` и т. Д. Эта страница будет хорошим местом для начала.

```
assertEquals(9, c); // This test passes
assertEquals(10, c); //Test fails
```

Эти методы являются членами класса `Assert` . Некоторые другие полезные методы: `assertFalse()` , `assertNotNull()` , `assertTrue` и т. Д. Вот подробное пояснение .

Информация для аннотации для теста JUnit:

@Test: аннотация Test указывает JUnit, что общедоступный метод void, к которому он присоединен, может быть запущен в качестве тестового примера. Чтобы запустить этот метод, JUnit сначала создает новый экземпляр класса, затем вызывает аннотированный метод.

@ **Прежде:** при написании тестов обычно обнаруживается, что для нескольких тестов нужны похожие объекты, созданные до их запуска. `@Before` метода public void с `@Before` заставляет этот метод запускаться до метода Test.

@After: Если вы выделяете внешние ресурсы в методе Before, вам необходимо освободить их после **запуска** теста. `@After` метода public void с `@After` приводит к тому, что этот метод запускается после метода Test. Все методы `@After` гарантированно выполняются, даже если метод Before или Test выдает исключение

Совет Быстрое создание тестовых классов в Android Studio

- Поместите курсор на имя класса, для которого вы хотите создать тестовый класс.
- Нажмите Alt + Enter (Windows).
- Выберите «Создать тест», нажмите «Возврат».
- Выберите методы, для которых вы хотите создать методы тестирования, нажмите «OK».
- Выберите каталог, в котором вы хотите создать тестовый класс.
- Вы закончили, это то, что вы получаете, это ваш первый тест.

Совет Легко выполнять тесты в Android Studio

- Щелкните правой кнопкой мыши тестовый пакет.
- Выберите Run 'Tests in ...
- Все тесты в пакете будут выполняться сразу.

Перемещение бизнес-логики из компонентов Android

Значительная часть результатов тестирования локальных JVM-модулей объясняется тем, как вы разрабатываете приложение. Вы должны проектировать его таким образом, чтобы вы могли отделить свою бизнес-логику от своих компонентов Android. Вот пример такого способа использования шаблона Model-View-Presenter . Давайте рассмотрим это, выполнив базовый экран регистрации, который требует только имя пользователя и пароль. Наше Android-приложение отвечает за проверку того, что имя пользователя, которое пользователь предоставляет, не пуст и что пароль имеет длину не менее восьми символов и содержит хотя бы одну цифру. Если имя пользователя / пароль действительны, мы выполняем наш вызов api для вызова, иначе мы выводим сообщение об ошибке.

Пример, в котором бизнес-логика сочетается с Android-компонентом.

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

Пример, где бизнес-логика отделена от Android-компонента.

Здесь мы определяем в одном классе LoginContract, который будет содержать различные взаимодействия между нашими различными классами.

```
public interface LoginContract {
    public interface View {
        performSignUpApiCall(String username, String password);
        displayInvalidCredentialsErrorMessage();
    }
    public interface Presenter {
        void validateUserCredentials(String username, String password);
    }
}
```

Наш LoginActivity по большей части тот же, за исключением того, что мы удалили ответственность за то, чтобы знать, как проверить форму регистрации пользователя (нашу бизнес-логику). LoginActivity теперь будет использовать наш новый LoginPresenter для проверки.

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

Теперь ваша бизнес-логика будет находиться в вашем новом классе 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();
        }
    }
}
```

И теперь мы можем создать локальные тесты JVM для вашего нового класса 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);
    }
}
```

Как вы можете видеть, когда мы извлекли нашу бизнес-логику из LoginActivity и поместили ее в POJO LoginPresenter. Теперь мы можем создавать локальные тесты JVM для нашей бизнес-логики.

Следует отметить, что из-за наших изменений в архитектуре существуют различные другие последствия, так как мы близки к соблюдению каждого класса с одной ответственностью, дополнительными классами и т. Д. Это просто побочные эффекты того, как я решил пойти на это развязка через стиль MVP. MVP - это всего лишь один из способов сделать это, но есть и другие альтернативы, которые вы можете посмотреть на MVVM . Вам просто нужно выбрать лучшую систему, которая работает для вас.

Начало работы с JUnit

Настроить

Чтобы начать модульное тестирование вашего проекта Android с помощью JUnit, вам нужно добавить зависимость JUnit к вашему проекту, и вам нужно создать тестовый источник, который будет содержать исходный код для модульных тестов. Проекты, созданные с помощью Android Studio, часто уже включают зависимость JUnit и исходный набор тестов

Добавьте следующую строку в файл `build.gradle` пределах зависимостей. `Closure` :

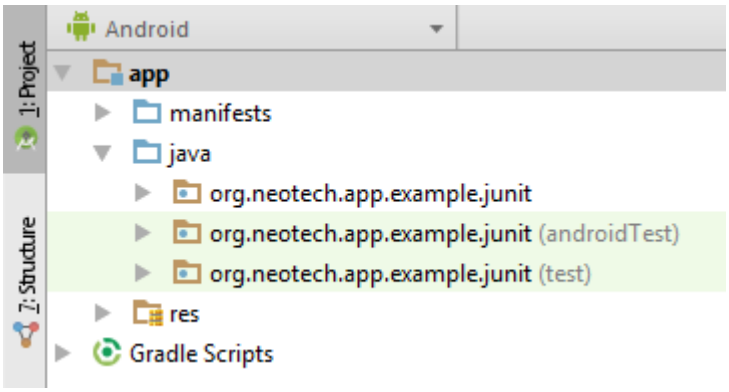
```
testCompile 'junit:junit:4.12'
```

Тест-классы JUnit расположены в специальном наборе-источнике с именем `test` . Если этот источник не существует, вам нужно создать новую папку самостоятельно. Структура папок стандартного проекта Android Studio (Gradle) выглядит следующим образом:

```
<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)
```

Если ваш проект не имеет папки `/app/src/test` вам необходимо создать его самостоятельно. В `test` папке вам также нужна папка `java` (создайте ее, если она не существует). Папка java в наборе `test` источников должна содержать ту же структуру пакета, что и ваш `main` источник.

Если правильная настройка вашей структуры проекта (в Android-представлении в Android Studio) должна выглядеть так:



Примечание. Вы не обязательно должны иметь исходный набор `androidTest` , этот набор источников часто встречается в проектах, созданных Android Studio, и здесь приводится для справки.

Написание теста

1. **Создайте новый класс в `test` источнике.**
   Щелкните правой кнопкой мыши тестовый источник, заданный в представлении проекта, выберите « `New` > « `Java class` .
   Наиболее используемым шаблоном имен является использование имени класса, которое вы собираетесь тестировать, с добавлением `Test` . Таким образом, `StringUtilities` становится `StringUtilitiesTest` .

2. **Добавьте аннотацию `@RunWith`**
   Аннотации `@RunWith` необходимы, чтобы JUnit запускал тесты, которые мы собираемся определить в нашем тестовом классе. По умолчанию JUnit runner (для JUnit 4) является `BlockJUnit4ClassRunner` но вместо того, чтобы использовать этот запуск напрямую, более удобно использовать псевдоним `JUnit4` который является сокращением для бегуна JUnit по умолчанию.

   ```
   @RunWith(JUnit4.class)
   public class StringUtilitiesTest {

   }
   ```

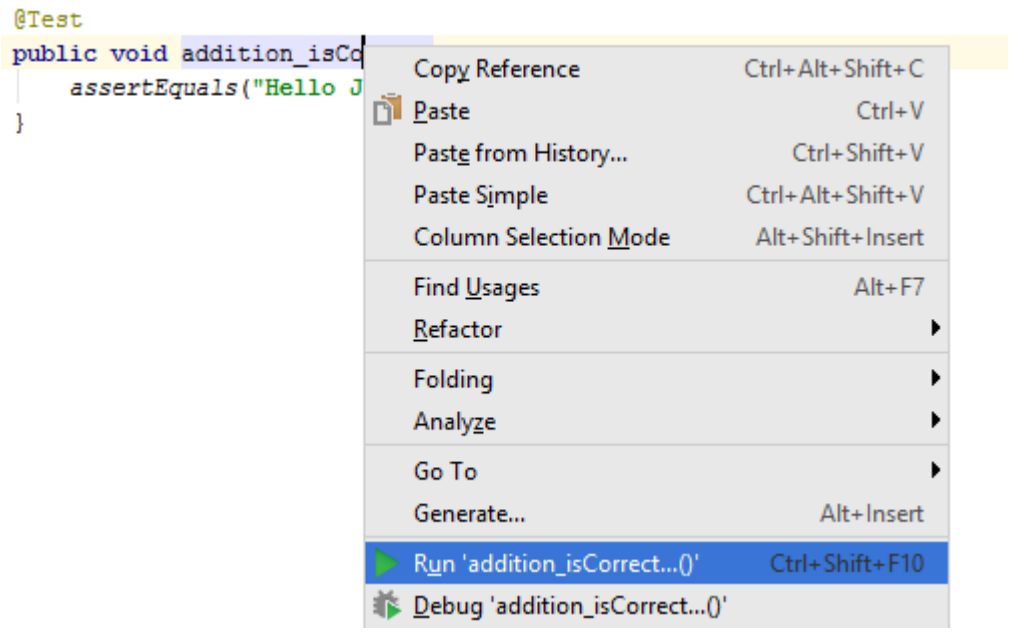
3. **Создать тест**
   Единичный тест - это просто метод, который в большинстве случаев не должен прерываться при запуске. Другими словами, это не должно вызывать исключения. Внутри тестового метода вы почти всегда найдете утверждения, которые проверяют, выполнены ли особые условия. Если утверждение не выполняется, оно вызывает исключение, из-за которого метод / тест терпит неудачу. Метод проверки всегда аннотируется аннотацией `@Test` . Без этой аннотации JUnit не будет автоматически запускать тест.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

    @Test
    public void addition_isCorrect() throws Exception {
        assertEquals("Hello JUnit", "Hello" + " " + "JUnit");
    }
}
```

Примечание: в отличие от стандартных имен меток метода именования методов Java имена меток часто содержат символы подчеркивания.

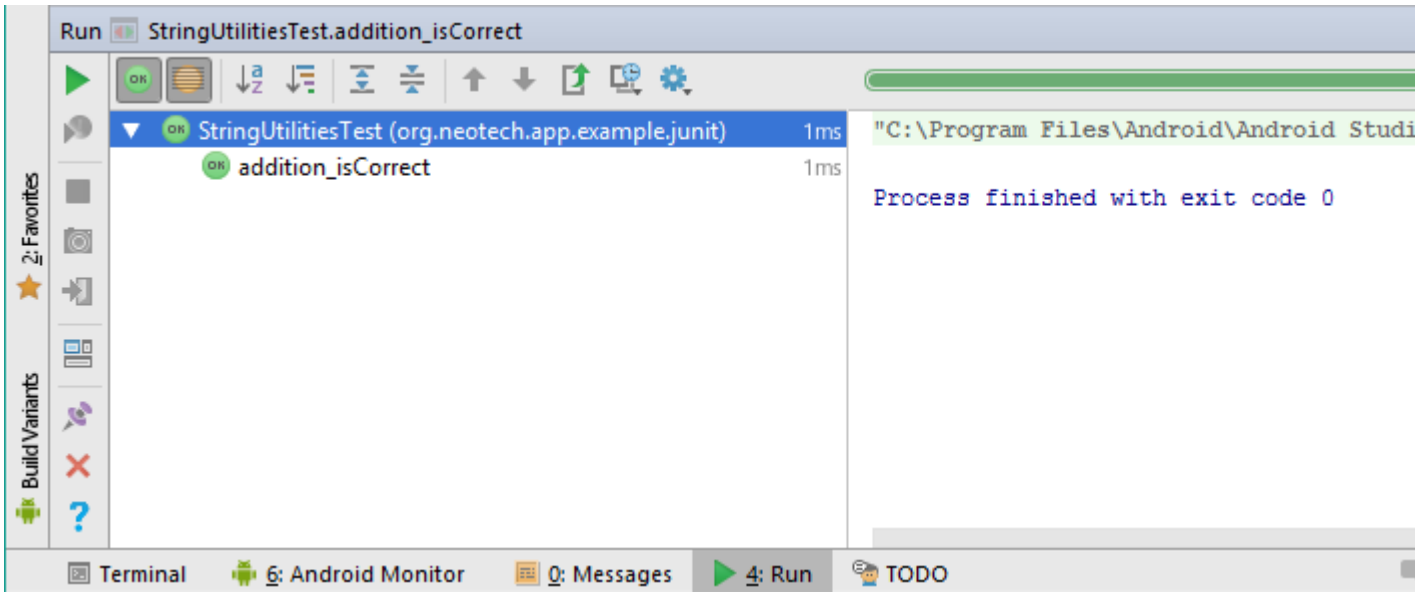
Выполнение теста

1. **метод**
   Чтобы запустить один метод тестирования, вы можете щелкнуть правой кнопкой мыши метод и нажать « `Run 'addition_isCorrect()'` или использовать сочетание клавиш



`ctrl+shift+f10` .

   Если все настроено правильно, JUnit начинает запускать метод, и вы должны увидеть следующий интерфейс в Android Studio:



2. **Учебный класс**
   Вы также можете запустить все тесты, определенные в одном классе, щелкнув правой кнопкой мыши класс в представлении проекта и нажав « `Run 'StringUtilitiesTest '` или воспользовавшись сочетанием клавиш `ctrl+shift+f10` если вы выбрали класс в представлении проекта.

3. **Пакет (все)**
   Если вы не будете запускать все тесты, определенные в проекте или в пакете, вы можете просто щелкнуть правой кнопкой мыши по пакету и нажать « `Run ...` же, как вы будете запускать все тесты, определенные в одном классе.

Исключения

JUnit также может использоваться для проверки того, выбрал ли метод конкретное исключение для данного ввода.

В этом примере мы проверим, действительно ли следующий метод генерирует исключение, если логический формат (ввод) не распознается / неизвестен:

```
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);
    }
}
```

Добавляя `expected` параметр к аннотации `@Test` , можно определить, какое исключение будет `@Test` . Тестирование модуля завершится неудачно, если это исключение не

произойдет, и преуспеть, если действительно исключено исключение:

```
@Test(expected = IllegalArgumentException.class)
public void parseBoolean_parsesInvalidFormat_throwsException(){
    StringUtilities.parseBoolean("Hello JUnit");
}
```

Это хорошо работает, однако, это ограничивает вас только одним тестовым случаем в рамках метода. Иногда вам может потребоваться проверить несколько случаев в рамках одного метода. Техника, которая часто используется для преодоления этого ограничения, заключается в использовании блоков `try-catch` и `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){
    }
}
```

Примечание. Некоторые считают, что это плохая практика для тестирования более чем одного случая внутри модульного теста.

Статический импорт

JUnit определяет довольно некоторые методы `assertEquals` по крайней мере один для каждого примитивного типа, и один для объектов доступен. Эти методы по умолчанию не имеют прямого доступа для вызова и должны быть вызваны следующим образом: `Assert.assertEquals` . Но поскольку эти методы используются так часто, люди почти всегда используют статический импорт, поэтому метод может быть непосредственно использован так, как если бы он был частью самого класса.

Чтобы добавить статический импорт для метода `assertEquals` используйте следующий оператор импорта:

```
import static org.junit.Assert.assertEquals;
```

Вы также можете статически импортировать все методы assert, включая `assertArrayEquals` , `assertNotNull` и `assertFalse` и т. Д., Используя следующий статический импорт:

```
import static org.junit.Assert.*;
```

Без статического импорта:

```
@Test
public void addition_isCorrect(){
    Assert.assertEquals(4 , 2 + 2);
}
```

Со статическим импортом:

```
@Test
public void addition_isCorrect(){
    assertEquals(4 , 2 + 2);
}
```

Прочитайте Модульное тестирование в Android с помощью JUnit онлайн:
https://riptutorial.com/ru/android/topic/3205/модульное-тестирование-в-android-с-помощью-junit

глава 170: Мультидекс и предел метода Dex

Вступление

DEX означает исполняемые файлы байт-кода Android (APK) в виде файлов Dalvik Executable (DEX), которые содержат скомпилированный код, используемый для запуска вашего приложения.

Спецификация исполняемого файла Dalvik ограничивает общее количество методов, на которые можно ссылаться в одном файле DEX, до 65 536 (64 КБ), включая методы каркаса Android, библиотечные методы и методы в вашем собственном коде.

Чтобы преодолеть этот предел, необходимо настроить процесс сборки приложений для создания более одного файла DEX, известного как Multidex.

замечания

Что такое dex?

Dex - это имя формата файла и кодировки, с которой компилируется код Java Java. Ранние версии Android загружали и исполняли двоичные файлы `dex` непосредственно на виртуальной машине Dalvik. В более поздних версиях Android используется Android Runtime (ART), которая обрабатывает файлы `dex` как промежуточное представление и выполняет дальнейшие компиляции на нем до запуска приложения.

Dex - очень старый формат файла с точки зрения продолжительности жизни смартфонов и предназначен для устройств, основная память которых измерена в десятках мегабайт. Дизайнерские ограничения тех дней остались с нами по сей день.

Эта проблема:

Формат файла `dex` кодирует ограничение на количество методов, на которые можно ссылаться в одном бинарном файле. Поскольку часть формата файла, в которой хранится количество ссылок, составляет два байта, максимальное число ссылок на методы - `0xFFFF` или 65535. Если приложение содержит больше, чем это число ссылок на методы, оно не скомпилируется.

Что с этим делать:

Google предоставил возможность обойти эту проблему, называемую Multidex. Он имеет компоненты времени компиляции и времени выполнения. Как следует из его названия, во время компиляции он будет делить код между одним или несколькими файлами `dex` . Во время выполнения он научит стандартного `ClassLoader` как искать классы из этих файлов.

Этот подход хорошо работает на новых устройствах, но имеет некоторые существенные недостатки. Это может значительно увеличить время запуска приложений, а на старых устройствах может возникнуть отказ `Application Not Responding` .

По возможности следует избегать использования Multidex, хотя и эффективно.

Как избежать ограничения:

Прежде чем настраивать приложение, чтобы разрешить использование ссылок на 64K или более, вы должны предпринять шаги, чтобы уменьшить общее количество ссылок, вызванных вашим кодом приложения, включая методы, определенные вашим кодом приложения или включенными библиотеками. Следующие стратегии помогут вам избежать попадания лимита ссылки dex:

- **Просмотрите прямые и транзитивные зависимости вашего приложения.** Убедитесь, что любая большая зависимость библиотеки, которую вы включаете в приложение, используется таким образом, чтобы перераспределить количество добавляемого кода в приложение. Общим анти-шаблоном является включение очень большой библиотеки, потому что несколько полезных методов были полезны. Сокращение зависимостей от кода приложения часто помогает избежать ограничения ссылки на dex.
- **Удалите неиспользуемый код с помощью ProGuard.** Настройте параметры ProGuard для вашего приложения, чтобы запустить ProGuard, и убедитесь, что у вас есть сокращения для релизов. Включение усадки гарантирует, что вы не отправляете неиспользованный код с вашими APK.

Первый момент требует усердия и дисциплины со стороны разработчика. При включении сторонних библиотек следует учитывать размер библиотеки. Например, две популярные библиотеки JSON - это Джексон и Гсон. Функционально они довольно похожи, но Gson имеет тенденцию видеть большее использование в Android. Одна из причин заключается в том, что Джексон весит около 9000 методов, тогда как Gson составляет 1900 человек.

Существует несколько инструментов, помогающих разработчикам отслеживать размер их приложения:

- dexcount-gradle-plugin сообщает количество ссылок на методы в вашем APK или AAR для каждой сборки
- dex-method-counts - это средство командной строки, которое подсчитывает количество ссылок на методы в APK

- www.methodscount.com - это веб-служба, которая будет рассчитывать ссылки на методы в любом APK, который вы загружаете.

Examples

Мультидекс с помощью MultiDexApplication напрямую

Используйте этот параметр, если вам не нужен подкласс `Application` .

Это самый простой вариант, но таким образом вы не можете предоставить свой собственный подкласс `Application` . Если необходим подкласс `Application` , вам придется перейти на один из других вариантов, чтобы сделать это.

Для этого параметра просто укажите полное имя класса `android.support.multidex.MultiDexApplication` для свойства `android:name` тега `application` в 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, расширяя приложение

Используйте этот параметр, если для вашего проекта требуется подкласс `Application` .

Задайте этот подкласс `Application` используя свойство `android:name` в файле манифеста внутри тега `application` .

В подклассе `Application` добавьте переопределение метода `attachBaseContext()` , и в этом методе вызовите `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);
    }
}
```

Убедитесь, что подкласс `Application` указан в теге `application` вашего AndroidManifest.xml:

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

Включение Multidex

Чтобы включить конфигурацию multidex, вам необходимо:

- изменить конфигурацию сборки Gradle
- использовать `MultiDexApplication` или включить MultiDex в своем классе `Application`

Конфигурация гребенки

В `app/build.gradle` добавьте следующие части:

```
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'
}
```

Включить MultiDex в приложении

Затем выполните один из трех вариантов:

- Multidex, расширяя приложение
- Multidex путем расширения MultiDexApplication

- Мультидекс с помощью MultiDexApplication напрямую

Когда эти настройки конфигурации добавляются в приложение, инструменты сборки Android строят первичный dex (classes.dex) и поддерживают (classes2.dex, classes3.dex) по мере необходимости.
Затем система сборки упаковывает их в файл APK для распространения.

Ссылки метода подсчета на каждом сборке (плагин Gradle Dexcount)

Плагин dexcount подсчитывает методы и количество ресурсов класса после успешной сборки.

Добавьте плагин в `app/build.gradle` :

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

buildscript {
    repositories {
        mavenCentral() // or jcenter()
    }

    dependencies {
        classpath 'com.getkeepsafe.dexcount:dexcount-gradle-plugin:0.5.5'
    }
}
```

Примените плагин в файле `app/build.gradle` :

```
apply plugin: 'com.getkeepsafe.dexcount'
```

Ищите выходные данные, сгенерированные плагином:

../app/build/outputs/dexcount

Особенно полезной является диаграмма .html в:

../app/build/outputs/dexcount/debugChart/index.html

Multidex путем расширения MultiDexApplication

Это очень похоже на использование подкласса `Application` и переопределение `attachBaseContext()` .

Однако, используя этот метод, вам не нужно переопределять `attachBaseContext()` поскольку это уже сделано в суперклассе `MultiDexApplication` .

Расширьте `MultiDexApplication` вместо `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()

     //.........
}
```

Добавьте этот класс в свой AndroidManifest.xml точно так же, как если бы вы расширяли приложение:

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

Прочитайте Мультидекс и предел метода Dex онлайн:
https://riptutorial.com/ru/android/topic/1887/мультидекс-и-предел-метода-dex

глава 171: Написание UI-тестов - Android

Вступление

Фокус этого документа заключается в представлении целей и способов написания пользовательских интерфейсов Android и интеграционных тестов. Эспрессо и UIAutomator предоставляются Google, поэтому основное внимание должно быть уделено этим инструментам и их соответствующим упаковкам, например, Appium, Spoon и т. Д.

Синтаксис

- **Ресурс холостого хода**
- String getName () - возвращает имя ресурса холостого хода (используется для ведения журнала и идемпотентности регистрации).
- boolean isIdleNow () - возвращает true, если ресурс в настоящее время не используется.
- void registerIdleTransitionCallback (IdlingResource.ResourceCallback callback) - Регистрирует данный IdlingResource.ResourceCallback с ресурсом

замечания

Правила JUnit:

Как вы можете видеть в примере MockWebServer и ActivityTestRule, все они относятся к категории правил JUnit, которые вы можете создать самостоятельно, которые затем должны быть выполнены для каждого теста, определяющего его поведение @see: https://github.com/junit-team/junit4/ вики / правила

Appium

параметры

Поскольку параметры имеют некоторые проблемы, помещая их здесь, пока не будет устранена ошибка в документации:

параметр	подробности
Класс activityClass	какую деятельность начать
initialTouchMode	если при запуске деятельность должна быть помещена в режим

параметр	подробности
	касания: https://android-developers.blogspot.de/2008/12/touch-mode.html
launchActivity	true, если действие должно быть запущено один раз для каждого метода тестирования. Он будет запущен до первого метода Before и завершен после последнего метода After.

Examples

Пример MockWebServer

В случае, если ваши действия, фрагменты и пользовательский интерфейс требуют некоторой фоновой обработки, полезно использовать MockWebServer, который работает локально на устройстве Android, что обеспечивает закрытую и проверяемую среду для вашего пользовательского интерфейса.

https://github.com/square/okhttp/tree/master/mockwebserver

Первый шаг включает в себя зависимость градиента:

```
testCompile 'com.squareup.okhttp3:mockwebserver:(insert latest version)'
```

Теперь шаги для запуска и использования макетного сервера:

- создать макет серверного объекта
- запустите его по определенному адресу и порту (обычно localhost: portnumber)
- выдавать ответы для конкретных запросов
- начать тест

Это хорошо объяснено на странице github mockwebserver, но в нашем случае мы хотим, чтобы что-то было более приятным и многоразовым для всех тестов, и правила JUnit пойдут в игру:

```
/**
 *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”);
        }
    }
}
```

Теперь давайте предположим, что мы имеем ту же самую активность, что и в предыдущем примере, только в этом случае, когда мы нажимаем кнопку, приложение будет извлекать что-то из сети, например: https://someapi.com/name

Это вернет некоторую текстовую строку, которая будет конкатенирована в тексте закутки, например, NAME + текст, который вы ввели.

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

Я бы предложил обернуть диспетчера в какой-то Builder, чтобы вы могли легко связать и добавить новые ответы для своих экранов. например

```
 return newDispatcherBuilder()
            .withSerializedJSONBody("/authenticate", Mocks.getAuthenticationResponse())
```

```
            .withSerializedJSONBody("/getUserInfo", Mocks.getUserInfo())
            .withSerializedJSONBody("/checkNotBot", Mocks.checkNotBot());
```

IdlingResource

Сила ресурсов холостого хода заключается в том, что вам не нужно ждать обработки некоторых приложений (сетей, вычислений, анимаций и т. Д.), Чтобы закончить `sleep()` , что приводит к отслаиванию и / или продлению тестов. Официальную документацию можно найти здесь .

Реализация

Для реализации интерфейса `IdlingResource` необходимо выполнить `IdlingResource` :

- **`getName()`** - возвращает имя вашего ресурса холостого хода.
- **`isIdleNow()`** - проверяет, является ли ваш объект xyz, операция и т. д. в настоящий момент бездействующим.
- **`registerIdleTransitionCallback`** (`IdlingResource.ResourceCallback` callback) - обеспечивает обратный вызов, который вы должны вызывать, когда ваш объект переходит в режим ожидания.

Теперь вы должны создать свою собственную логику и определить, когда ваше приложение простаивает, а когда нет, поскольку это зависит от приложения. Ниже вы найдете простой пример, просто чтобы показать, как он работает. В Интернете есть другие примеры, но конкретная реализация приложения приводит к конкретным реализациям ресурсов бездействия.

ЗАМЕТКИ

- Были примеры Google, где они помещали `IdlingResources` в код приложения. **Не делайте этого.** Предположительно, они разместили его там, чтобы показать, как они работают.
- Сохранение вашего кода в чистоте и соблюдение единого принципа ответственности зависит от вас!

пример

Скажем, у вас есть активность, которая делает странные вещи и занимает много времени для загрузки фрагмента, и, таким образом, ваши тесты Espresso терпят неудачу, не имея возможности найти ресурсы из вашего фрагмента (вы должны изменить, как создается ваша деятельность и когда чтобы ускорить его). Но в любом случае, чтобы это было просто, следующий пример показывает, как это должно выглядеть.

Наш пример ресурса холостого хода получит два объекта:

- **Тег** фрагмента, который вам нужно найти и ждать, чтобы привязаться к этой деятельности.
- Объект **FragmentManager,** который используется для поиска фрагмента.

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

Теперь, когда вы создали свой `IdlingResource` , вам нужно использовать его где-то в порядке?

использование

Давайте пропустим всю установку тестового класса и посмотрим, как будет выглядеть тестовый пример:

```
@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);
}
```

Комбинация с правилом JUnit

Это не сложно; вы также можете применить ресурс холостого хода в форме тестового правила JUnit. Например, скажем, что у вас есть SDK, в котором есть Volley, и вы хотите, чтобы Espresso подождал его. Вместо того, чтобы проходить через каждый тестовый пример или применять его в настройке, вы можете создать правило JUnit и просто написать:

```
@Rule
public final SDKIdlingRule mSdkIdlingRule = new
SDKIdlingRule(SDKInstanceHolder.getInstance());
```

Теперь, поскольку это пример, не считайте это само собой разумеющимся; весь код здесь мнимый и используется только в демонстрационных целях:

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

Прочитайте Написание UI-тестов - Android онлайн: https://riptutorial.com/ru/android/topic/3530/написание-ui-тестов---android

глава 172: Настройка Jenkins CI для Android-проектов

Examples

Пошаговый подход к настройке Jenkins для Android

Это пошаговое руководство по настройке процесса автоматической сборки с использованием Jenkins CI для ваших проектов Android. Следующие шаги предполагают, что у вас есть новое оборудование с любым вкусом Linux. Также учитывается, что у вас может быть удаленная машина.

ЧАСТЬ I: первоначальная настройка на вашем компьютере

1. Войдите через *ssh* на свою машину Ubuntu:

   ```
   ssh username@xxx.xxx.xxx
   ```

2. Загрузите версию Android SDK на свой компьютер:

   ```
   wget https://dl.google.com/android/android-sdk_r24.4.1-linux.tgz
   ```

3. Распакуйте загруженный файл *tar* :

   ```
   sudo apt-get install tar
   tar -xvf android-sdk_r24.4.1-linux.tgz
   ```

4. Теперь вам нужно установить Java 8 на свою машину Ubuntu, что является требованием для Android, основанного на Nougat. Дженкинсу потребуется установить JDK и JRE 7, используя следующие шаги:

   ```
   sudo apt-get install python-software-properties
   sudo add-apt-repository ppa: webupd8team / java
   Обновление sudo apt-get
   apt-get install openjdk-8-jdk
   ```

5. Теперь установите Jenkins на машину 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'
   ```

Обновление sudo apt-get
sudo apt-get install jenkins

6. Загрузите последнюю версию Gradle для настройки Android:

   wget https://services.gradle.org/distributions/gradle-2.14.1-all.zip
   unzip gradle-2.14.1-all.zip

7. Настройте Android на своей машине Ubuntu. Сначала перейдите в папку *инструментов в* папке Android SDK, загруженной на шаге 2:

   cd android-sdk-linux / tools **// списки доступных SDK**
   android update sdk --no-ui **// Обновляет версию SDK**
   Список android sdk -a | grep «SDK Build-tools» **// перечисляет доступные инструменты сборки**
   android update sdk -a -u -t 4 **// обновляет версию инструментов для сборки до версии, указанной в списке 4. CMD.**
   обновить java

8. Установите *Git* или любой другой VCS на вашем компьютере:

   sudo apt-get install git

9. Теперь войдите в Jenkins, используя ваш интернет-браузер. Введите `ipAddress:8080` в адресную строку.

10. Чтобы получить пароль для первого входа в систему, проверьте соответствующий файл следующим образом (для доступа к этому файлу вам понадобятся права su):

    cat / var / lib / jenkins / secrets / initialAdminPassword

ЧАСТЬ II: Настройте Jenkins для создания приложений для Android

1. После входа в систему перейдите по следующему пути:

   Дженкинс> Управление Дженкинсом> Глобальная настройка инструмента

2. В этом месте добавьте `JAVA_HOME` со следующими данными:

   Имя = JAVA_HOME
   JAVA_HOME = / usr / lib / jvm / java-8-openjdk-amd64

3. Также добавьте следующие значения в *Git* и сохраните переменные среды:

Имя = По умолчанию
/ USR / бен / мерзавец

4. Теперь переходим к следующему пути:

   Дженкинс> Управление Дженкинсом> Конфигурация

5. В этом месте добавьте `ANDROID_HOME` в «глобальные свойства»:

   Имя = ANDROID_HOME
   Value = / home / username / android-sdk-linux

Часть III: Создайте Jenkins Job для своего Android-проекта

1. Нажмите « *Новый элемент*» на главном экране Jenkins.
2. Добавьте *название* и *описание проекта* .
3. На вкладке « *Общие* » выберите « *Дополнительно*» . Затем выберите « *Использовать настраиваемое рабочее пространство*» :

   Directory / home / user / Code / ProjectFolder

4. В управлении исходным кодом выберите *Git* . Я использую *Bitbucket* для этого примера:

   URL-адрес репозитория = https: // имя пользователя: пароль@ bitbucket.org/project/projectname.git

5. Выберите дополнительное поведение для вашего репозитория:

   Очистить перед покупкой
   Оформить заказ в подкаталог. Локальный подкаталог для repo / home / user / Code / ProjectFolder

6. Выберите ветку, которую вы хотите построить:

   */мастер

7. На вкладке « *Сборка* » выберите « *Выполнить оболочку*» в *шаге «Добавить сборку»* .
8. В *оболочке Execute* добавьте следующую команду:

   cd / home / user / Code / ProjectFolder && gradle clean assemble --no-daemon

9. Если вы хотите запустить Lint в проекте, добавьте еще один шаг сборки в *оболочку Execute* :

```
/home/user/gradle/gradle-2.14.1/bin/gradle lint
```

Теперь ваша система, наконец, настроена на создание проектов Android с использованием Jenkins. Эта настройка делает вашу жизнь намного проще для выпуска сборок для команд QA и UAT.

PS: Поскольку Jenkins - другой пользователь на вашей машине Ubuntu, вы должны предоставить ему права на создание папок в своем рабочем пространстве, выполнив следующую команду:

```
chown -R jenkins .git
```

глава 173: Начало работы с OpenGL ES 2.0+

Вступление

Эта тема посвящена настройке и использованию **OpenGL ES 2.0+** на Android. OpenGL ES является стандартом для 2D и 3D ускоренной графики на встроенных системах - в том числе консолей, смартфонов, приборов и транспортных средств.

Examples

Настройка GLSurfaceView и OpenGL ES 2.0+

Чтобы использовать OpenGL ES в своем приложении, вы должны добавить это в манифест:

```
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
```

Создайте расширенный GLSurfaceView:

```
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);
        }
    }
}
```

Добавьте `MyGLSurfaceView` в свой макет:

```
<com.example.app.MyGLSurfaceView
    android:id="@+id/gles_renderer"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Чтобы использовать более новую версию OpenGL ES, просто измените номер версии в своем манифесте, в статическом импорте и изменении `setEGLContextClientVersion` .

Компиляция и связывание шейдеров GLSL-ES из файла активов

Папка « Активы » является наиболее распространенным местом для хранения ваших шейдерных файлов GLSL-ES. Чтобы использовать их в вашем приложении OpenGL ES, вам необходимо сначала загрузить их в строку. Эти функции создают строку из файла активов:

```
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;
    }
}
```

Теперь вам нужно создать функцию, которая компилирует шейдер, хранящийся в укусе:

```
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;
}
```

Теперь вы можете загружать, компилировать и связывать свои шейдеры:

```
// 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);
}
```

Если ошибок нет, ваша программа шейдеров готова к использованию:

```
glUseProgram(shaderProgram);
```

Прочитайте Начало работы с OpenGL ES 2.0+ онлайн:
https://riptutorial.com/ru/android/topic/8662/начало-работы-с-opengl-es-2-0plus

глава 174: Неявные намерения

Синтаксис

- Намерение ()
- Намерение (намерение о)
- Intent (String action)
- Намерение (String action, Uri uri)
- Intent (Context packageContext, Class <?> Cls)
- Intent (String action, Uri uri, Context packageContext, Class <?> Cls)

параметры

параметры	подробности
о	умысел
действие	`String`: действие Intent, такое как `ACTION_VIEW`.
URI	`Uri`: URI данных о намерениях.
packageContext	`Context`: контекст пакета приложений, реализующего этот класс.
ЦБС	`Class`: класс компонента, который должен использоваться для намерения.

замечания

- Подробнее о намерении
- Подробнее о типах намерений

Examples

Неявные и явные намерения

Явное намерение используется для запуска активности или службы в одном и том же пакете приложений. В этом случае явно указывается имя предполагаемого класса:

```
Intent intent = new Intent(this, MyComponent.class);
startActivity(intent);
```

Тем не менее, неявное намерение отправляется по всей системе для любого приложения, установленного на устройстве пользователя, которое может справиться с этим намерением. Это используется для обмена информацией между различными приложениями.

```
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
}
```

Более подробную информацию о различиях можно найти в Android разработчика документации здесь: Намерение Разрешение

Неявные намерения

Неявные намерения не называют конкретный компонент, а вместо этого объявляют общее действие для выполнения, которое позволяет компоненту из другого приложения обрабатывать его.

Например, если вы хотите показать пользователю местоположение на карте, вы можете использовать неявное намерение запросить, чтобы другое способное приложение отображало указанное местоположение на карте.

Пример:

```
// 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);
}
```

Прочитайте Неявные намерения онлайн: https://riptutorial.com/ru/android/topic/5336/неявные-намерения

глава 175: Нижние листы

Вступление

Нижний лист - это лист, который скользит вверх от нижнего края экрана.

замечания

Нижние листы скользят вверх от нижней части экрана, чтобы показать больше контента. Они были добавлены в библиотеку поддержки Android в версии v.2.2.0.

Examples

BottomSheetBehavior как карты Google

2.1.x

Этот пример зависит от библиотеки поддержки 23.4.0. +.

BottomSheetBehavior характеризуется:

1. Две панели инструментов с анимацией, которые реагируют на движение нижнего листа.
2. FAB, который скрывается, когда он находится рядом с «модальной панелью инструментов» (тот, который появляется, когда вы сдвигаетесь вверх).
3. Изображение заднего плана за нижним листом с каким-то эффектом параллакса.
4. Заголовок (TextView) на панели инструментов, который появляется, когда нижний лист достигает его.
5. Панель состояния уведомлений может превращать свой фон в прозрачный или полный цвет.
6. Пользовательское поведение нижнего листа с состоянием «якорь».

Теперь давайте проверим их один за другим:

Панели инструментов
Когда вы открываете это представление в Картах Google, вы можете увидеть панель инструментов, где вы можете искать, это единственный, что я не делаю точно так же, как Карты Google, потому что я хотел сделать это более общим. Во всяком случае , что `ToolBar` находится внутри `AppBarLayout` и он получил скрытый , когда вы начинаете перетаскивание BottomSheet и появляется снова , когда BottomSheet достигают `COLLAPSED` состояние.

Для этого вам необходимо:

- создать `Behavior` и расширить его из `AppBarLayout.ScrollingViewBehavior`
- переопределить `layoutDependsOn` и `onDependentViewChanged` . Сделав это, вы будете слушать движения нижнего листа.
- создайте некоторые методы для скрытия и отображения AppBarLayout / ToolBar с анимацией.

Так я сделал это для первой панели инструментов или 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();
}
```

Вот полный файл, если вам это нужно

Вторая панель инструментов или «Модальная» панель инструментов:
Вы должны переопределить те же методы, но в этом вам нужно позаботиться о более поведении:

- показать / скрыть панель инструментов с анимацией
- изменить цвет строки состояния / фона
- показать / скрыть заголовок BottomSheet в ToolBar
- закройте нижний лист или отправьте его в состояние свертывания

Код для этого немного обширен, поэтому я позволю ссылку

FAB

Это также пользовательское поведение, но распространяется от `FloatingActionButton.Behavior` . В `onDependentViewChanged` вам нужно посмотреть, когда он достигнет «offSet» или указать, где вы хотите скрыть это. В моем случае я хочу скрыть его, когда он находится рядом со второй панелью инструментов, поэтому я копаю в родительском блоке FAB (CoordinatorLayout), который ищет AppBarLayout, который содержит ToolBar, тогда я использую позицию ToolBar, такую как `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;
}
```

Полная пользовательская ссылка на поведение FAB

Изображение за нижним листом с эффектом параллакса :
Как и другие, это обычное поведение, единственная «сложная» вещь в этом заключается в небольшом алгоритме, который удерживает изображение привязанным к BottomSheet и избегает краха изображения, как эффект параллакса по умолчанию:

```
@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;
}
```

Полный файл для фонового изображения с эффектом параллакса

Теперь для завершения: **пользовательское поведение нижнего листа**
Для достижения 3-х шагов сначала вам нужно понять, что по умолчанию у BottomSheetBehavior есть 5 состояний: `STATE_DRAGGING`, `STATE_SETTLING`, `STATE_EXPANDED`, `STATE_COLLAPSED`, `STATE_HIDDEN` и для поведения Google Maps вам нужно добавить среднее состояние между свернутыми и расширенными: `STATE_ANCHOR_POINT` .

Я попытался расширить дефолтное значение bottomSheetBehavior без каких-либо успехов, поэтому просто скопировал весь код и изменил то, что мне нужно.
Чтобы достичь того, о чем я говорю, выполните следующие шаги:

1. Создайте класс Java и расширьте его из `CoordinatorLayout.Behavior<V>`
2. Скопируйте код вставки из файла `BottomSheetBehavior` умолчанию в новый.
3. Измените метод `clampViewPositionVertical` следующим кодом:

```
@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. Добавить новое состояние

   public static final int STATE_ANCHOR_POINT = X;

5. Измените следующие методы: `onLayoutChild` , `onStopNestedScroll` , `BottomSheetBehavior<V> from(V view)` и `setState` (необязательно)

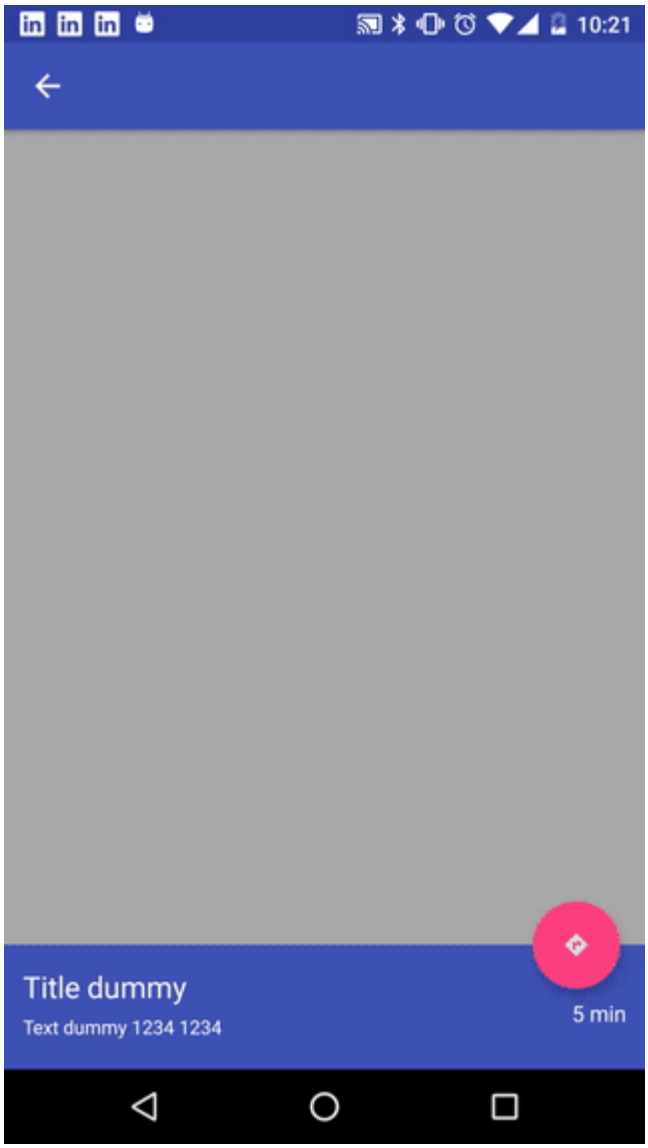
```
public boolean onLayoutChild(CoordinatorLayout parent, V child, int layoutDirection) {
    // First let the parent lay it out
    if (mState != STATE_DRAGGING && mState != STATE_SETTLING) {
        if (ViewCompat.getFitsSystemWindows(parent) &&
                !ViewCompat.getFitsSystemWindows(child)) {
            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;
}
```

Ссылка на весь проект, где вы можете увидеть все пользовательские поведения

И вот как это выглядит:

[

]

Быстрая установка

Убедитесь, что в файл build.gradle вашего приложения добавлена следующая зависимость:

```
compile 'com.android.support:design:25.3.1'
```

Затем вы можете использовать нижний лист, используя следующие параметры:

- `BottomSheetBehavior` для использования с `CoordinatorLayout`
- `BottomSheetDialog` который является диалогом с поведением нижнего листа
- `BottomSheetDialogFragment` который является расширением `DialogFragment` , который создает `BottomSheetDialog` вместо стандартного диалога.

Стойкие нижние листы

Вы можете достичь `BottomSheetBehavior` листа, содержащего `BottomSheetBehavior` для ребенка. Вид `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>
```

Затем в вашем коде вы можете создать ссылку, используя:

```
 // The View with the BottomSheetBehavior
 View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
 BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

Вы можете установить состояние своего метода BottomSheetBehavior с помощью метода setState () :

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

Вы можете использовать одно из следующих состояний:

- `STATE_COLLAPSED` : это `STATE_COLLAPSED` состояние по умолчанию и показывает только часть макета по дну. Высота можно контролировать с помощью атрибута `app:behavior_peekHeight` (по умолчанию 0)
- `STATE_EXPANDED` : полностью расширенное состояние нижнего листа, где видна вся нижняя `STATE_EXPANDED` (если ее высота меньше, чем содержащая `CoordinatorLayout`) или заполняется весь `CoordinatorLayout`
- `STATE_HIDDEN` : отключено по умолчанию (и включено с атрибутом `app:behavior_hideable`), что позволяет пользователям прокручивать нижний лист, чтобы полностью скрыть нижний лист

Если вы хотите получать обратные вызовы изменений состояния, вы можете добавить `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
   }
 });
```

Модальные нижние листы с BottomSheetDialogFragment

Вы можете реализовать модальные нижние листы, используя `BottomSheetDialogFragment` .

`BottomSheetDialogFragment` - это модальный нижний лист.
Это версия `DialogFragment` которая показывает нижний лист с использованием `BottomSheetDialog` вместо плавающего диалога.

Просто определите фрагмент:

```
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);
    }
}
```

Затем используйте этот код, чтобы показать фрагмент:

```
MyBottomSheetDialogFragment mySheetDialog = new MyBottomSheetDialogFragment();
FragmentManager fm = getSupportFragmentManager();
mySheetDialog.show(fm, "modalSheetDialog");
```

Этот фрагмент создаст `BottomSheetDialog` .

Модальные нижние листы с BottomSheetDialog

`BottomSheetDialog` - это диалог в виде нижнего листа

Просто используйте:

```
//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();
```

В этом случае вам не нужно вводить поведение BottomSheet.

Open BottomSheet DialogFragment в расширенном режиме по умолчанию.

BottomSheet DialogFragment по умолчанию открывается в `STATE_COLLAPSED` . Который может быть принудительно открыт для `STATE_EXPANDED` и `STATE_EXPANDED` полный экран устройства с помощью следующего шаблона кода.

@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;
}
```

Хотя диалогическая анимация немного заметна, но задача открытия диалога DialogFragment в полноэкранном режиме очень хорошо.

Прочитайте Нижние листы онлайн: https://riptutorial.com/ru/android/topic/5702/нижние-листы

глава 176: Низкая энергия Bluetooth

Вступление

Эта документация предназначена для усовершенствования исходной документации, и она сосредоточится на новейшем интерфейсе Bluetooth LE, представленном в Android 5.0 (API 21). Будут рассмотрены как центральные, так и периферийные роли, а также как начать сканирование и рекламные операции.

Examples

Поиск устройств BLE

Для использования API-интерфейсов Bluetooth необходимы следующие разрешения:

```
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
```

Если вы настроили таргетинг на устройства с Android 6.0 (**API Level 23**) или выше и хотите выполнять операции сканирования / рекламы, вам потребуется разрешение на размещение:

```
android.permission.ACCESS_FINE_LOCATION
```

или же

```
android.permission.ACCESS_COARSE_LOCATION
```

Примечание. Устройства с Android 6.0 (API Level 23) или выше также должны иметь службы определения местоположения.

Для запуска сканирования / рекламных операций требуется объект BluetoothAdapter:

```
BluetoothManager bluetoothManager = (BluetoothManager)
context.getSystemService(Context.BLUETOOTH_SERVICE);
bluetoothAdapter = bluetoothManager.getAdapter();
```

Метод `startScan (ScanCallback callback)` класса BluetoothLeScanner является самым основным способом запуска операции сканирования. Для получения результатов требуется объект `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());
       }
    });
```

Подключение к серверу GATT

После того, как вы обнаружили желаемый объект BluetoothDevice, вы можете подключиться к нему с помощью своего `connectGatt()` который принимает в качестве параметров объект Context, логическое значение, указывающее, следует ли автоматически подключаться к устройству BLE и ссылку BluetoothGattCallback, где события подключения и операции с клиентом результаты будут доставляться:

```
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        device.connectGatt(context, false, bluetoothGattCallback,
BluetoothDevice.TRANSPORT_AUTO);
    } else {
        device.connectGatt(context, false, bluetoothGattCallback);
    }
```

Переопределить `onConnectionStateChange` в BluetoothGattCallback для получения соединения с событиями разъединения:

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

Запись и чтение из характеристик

Как только вы подключитесь к серверу Gatt, вы будете взаимодействовать с ним, написав и прочитав характеристики сервера. Для этого сначала вам нужно выяснить, какие услуги доступны на этом сервере и какие характеристики доступны для каждой службы:

```
 @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
                    }
                }
            }
           }
```

Основная операция записи выполняется следующим образом:

```
characteristic.setValue(newValue);
characteristic.setWriteType(BluetoothGattCharacteristic.WRITE_TYPE_DEFAULT);
gatt.writeCharacteristic(characteristic);
```

Когда процесс записи завершится, будет `onCharacteristicWrite` метод `onCharacteristicWrite` вашего `BluetoothGattCallback` :

```
    @Override
    public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
        super.onCharacteristicWrite(gatt, characteristic, status);
        Log.d(TAG, "Characteristic " + characteristic.getUuid() + " written);
    }
```

Основная операция записи выполняется следующим образом:

```
gatt.readCharacteristic(characteristic);
```

Когда процесс записи завершится, будет `onCharacteristicRead` метод `onCharacteristicRead` вашего `BluetoothGattCallback` :

```
@Override
public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
  super.onCharacteristicRead(gatt, characteristic, status);
  byte[] value = characteristic.getValue();
  }
```

Подписка на уведомления с сервера Gatt

Вы можете запросить уведомление у сервера Gatt при изменении значения признака:

```
gatt.setCharacteristicNotification(characteristic, true);
BluetoothGattDescriptor descriptor = characteristic.getDescriptor(
```

```
    UUID.fromString("00002902-0000-1000-8000-00805f9b34fb");
descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
mBluetoothGatt.writeDescriptor(descriptor);
```

Все уведомления с сервера будут получены в методе `onCharacteristicChanged` вашего BluetoothGattCallback:

```
    @Override
    public void onCharacteristicChanged(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic) {
        super.onCharacteristicChanged(gatt, characteristic);
        byte[] newValue = characteristic.getValue();
}
```

Реклама устройства BLE

Вы можете использовать Bluetooth LE Advertising для передачи пакетов данных всем соседним устройствам без необходимости устанавливать соединение в первую очередь. Имейте в виду, что существует строгий предел 31 байта рекламных данных. Реклама вашего устройства также является первым шагом к тому, чтобы другие пользователи могли подключиться к вам.

Поскольку не все устройства поддерживают Bluetooth LE Advertising, первым шагом является проверка того, что ваше устройство имеет все необходимые требования для его поддержки. После этого вы можете инициализировать объект `BluetoothLeAdvertiser` и с ним вы можете начать рекламные операции:

```
        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);
        }
```

Использование сервера Gatt

Чтобы ваше устройство выступало в качестве периферии, сначала вам нужно открыть `BluetoothGattServer` и заполнить его, по крайней мере, одним `BluetoothGattService` и одним `BluetoothGattCharacteristic` :

```
BluetoothGattServer server=bluetoothManager.openGattServer(context,
bluetoothGattServerCallback);

BluetoothGattService service = new BluetoothGattService(SERVICE_UUID,
BluetoothGattService.SERVICE_TYPE_PRIMARY);
```

Это пример BluetoothGattCharacteristic с полными правами на запись, чтение и уведомление. В соответствии с вашими потребностями вы можете настроить мелодию разрешений, предоставляемых этим признаком:

```
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);
```

`BluetoothGattServerCallback` отвечает за получение всех событий, связанных с вашим `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);
                }
```

Всякий раз, когда вы получаете запрос на запись / чтение на характеристику или дескриптор, вы должны отправить ответ на него, чтобы запрос был успешно завершен:

```
@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);
}
```

Прочитайте Низкая энергия Bluetooth онлайн: https://riptutorial.com/ru/android/topic/10020/низкая-энергия-bluetooth

глава 177: Нить

Examples

Пример темы с описанием

При запуске приложения сначала выполняется основной поток. Этот основной поток обрабатывает все концепции приложения пользовательского интерфейса. Если мы хотим долго запускать задачу, в которой нам не нужен пользовательский интерфейс, мы используем thread для выполнения этой задачи в фоновом режиме.

Вот пример Thread, который описывает удар:

```
new Thread(new Runnable() {
    public void run() {
        for(int i = 1; i < 5;i++) {
            System.out.println(i);
        }
    }
}).start();
```

Мы можем создать поток, создав объект Thread, у которого есть `Thread.run()` для запуска thread.Here, метод `run()` вызывается методом `start()` .

Мы также можем запускать несколько потоков независимо друг от друга, что называется MultiThreading. В этом потоке также есть функциональность сна, с помощью которой текущий исполняемый поток будет спать (временно прекратить выполнение) за указанное количество времени. Но sleep бросает InterruptedException Итак, мы должны справиться с этим, используя try / catch, как это.

```
try{Thread.sleep(500);}catch(InterruptedException e){System.out.println(e);}
```

Обновление пользовательского интерфейса из фоновой темы

Обычно используется фоновый поток для выполнения сетевых операций или длительных задач, а затем обновлять пользовательский интерфейс при необходимости.

Это создает проблему, так как только основной поток может обновлять пользовательский интерфейс.

Решение заключается в использовании `runOnUiThread()` , поскольку он позволяет вам инициировать выполнение кода в потоке пользовательского интерфейса из фонового потока.

В этом простом примере поток запускается при создании Activity, выполняется до тех пор,

пока магическое число `42` будет произвольно сгенерировано, а затем использует метод `runOnUiThread()` для обновления пользовательского интерфейса после выполнения этого условия.

```
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();
    }
}
```

Прочитайте Нить онлайн: https://riptutorial.com/ru/android/topic/7131/нить

глава 178: Нож для масла

Вступление

Butterknife - это инструмент привязки к виду, который использует аннотации для создания шаблона кода для нас. Этот инструмент разработан Jake Wharton at Square и по существу используется для сохранения повторяющихся строк кода, таких как `findViewById(R.id.view)` при работе с представлениями, что делает наш код намного более чистым.

Чтобы быть ясным, Butterknife **не является библиотекой инъекций зависимости** . Butterknife вводит код во время компиляции. Это очень похоже на работу Аннотаций Android.

замечания

Нож для масла

Связывание полей и методов для представлений Android, использующих обработку аннотаций для создания шаблона кода для вас.

- Устраните вызовы findViewById с помощью полей @BindView.
- Группируйте несколько видов в списке или массиве. Действуйте со всеми из них сразу с действиями, сеттерами или свойствами.
- Устранение анонимных внутренних классов для слушателей путем аннотирования методов с помощью @OnClick и других.
- Устраните поиск ресурсов, используя аннотации ресурсов в полях.

Дополнительная информация: http://jakewharton.github.io/butterknife/

Лицензия

Examples

Настройка ButterKnife в вашем проекте

Настройте проект `build.gradle` на уровне `build.gradle` чтобы включить плагин `android-apt` :

```
buildscript {
   repositories {
      mavenCentral()
   }

   dependencies {
      classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
   }
}
```

Затем примените плагин `android-apt` в модуле `build.gradle` уровне `build.gradle` и добавьте зависимости ButterKnife:

```
apply plugin: 'android-apt'

android {
    ...
}

dependencies {
      compile 'com.jakewharton:butterknife:8.5.1'
      annotationProcessor 'com.jakewharton:butterknife-compiler:8.5.1'
}
```

Примечание. Если вы используете новый компилятор Jack с версией 2.2.0 или новее, вам не нужен плагин `android-apt` и вместо него вместо apt можно использовать `annotationProcessor` при объявлении зависимости компилятора.

Чтобы использовать аннотации ButterKnife, вы не должны забывать о привязке их к `onCreate()` ваших действий или `onCreateView()` ваших фрагментов:

```
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;
  }

}
```

Снимки версии разработки доступны в репозитории снимков Sonatype .

Ниже приведены дополнительные шаги, которые необходимо предпринять для использования ButterKnife в библиотечном проекте

Чтобы использовать ButterKnife в проекте библиотеки, добавьте плагин к вашему проекту `build.gradle` :

```
buildscript {
    dependencies {
        classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
    }
}
```

... и затем применить к вашему модулю, добавив эти строки в верхней части вашего уровня библиотеки `build.gradle` :

```
apply plugin: 'com.android.library'
// ...
apply plugin: 'com.jakewharton.butterknife'
```

Теперь убедитесь, что вы используете `R2` вместо `R` внутри всех аннотаций 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
  }

}
```

Связывание взглядов с использованием ButterKnife

мы можем аннотировать поля с `@BindView` и идентификатором вида Butter Knife, чтобы найти и автоматически `@BindView` соответствующий вид в нашем макете.

Привязки

Связывание в действии

```
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...
  }
}
```

Связывание в фрагментах

```
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();
    }
}
```

Связывание просмотров в диалогах

Мы можем использовать `ButterKnife.findById` для поиска представлений в представлении, активности или диалоге. Он использует generics для вывода возвращаемого типа и автоматически выполняет бросок.

```
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);
```

Связывание просмотров в 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);
    }
  }
```

Ресурсы привязки

Помимо того , что полезно для связывания точки зрения, можно было бы также использовать нож для масла , чтобы связать ресурсы , такие как те , которые определены в `strings.xml` , `drawables.xml` , `colors.xml` , `dimens.xml` и т.д.

```
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);
```

```
    }

}
```

Список просмотра привязки

Вы можете группировать несколько видов в список или массив. Это очень полезно, когда нам нужно выполнить одно действие сразу для нескольких видов.

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

Дополнительные привязки

По умолчанию `@Bind` привязки `@Bind` и прослушивателя. Исключение выдается, если целевое представление не может быть найдено. Но если мы не уверены, будет ли это вид или нет, мы можем добавить `@Nullable` аннотацию к полям или `@Optional` аннотации методам для подавления этого поведения и создания необязательной привязки.

```
@Nullable
@BindView(R.id.might_not_be_there) TextView mightNotBeThere;

@Optional
@OnClick(R.id.maybe_missing)
void onMaybeMissingClicked() {
  // TODO ...
}
```

Связывание слушателей с использованием ButterKnife

Слушатель OnClick:

```
@OnClick(R.id.login)
public void login(View view) {
  // Additional logic
}
```

Все аргументы метода слушателя являются необязательными:

```
@OnClick(R.id.login)
public void login() {
   // Additional logic
}
```

Конкретный тип будет автоматически лить:

```
@OnClick(R.id.submit)
public void sayHi(Button button) {
  button.setText("Hello!");
}
```

Несколько идентификаторов в одной привязке для обработки общих событий:

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

Пользовательские виды могут привязываться к своим собственным слушателям, не указывая идентификатор:

```
public class CustomButton extends Button {
  @OnClick
  public void onClick() {
    // TODO
  }
}
```

Несвязанные виды в ButterKnife

Фрагменты имеют иной жизненный цикл представления, чем действия. При привязке фрагмента в onCreateView задайте представления null в onDestroyView. Butter Knife возвращает экземпляр Unbinder, когда вы вызываете bind для этого. Вызовите его метод unbind в соответствующем обратном вызове жизненного цикла.

Пример:

```
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();
  }
}
```

> Примечание. Вызов unbind () в onDestroyView () не требуется, но рекомендуется, так как он сохраняет довольно немного памяти, если ваше приложение имеет большую заднюю часть.

Плагин Android Studio ButterKnife

Android ButterKnife Zelezny

> Плагин для генерации инъекций ButterKnife из выбранных макетов XML в действиях / фрагментах / адаптерах.

Примечание. Убедитесь, что вы сделали правильный щелчок для ***your_xml_layou*** `(R.layout.your_xml_layou`), иначе в *меню Generate* не будет использоваться опция форсунки Butterknife.

```java
/**
 * Main UI for setting up GridWichterle.
 *
 * @author Michal Matl (michal.matl@inmite.eu)
 */
public class SettingsActivity extends FragmentActivity {

    private Config mConfig;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);
        ButterKnife.inject(this);

        Intent intent = new Intent(this, GridOverlayService.class);
        startService(intent);

        setupViews();
    }
```

Ссылка: Плагин Jetbrains Android ButterKnife Zelezny

Прочитайте Нож для масла онлайн: https://riptutorial.com/ru/android/topic/1072/нож-для-масла

глава 179: область

Вступление

База данных Realm Mobile является альтернативой SQLite. База данных Realm Mobile намного быстрее, чем ORM, и часто быстрее, чем сырой SQLite.

Выгоды

Автономная функциональность, быстрые запросы, безопасная потоковая передача, кросс-платформенные приложения, шифрование, реактивная архитектура.

замечания

Когда вы используете Realm, вы должны помнить, что вы не должны передавать объекты RealmObjects, RealmResults и Realm между потоками. Если вам нужен запрос по данному потоку, откройте экземпляр Realm в этом потоке. При завершении потока вы должны закрыть Царство.

> ***ПРАВОВОЕ ПРИМЕЧАНИЕ*** . Вы понимаете, что Программное обеспечение может содержать криптографические функции, которые могут быть подвержены экспортным ограничениям, и вы представляете и гарантируете, что **вы не находитесь в стране, на** которую распространяется экспортное ограничение или эмбарго Соединенных Штатов, **включая Кубу, Иран, Север Кореи, Судана, Сирии или региона Крыма** и что вы не находитесь в списке запрещенных лиц Департамента торговли, Непроверенных Сторон или связаны с Ограниченной организацией.

Examples

Добавление королевства в ваш проект

Добавьте следующую зависимость для вашего уровня **проекта** `build.gradle` файла.

```
dependencies {
    classpath "io.realm:realm-gradle-plugin:3.1.2"
}
```

Добавьте в верхнюю часть файла `build.gradle` уровне **приложения** `build.gradle` .

```
apply plugin: 'realm-android'
```

Завершите синхронизацию градиента, и теперь вы добавили Realm в качестве зависимости

от вашего проекта!

Realm требует первоначального вызова с 2.0.0 перед его использованием. Вы можете сделать это в своем классе `Application` или в методе `onCreate` первого действия.

```
Realm.init(this); // added in Realm 2.0.0
Realm.setDefaultConfiguration(new RealmConfiguration.Builder().build());
```

Модели царства

Модели Realm должны расширять базовый класс `RealmObject` , они определяют схему базовой базы данных.

Поддерживаемые типы полей: `boolean` , `byte` , `short` , `int` , `long` , `float` , `double` , `String` , `Date` , `byte[]` , ссылки на другие `RealmObject` s и `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
}
```

Если вы добавите (или удалите) новое поле в свой RealmObject (или добавите новый класс RealmObject или удалите уже существующий), потребуется **перенастройка** . Вы можете либо установить `deleteIfMigrationNeeded()` в свой `RealmConfiguration.Builder` , либо определить необходимую миграцию. Миграция также требуется при добавлении (или удалении) `@Required` или `@Index` или `@PrimaryKey` .

Отношения должны устанавливаться вручную, они НЕ автоматические на основе первичных ключей.

Начиная с 0.88.0, также можно использовать публичные поля вместо частных полей / геттеров / сеттеров в классах RealmObject.

Также возможно реализовать `RealmModel` вместо расширения `RealmObject` , если класс также аннотируется с помощью `@RealmClass` .

```
@RealmClass
public class Person implements RealmModel {
    // ...
}
```

В этом случае такие методы, как `person.deleteFromRealm()` или `person.addChangeListener()` , заменяются `RealmObject.deleteFromRealm(person)` и `RealmObject.addChangeListener(person)` .

Ограничения заключаются в `RealmObject` , что `RealmObject` может быть расширен только `RealmObject` , и нет поддержки `final` , `volatile` и `transient` полей.

Важно, чтобы *управляемый* класс RealmObject мог быть изменен только в транзакции. *Управляемый* объект RealmObject не может быть передан между потоками.

Список примитивов (RealmList)

В настоящее время Realm не поддерживает сохранение списка примитивов. Он находится в списке задач (GitHub issue # 575), но пока что это обходной путь.

Создайте новый класс для вашего примитивного типа, это использует Integer, но измените его на все, что вы хотите сохранить.

```
public class RealmInteger extends RealmObject {
    private int val;

    public RealmInteger() {
    }

    public RealmInteger(int val) {
        this.val = val;
    }

    // Getters and setters
}
```

Теперь вы можете использовать это в своем `RealmObject` .

```
public class MainObject extends RealmObject {

    private String name;
    private RealmList<RealmInteger> ints;

    // Getters and setters
}
```

Если вы используете `GSON` для заполнения вашего `RealmObject` , вам нужно будет добавить адаптер настраиваемого типа.

```
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();
```

примерочных с-ресурсы

```
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
}
```

Ресурс Try with resources может использоваться только от KITKAT (minSDK 19)

Сортированные запросы

Чтобы отсортировать запрос, вместо использования `findAll()` вы должны использовать `findAllSorted()` .

```
RealmResults<SomeObject> results = realm.where(SomeObject.class)
                                        .findAllSorted("sortField", Sort.ASCENDING);
```

Замечания:

`sort()` возвращает полностью новые данные RealmResults, которые были отсортированы, но обновление этого RealmResults сбрасывает его. Если вы используете `sort()` , вы всегда должны повторно сортировать его в своем `RealmChangeListener` , удалите `RealmChangeListener` из предыдущих `RealmResults` и добавьте его в возвращаемые новые `RealmResults` . Использование `sort()` в `RealmResults` возвращаемом асинхронным запросом, который еще не загружен, завершится с ошибкой.

`findAllSorted()` всегда возвращает результаты, отсортированные по полю, даже если он обновляется. Рекомендуется использовать `findAllSorted()` .

Асинхронные запросы

Каждый синхронный метод запроса (например, `findAll()` или `findAllSorted()`) имеет асинхронный аналог (`findAllAsync()` / `findAllSortedAsync()`).

Асинхронные запросы выгружают оценку `RealmResults` в другой поток. Чтобы получить эти результаты в текущем потоке, текущий поток должен быть потоком петлителя (чтение: асинхронные запросы обычно работают только с потоком пользовательского интерфейса).

```
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);
```

Использование Realm с RxJava

Для запросов Realm предоставляет метод `realmResults.asObservable()` . Наблюдение результатов возможно только на потоках петлителя (как правило, в потоке пользовательского интерфейса).

Чтобы это работало, ваша конфигурация должна содержать следующее

```
realmConfiguration = new RealmConfiguration.Builder(context)       //
                         .rxFactory(new RealmObservableFactory()) //
                            //...
                         .build();
```

Впоследствии вы можете использовать свои результаты как наблюдаемые.

```
Observable<RealmResults<SomeObject>> observable = results.asObservable();
```

Для асинхронных запросов вы должны фильтровать результаты с помощью `isLoaded()` , чтобы вы получили событие только в том случае, когда запрос был выполнен. Этот `filter()` не требуется для синхронных запросов (`isLoaded()` всегда возвращает `true` при запросах синхронизации).

```
    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));
```

Для записи вам следует либо использовать метод `executeTransactionAsync()` , либо открыть экземпляр Realm в фоновом потоке, выполнить транзакцию синхронно, а затем закрыть экземпляр Realm.

```
public Subscription loadObjectsFromNetwork() {
    return objectApi.getObjects()
        .subscribeOn(Schedulers.io())
        .subscribe(response -> {
            try(Realm realmInstance = Realm.getDefaultInstance()) {
                realmInstance.executeTransaction(realm ->
realm.insertOrUpdate(response.objects));
            }
        });
}
```

Основное использование

Настройка экземпляра

Чтобы использовать Realm, вам сначала нужно получить его экземпляр. Каждый экземпляр Realm отображается в файл на диске. Самый простой способ получить экземпляр:

```
// 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();
```

Метод `Realm.getInstance()` создает файл базы данных, если он не был создан, иначе открывается файл. Объект `RealmConfiguration` контролирует все аспекты создания Realm - будь то `inMemory()` данных `inMemory()` , имя файла Realm, если Realm должен быть очищен, если требуется миграция, исходные данные и т. Д.

Обратите внимание, что вызовы `Realm.getInstance()` подсчитываются по ссылке (каждый вызов увеличивает счетчик), а счетчик уменьшается, когда `realm.close()` .

Закрытие экземпляра

В фоновых потоках *очень важно* ***закрыть*** экземпляр Realm, если он больше не используется (например, транзакция завершена и завершение потока завершено). Невозможность закрыть все экземпляры Realm в фоновом потоке приводит к фиксации версии и может привести к значительному увеличению размера файла.

```
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
```

Стоит отметить, что выше уровня API 19 можно заменить этот код следующим:

```
try(Realm realm = Realm.getDefaultInstance()) {
    // ...
}
```

модели

Следующим шагом будет создание ваших моделей. Здесь может быть задан вопрос: «Какова модель?». Модель представляет собой структуру, которая определяет свойства объекта, хранящегося в базе данных. Например, в следующем мы моделируем книгу.

```
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;
    }
}
```

Обратите внимание, что ваши модели должны расширять класс RealmObject. Основной ключ также указывается аннотацией `@PrimaryKey` . Первичные ключи могут быть нулевыми, но только один элемент может иметь `null` в качестве первичного ключа. Также вы можете использовать аннотацию `@Ignore` для полей, которые не должны сохраняться на диске:

```
@Ignore
private String isbn;
```

Вставка или обновление данных

Чтобы сохранить объект книги в экземпляр базы данных Realm, вы можете сначала создать экземпляр своей модели, а затем сохранить его в базе данных с помощью метода `copyToRealm` . Для создания или обновления вы можете использовать `copyToRealmOrUpdate` . (Более быстрая альтернатива - это новая добавленная `insertOrUpdate()`).

```
// Creating an instance of the model
Book book = new Book();
book.setId(1);
book.setTitle("Walking on air");
book.setAuthor("Taylor Swift")

// Store to the database
realm.executeTransaction(new Realm.Transaction() {
    @Override
    public void execute(Realm realm) {
        realm.insertOrUpdate(book);
    }
});
```

Обратите внимание, что все изменения в данных должны происходить в транзакции. Другой способ создания объекта - использовать следующий шаблон:

```
Book book = realm.createObject(Book.class, primaryKey);
...
```

Запрос базы данных

- Все книги:

```
RealmResults<Book> results = realm.where(Book.class).findAll();
```

- Все книги с идентификатором более 10:

```
RealmResults<Book> results = realm.where(Book.class)
                                  .greaterThan("id", 10)
                                  .findAll();
```

- Книги `'Taylor Swift'` или `'%Peter%'` :

```
RealmResults<Book> results = realm.where(Book.class)
                                  .beginGroup()
                                      .equalTo("author", "Taylor Swift")
                                      .or()
                                      .contains("author", "Peter")
                                  .endGroup().findAll();
```

Удаление объекта

Например, мы хотим удалить все книги Тейлора Свифта:

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

Прочитайте область онлайн: https://riptutorial.com/ru/android/topic/3187/область

глава 180: Обнаружение жеста

замечания

Официальная документация: Обнаружение общих жестов

Examples

Обнаружение прокрутки

```
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() {
    }

}
```

Применяется к представлению ...

```
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");
    }

});
```

Обнаружение основных жестов

```
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;
    }

}
```

Прочитайте Обнаружение жеста онлайн: https://riptutorial.com/ru/android/topic/4711/обнаружение-жеста

глава 181: Обнаружение события Shake в Android

Examples

Вейк-детектор в Android-примере

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

Использование обнаружения сейсмического тряски

Сейсмика - это Android-устройство для обнаружения тряски на Android. Чтобы использовать его, просто начинайте слушать события, возникающие при встряхивании.

```
@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();
}
```

Для того, чтобы определить различные порог ускорения использования `sd.setSensitivity(sensitivity)` с `sensitivity` по `SENSITIVITY_LIGHT` , `SENSITIVITY_MEDIUM` , `SENSITIVITY_HARD` или любого другого разумного целого значения. Данные значения по умолчанию варьируются от *11* до *15* .

Монтаж

```
compile 'com.squareup:seismic:1.0.2'
```

Прочитайте Обнаружение события Shake в Android онлайн:
https://riptutorial.com/ru/android/topic/4501/обнаружение-события-shake-в-android

глава 182: Обработка глубоких ссылок

Вступление

Глубокие ссылки - это URL-адреса, которые используют пользователей непосредственно для определенного контента в вашем приложении. Вы можете настроить глубокие ссылки, добавив фильтры намерений и извлечение данных из входящих намерений, чтобы привлечь пользователей на правый экран в вашем приложении.

параметры

`<data>` Атрибут	подробности
схема	Часть *схемы* URI (с учетом регистра). Примеры: `http` , `https` , `ftp`
хозяин	*Хост-* часть URI (с учетом регистра). Примеры: `google.com` , `example.org`
порт	Часть *порта* URI. Примеры: `80` , `443`
дорожка	Часть *пути* URI. Необходимо начинать с `/` . Примеры: `/` , `/about`
PATHPREFIX	Префикс для части *пути* URI. Примеры: `/item` , `/article`
pathPattern	Шаблон для соответствия для части *пути* URI. Примеры: `/item/.*` , `/article/[0-9]*`
MIMETYPE	Тип mime для соответствия. Примеры: `image/jpeg` , `audio/*`

замечания

`<intent-filter>`

Эта комбинация элементов `<action>` и `<category>` - это то, что сообщает системе Android, что определенное действие должно запускаться, когда пользователь нажимает ссылку в другом приложении.

```
<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>
```

Несколько тегов `<data>`

Набор глубоких ссылок, которые поддерживает ваш `<intent-filter>` является кросс-продуктом всех элементов `<data>` которые вы определяете в этом фильтре-намерении. Это демонстрируют примеры с несколькими доменами, несколькими путями и несколькими схемами.

Ресурсы

- Включение Deep Links для содержимого приложения (developer.android.com)
- `<intent-filter>` (developer.android.com

Examples

Простая глубокая ссылка

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>
```

Это позволит принять любую ссылку, начиная с `http://www.example.com` в качестве глубокой ссылки, чтобы начать свою `MainActivity` .

Несколько путей в одном домене

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>
```

Это запустит вашу `MainActivity` когда пользователь нажмет на любую из этих ссылок:

- `http://www.example.com/`
- `http://www.example.com/about`
- `http://www.example.com/map`

Несколько доменов и несколько путей

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>
```

Это запустит вашу MainActivity, когда пользователь нажмет на любую из этих ссылок:

- `http://www.example.com/`
- `http://www.example2.com/`
- `http://www.example.com/map`
- `http://www.example2.com/map`

И HTTP, и https для одного домена

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>
```

Это запустит вашу MainActivity, когда пользователь нажмет на любую из этих ссылок:

- `http://www.example.com/`
- `https://www.example.com/`
- `http://www.example.com/map`
- `https://www.example.com/map`

Получение параметров запроса

```
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");
        }
    }

}
```

Если пользователь нажимает ссылку на `http://www.example.com/map?param1=FOO&param2=BAR` , тогда `param1` здесь будет иметь значение `"FOO"` а `param2` будет иметь значение `"BAR"` .

Использование 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>
```

Это запустит вашу MainActivity, когда пользователь нажмет любую ссылку, начиная с `http://www.example.com/item` , например:

- `https://www.example.com/item`
- `http://www.example.com/item/1234`
- `https://www.example.com/item/xyz/details`

Прочитайте Обработка глубоких ссылок онлайн: https://riptutorial.com/ru/android/topic/3716/обработка-глубоких-ссылок

глава 183: Обработка событий касания и движения

Вступление

Краткое изложение некоторых основных сенсорных / движителей в Android API.

параметры

слушатель	подробности
onTouchListener	Ручки с одним касанием для кнопок, поверхностей и многое другое
onTouchEvent	Слушатель, который можно найти на поверхностях (например, SurfaceView). Не нужно устанавливать, как другие слушатели (e, g. OnTouchListener)
onLongTouch	Подобно onTouch, но прослушивает длинные нажатия кнопок, поверхностей и т. Д.

Examples

Кнопки

Сенсорные события, связанные с `Button` можно проверить следующим образом:

```
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;
    }
}
```

поверхность

Сенсорный обработчик событий для поверхностей (например, SurfaceView, GLSurfaceView и другие):

```
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;
    }
}
```

Или, альтернативно (на поверхности):

```
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;
    }
}
```

Обработка мультитачей на поверхности

```
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.
    }
}
```

Прочитайте Обработка событий касания и движения онлайн:
https://riptutorial.com/ru/android/topic/9315/обработка-событий-касания-и-движения

глава 184: Обработчик аннотации

Вступление

Обработчик аннотации - это инструмент, созданный в javac для сканирования и обработки аннотаций во время компиляции.

Аннотации - это класс метаданных, которые могут быть связаны с классами, методами, полями и даже другими аннотациями. Существует два способа доступа к этим аннотациям во время выполнения через отражение и во время компиляции с помощью обработчиков аннотаций.

Examples

@NonNull Annotation

```
public class Foo {
    private String name;
    public Foo(@NonNull String name){...};
    ...
}
```

Здесь @NonNull - это аннотация, которая обрабатывается временем компиляции студией android, чтобы предупредить вас о том, что для конкретной функции нужен параметр non null.

Типы аннотаций

Существует три типа аннотаций.

1. **Маркерная аннотация** - аннотация, не имеющая метода

   ```
   @interface CustomAnnotation {}
   ```

2. **Однозначная аннотация** - аннотация, которая имеет один метод

   ```
   @interface CustomAnnotation {
       int value();
   }
   ```

3. **Многозначная аннотация** - аннотация, содержащая несколько методов

   ```
   @interface CustomAnnotation{
       int value1();
       String value2();
   ```

```
    String value3();
}
```

Создание и использование пользовательских аннотаций

Для создания пользовательских аннотаций нам необходимо решить

- Цель - по которой эти аннотации будут работать как на уровне поля, уровне метода, так и на уровне типа и т. Д.
- Сохранение - на каком уровне будет доступна аннотация.

Для этого мы создали пользовательские аннотации. Ознакомьтесь с этими наиболее часто используемыми:

@Target

Element Types	Where the a
TYPE	class, interfac
FIELD	fields
METHOD	methods
CONSTRUCTOR	constructors
LOCAL_VARIABLE	local variable
ANNOTATION_TYPE	annotation ty
PARAMETER	parameter

@Retention

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

Создание пользовательской аннотации

```
@Retention(RetentionPolicy.SOURCE) // will not be available in compiled class
@Target(ElementType.METHOD) // can be applied to methods only
@interface CustomAnnotation{
      int value();
}
```

Использование пользовательской аннотации

```
class Foo{
  @CustomAnnotation(value = 1)  // will be used by an annotation processor
  public void foo(){..}
}
```

значение, предоставленное внутри `@CustomAnnotation` будет потребляться процессором Annotation, это может генерировать код во время компиляции и т. д.

Прочитайте Обработчик аннотации онлайн: https://riptutorial.com/ru/android/topic/10726/обработчик-аннотации

глава 185: Обратный вызов

Examples

Пример обратного вызова с помощью Instagram OAuth

Одним из вариантов использования *обратных вызовов* является OAuth. Давайте сделаем это с помощью входа в Instagram: если пользователь вводит свои учетные данные и нажимает кнопку « *Вход»* , Instagram будет проверять учетные данные и возвращать `access_token` . Нам нужно, чтобы `access_token` в нашем приложении.

Чтобы наше приложение могло слушать такие ссылки, нам нужно добавить URL-адрес обратного вызова к нашей `Activity` . Мы можем сделать это, добавив в нашу `Activity` `<intent-filter/>` , который будет реагировать на этот URL обратного вызова. Предположим, что наш URL обратного вызова - `appSchema://appName.com` . Затем вы должны добавить следующие строки в свою желаемую `Activity` в файле *Manifest.xml* :

```
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.BROWSABLE"/>
<data android:host="appName.com" android:scheme="appSchema"/>
```

Объяснение строк выше:

- `<category android:name="android.intent.category.BROWSABLE"/>` позволяет целевой активности разрешить запуск веб-браузера для отображения данных, на которые ссылается ссылка.
- `<data android:host="appName.com" android:scheme="appSchema"/>` указывает нашу схему и хост нашего URL-адреса обратного вызова.
- Все вместе эти строки вызовут открытие определенного `Activity` , когда URL-адрес обратного вызова вызывается в браузере.

Теперь, чтобы получить содержимое URL-адреса в вашей `Activity` , вам необходимо переопределить метод `onResume()` следующим образом:

```
@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.
}
```

Теперь вы получили `access_token` из Instagram, который используется в различных конечных

точках API Instagram.

Прочитайте Обратный вызов онлайн: https://riptutorial.com/ru/android/topic/4790/обратный-вызов

глава 186: обслуживание

Вступление

Служба работает в **фоновом режиме** для выполнения длительных операций или для выполнения работы для удаленных процессов. Служба не предоставляет никакого пользовательского интерфейса, который выполняется только в фоновом режиме с помощью ввода пользователя. Например, служба может воспроизводить музыку в фоновом режиме, когда пользователь находится в другом приложении, или может загружать данные из Интернета, не блокируя взаимодействие пользователя с устройством Android.

замечания

Если вы еще не определили свою службу в своем AndroidManifest.xml, при попытке запустить ее вы получите `ServiceNotFoundException` .

Замечания:

Информацию о `IntentService` см. Здесь: Пример IntentService

Examples

Запуск службы

Запуск службы очень прост, просто позвоните `startService` с намерением, изнутри Activity:

```
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.
```

Жизненный цикл службы

Жизненный цикл службы имеет следующие обратные вызовы

- `onCreate()` :

Выполняется, когда служба сначала создается для настройки исходных конфигураций, которые могут вам понадобиться. Этот метод выполняется только в том случае, если служба еще не запущена.

- `onStartCommand()` :

Выполняется каждый раз, когда `startService()` вызывается другим компонентом, например Activity или BroadcastReceiver. Когда вы используете этот метод, Служба будет работать до тех пор, пока вы не назовете `stopSelf()` или `stopService()` . Обратите внимание, что независимо от того, сколько раз вы вызываете `onStartCommand()` , методы `stopSelf()` и `stopService()` должны быть вызваны только один раз, чтобы остановить службу.

- `onBind()` :

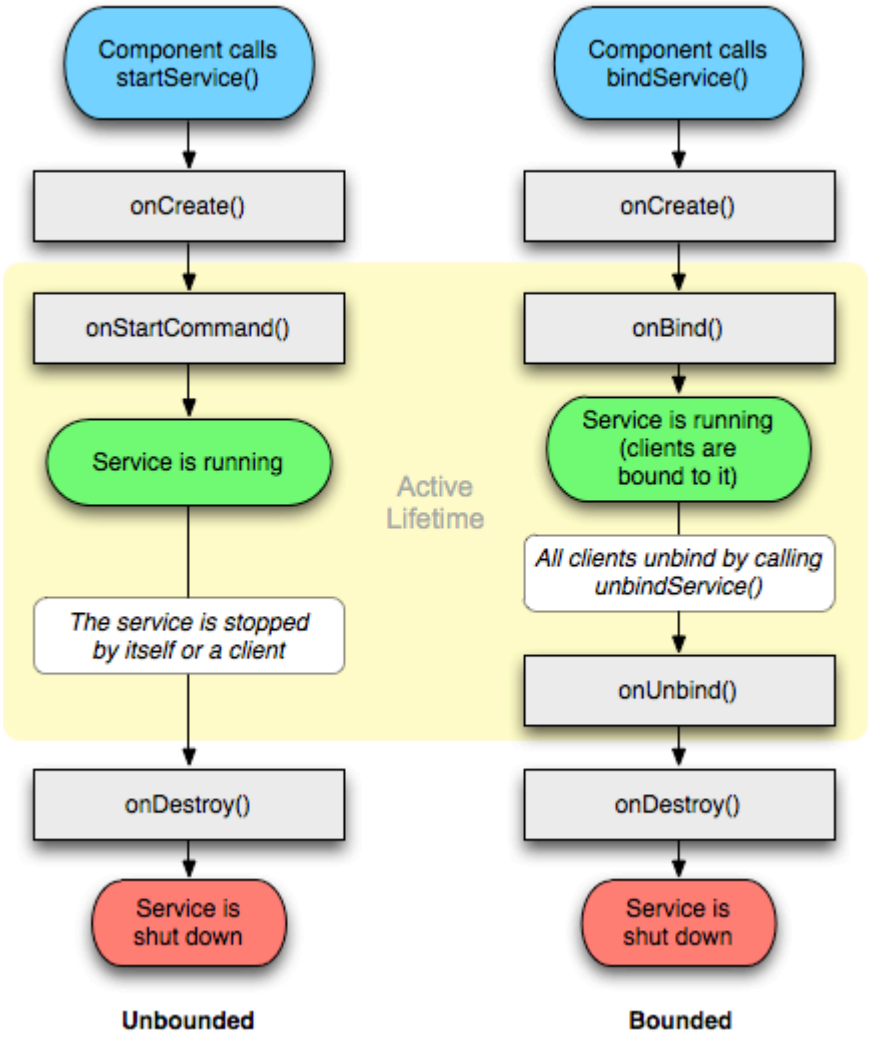
Выполняется, когда компонент вызывает `bindService()` и возвращает экземпляр IBInder, предоставляя канал связи Службе. Вызов функции `bindService()` будет поддерживать работу службы, если к ней привязаны клиенты.

- `onDestroy()` :

Выполняется, когда служба больше не используется и позволяет удалять ресурсы, которые были выделены.

Важно отметить, что в течение жизненного цикла службы могут быть вызваны другие обратные вызовы, такие как `onConfigurationChanged()` и `onLowMemory()`

https://developer.android.com/guide/components/services.html



Определение процесса обслуживания

Поле `android:process` определяет имя процесса, в котором должна запускаться служба. Обычно все компоненты приложения запускаются в процессе по умолчанию, созданном для приложения. Однако компонент может переопределить значение по умолчанию со своим собственным атрибутом процесса, что позволяет распространять ваше приложение на нескольких процессах.

Если имя, присвоенное этому атрибуту, начинается с двоеточия (':'), служба будет запускаться в отдельном отдельном процессе.

```
<service
  android:name="com.example.appName"
  android:process=":externalProcess" />
```

Если имя процесса начинается с символа в нижнем регистре, служба будет выполняться в глобальном процессе этого имени при условии, что у него есть разрешение на это. Это позволяет компонентам в разных приложениях совместно использовать процесс, уменьшая использование ресурсов.

Создание связанного сервиса с помощью Binder

Создайте класс, который расширяет класс `Service` и переопределенный метод `onBind` возвращает ваш локальный экземпляр связующего:

```
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;
    }
}
```

Затем в вашей активности `onStart` с сервисом в `onStart` , используя экземпляр `ServiceConnection` и отвяжите его в `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;
        }
    };
}
```

Создание удаленного сервиса (через AIDL)

Опишите свой интерфейс доступа к сервису через файл `.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();
```

```
}
```

Теперь после сборки приложение sdk tools создаст соответствующий `.java` файл. Этот файл будет содержать класс `Stub` который реализует наш интерфейс helpl и который нам нужно расширить:

```
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;
    }
}
```

Затем в действии:

```
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;
    }
}
```

Создание несвязанного сервиса

Первое, что нужно сделать, - добавить службу в `AndroidManifest.xml` внутри `<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>
```

Если вы намерены управлять классом обслуживания в отдельном пакете (например: . AllServices.RecordingService), вам необходимо указать, где находится ваша служба. Итак, в приведенном выше случае мы изменим:

```
android:name=".RecordingService"
```

в

```
android:name=".AllServices.RecordingService"
```

или самый простой способ сделать это - указать полное имя пакета.

Затем мы создаем фактический класс обслуживания:

```
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();
    }

}
```

Вся эта услуга показывает уведомление при запуске и может отображать тосты, когда `doSomething()` метод `doSomething()` .

Как вы заметили, он реализован как одноэлементный , отслеживая свой собственный экземпляр, но без обычного статического однопользовательского метода, поскольку службы являются, естественно, одноточечными и создаются намерениями. Экземпляр полезен снаружи, чтобы получить «дескриптор» службы при ее запуске.

Наконец, нам нужно запустить и остановить службу из одной активности:

```
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);
    }
}
```

В этом примере служба запускается и останавливается одним и тем же методом, в зависимости от текущего состояния.

Мы также можем вызвать метод `doSomething()` из нашей деятельности:

```
public void makeServiceDoSomething(){
    if( RecordingService.isRunning )
        RecordingService.instance.doSomething();
}
```

Прочитайте обслуживание онлайн: https://riptutorial.com/ru/android/topic/137/обслуживание

глава 187: Окио

Examples

Загрузить / Внедрить

Загрузите последнюю версию JAR или возьмите Maven:

```
<dependency>
    <groupId>com.squareup.okio</groupId>
    <artifactId>okio</artifactId>
    <version>1.12.0</version>
</dependency>
```

или Gradle:

```
compile 'com.squareup.okio:okio:1.12.0'
```

PNG-декодер

Декодирование фрагментов PNG-файла демонстрирует на практике Окио.

```
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 и Buffers

ByteStrings и Buffers

Окио построено на двух типах, которые добавляют много возможностей в простой API:

ByteString - неизменная последовательность байтов. Для символьных данных String является фундаментальным. ByteString - потерянный брат String, позволяющий легко обрабатывать двоичные данные как ценность. Этот класс эргономичен: он умеет кодировать и декодировать себя как hex, base64 и UTF-8.

Буфер - изменяемая последовательность байтов. Как ArrayList, вам не нужно заранее настраивать буфер. Вы читаете и записываете буферы в очередь: записываете данные до конца и читаете их спереди. Нет никаких обязательств по управлению позициями, лимитами или возможностями.

Внутренне, `ByteString` и `Buffer` делают некоторые умные вещи для экономии процессора и памяти. Если вы кодируете строку UTF-8 как ByteString, она кэширует ссылку на эту строку, так что, если вы ее декодируете позже, работы не будет.

`Buffer` реализуется как связанный список сегментов. Когда вы перемещаете данные из одного буфера в другой, он переназначает владение сегментами, а не копирует данные. Этот подход особенно полезен для многопоточных программ: поток, который ведет переговоры с сетью, может обмениваться данными с рабочим потоком без какого-либо копирования или церемонии.

Прочитайте Окио онлайн: https://riptutorial.com/ru/android/topic/9952/окио

глава 188: Определить значение шага (приращение) для пользовательского RangeSeekBar

Вступление

Настройка Android RangeSeekBar, предложенная Алексом Флореску на странице https://github.com/anothem/android-range-seek-bar

Он позволяет определять значение шага (приращение) при перемещении строки поиска

замечания

1- Добавьте атрибут increment в attrs.xml

```
<attr name="increment" format="integer|float"/>
```

2- Определите значение по умолчанию в RangeSeekBar.java и создайте атрибут также

```
private static final int DEFAULT_INCREMENT = 1;
private int increment;
```

3- Инициировать значение приращения в private void init (контекст контекста, AttributeSet attrs)

```
if (attrs == null)
    increment = DEFAULT_INCREMENT;
else
    increment = a.getInt(R.styleable.RangeSeekBar_increment, DEFAULT_INCREMENT);
```

4- Определить значение приращения в защищенной синхронизированной пустоте onDraw (холст холста @NonNull)

Вам нужно будет заменить значение minText и maxText. Поэтому вместо:

- minText = valueToString (getSelectedMinValue ());
- maxText = valueToString (getSelectedMaxValue ());

У вас будет: 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 - Теперь вам просто нужно это использовать. Надеюсь, поможет

Examples

Определить значение шага 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" />
```

Прочитайте Определить значение шага (приращение) для пользовательского RangeSeekBar онлайн: https://riptutorial.com/ru/android/topic/8627/определить-значение-шага--приращение--для-пользовательского-rangeseekbar

глава 189: Оптимизация производительности

Вступление

Производительность ваших приложений является ключевым элементом пользовательского интерфейса. Старайтесь избегать плохих моделей, таких как работа над потоком пользовательского интерфейса и научиться писать быстрые и гибкие приложения.

Examples

Сохранить просмотр с помощью шаблона ViewHolder

Особенно в `ListView` вы можете столкнуться с проблемами производительности, выполняя слишком много вызовов `findViewById()` во время прокрутки. Используя шаблон `ViewHolder` , вы можете сохранить эти поисковые запросы и улучшить производительность `ListView` .

Если элемент списка содержит один `TextView` , создайте класс `ViewHolder` для хранения экземпляра:

```
static class ViewHolder {
    TextView myTextView;
}
```

При создании элемента списка присоедините объект `ViewHolder` к элементу списка:

```
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;
}
```

Используя эту модель, `findViewById()` будет вызываться только тогда , когда новый `View` создаются и `ListView` может гораздо более эффективно переработать свои взгляды.

Прочитайте Оптимизация производительности онлайн:
https://riptutorial.com/ru/android/topic/8711/оптимизация-производительности

глава 190: Оптимизация ядра Android

Examples

Низкая конфигурация ОЗУ

Android теперь поддерживает устройства с 512 МБ ОЗУ. Эта документация призвана помочь OEM-производителям оптимизировать и настроить Android 4.4 для устройств с малой памятью. Некоторые из этих оптимизаций достаточно универсальны, чтобы их можно было применять и к предыдущим выпускам.

Включить флаг с низким уровнем защиты устройства

Мы вводим новый API под названием ActivityManager.isLowRamDevice () для приложений, чтобы определить, должны ли они отключать определенные функции интенсивной работы с памятью, которые плохо работают на устройствах с низкой памятью.

Для устройств с 512 МБ ожидается, что этот API вернется: «true». Это может быть включено следующим системным свойством в файле makefile.

```
PRODUCT_PROPERTY_OVERRIDES += ro.config.low_ram=true
```

Отключить JIT

Общесистемное использование JIT-памяти зависит от количества запущенных приложений и от объема этих приложений. JIT устанавливает максимальный размер кэша кеша и затрагивает страницы в нем по мере необходимости. JIT стоит где-то между 3M и 6M в типичной операционной системе.

Большие приложения стремятся максимально быстро удалить кеш-код (который по умолчанию был 1M). В среднем, использование кеша JIT работает где-то между 100K и 200 K байт на приложение. Уменьшение максимального размера кеша может несколько помочь в использовании памяти, но если слишком низкий уровень, он отправит JIT в режим обжатия. Для устройств с очень низкой памятью мы рекомендуем полностью отключить JIT.

Этого можно добиться, добавив следующую строку в файл makefile:

```
PRODUCT_PROPERTY_OVERRIDES += dalvik.vm.jit.codecachesize=0
```

Как добавить контроллер процессора

Сам регулятор ЦП - это всего лишь 1 C-файл, который находится в файле kernel_source /

drivers / cpufreq /, например: cpufreq_smartass2.c. Вы сами отвечаете за поиск губернатора (смотрите в существующем ядре репо для своего устройства). Но для успешного вызова и компиляции этого файла в ваше ядро вам придется внести следующие изменения:

1. Скопируйте файл-регулятор (cpufreq_govname.c) и перейдите на kernel_source / drivers / cpufreq, а затем вставьте его.
2. и открыть Kconfig (это интерфейс макета меню config) при добавлении ядра, вы хотите, чтобы он отображался в вашей конфигурации. Вы можете сделать это, добавив выбор губернатора.

```
config CPU_FREQ_GOV_GOVNAMEHERE
tristate "'gov_name_lowercase' cpufreq governor"
depends on CPU_FREQ
help
governor' - a custom governor!
```

например, для smartassV2.

```
config CPU_FREQ_GOV_SMARTASS2
 tristate "'smartassV2' cpufreq governor"
 depends on CPU_FREQ
 help
 'smartassV2' - a "smart" optimized governor!
```

рядом с добавлением выбора, вы также должны заявить, что губернатор выбирается в качестве регулятора по умолчанию.

```
 config CPU_FREQ_DEFAULT_GOV_GOVNAMEHERE
 bool "gov_name_lowercase"
 select CPU_FREQ_GOV_GOVNAMEHERE
 help
 Use the CPUFreq governor 'govname' as default.
```

например, для smartassV2.

```
config CPU_FREQ_DEFAULT_GOV_SMARTASS2
 bool "smartass2"
 select CPU_FREQ_GOV_SMARTASS2
 help
 Use the CPUFreq governor 'smartassV2' as default.
```

- Не можете найти нужное место для его размещения? Просто найдите `"CPU_FREQ_GOV_CONSERVATIVE"` и поместите код ниже, то же самое `"CPU_FREQ_DEFAULT_GOV_CONSERVATIVE"` К `"CPU_FREQ_DEFAULT_GOV_CONSERVATIVE"`

Теперь, когда Kconfig закончен, вы можете сохранить и закрыть файл.

3. Еще находясь в папке `/drivers/cpufreq` , откройте Makefile. В Makefile добавьте строку, соответствующую вашему контроллеру CPU. например:

```
obj-$(CONFIG_CPU_FREQ_GOV_SMARTASS2)    += cpufreq_smartass2.o
```

Будь то, что вы не вызываете собственный файл C, но файл O! который является скомпилированным C-файлом. Сохраните файл.

4. Перейдите в: `kernel_source/includes/linux` . Теперь откройте `cpufreq.h` Прокрутите вниз, пока не увидите что-то вроде:

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_ONDEMAND)
 extern struct cpufreq_governor cpufreq_gov_ondemand;
 #define CPUFREQ_DEFAULT_GOVERNOR    (&amp;cpufreq_gov_ondemand)
```

(там также перечислены другие регуляторы процессора)

Теперь добавьте свою запись с выбранным контроллером процессора, например:

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_SMARTASS2)
 extern struct cpufreq_governor cpufreq_gov_smartass2;
 #define CPUFREQ_DEFAULT_GOVERNOR (&amp;cpufreq_gov_smartass2)
```

Сохраните файл и закройте его.

Начальная настройка регулятора CPU завершена. когда вы выполнили все шаги успешно, вы сможете выбрать своего губернатора из меню (`menuconfig` , `xconfig` , `gconfig` , `nconfig`). После проверки в меню он будет включен в ядро.

Зафиксируйте это почти так же, как указано выше: добавьте smartassV2 и lulzactive commandor commit

Планировщики ввода-вывода

Вы можете улучшить свое ядро, добавив новые планировщики ввода-вывода, если это необходимо. В глобальном масштабе правители и планировщики одинаковы; они оба обеспечивают способ работы системы. Однако для планировщиков это все о потоке данных ввода / вывода, за исключением настроек ЦП. Планировщики ввода-вывода решают, как будет запланирована предстоящая операция ввода-вывода. Стандартные планировщики, такие как *noop* или *cfq* , выполняют очень разумно.

Планировщики ввода-вывода можно найти в *файле kernel_source / block* .

1. Скопируйте файл планировщика ввода-вывода (например, *sio-iosched.c*) и перейдите на *kernel_source / block* . Вставьте туда файл планировщика.

2. Теперь откройте *Kconfig.iosched* и добавьте свой выбор в *Kconfig* , например, для *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. Затем установите опцию выбора по умолчанию следующим образом:

```
default "sio" if DEFAULT_SIO
```

   Сохраните файл.

4. Откройте *Makefile* в *файле kernel_source / block /* и просто добавьте следующую строку для *SIO* :

```
obj-$(CONFIG_IOSCHED_SIO)    += sio-iosched.o
```

   Сохраните файл, и все готово! Планировщики ввода-вывода должны теперь отображаться в конфигурации меню.

Аналогичная фиксация на GitHub: добавлен простой планировщик ввода-вывода .

Прочитайте Оптимизация ядра Android онлайн: https://riptutorial.com/ru/android/topic/9106/оптимизация-ядра-android

глава 191: Оптимизированный видеообзор

Вступление

Воспроизведение видео с помощью `VideoView` которое расширяет `SurfaceView` внутри строки `ListView` кажется, работает сначала, пока пользователь не попытается прокрутить список. Как только список начинает прокручиваться, видео становится черным (иногда отображается белый). Он продолжает играть в фоновом режиме, но вы больше не видите его, потому что он отображает остальную часть видео как черный ящик. С помощью пользовательского оптимизированного видеообъявления видео будут воспроизводиться на прокрутке в `ListView` же, как наша Instagram, Facebook, Twitter.

Examples

Оптимизированный видеообзор в ListView

Это пользовательский `VideoView` который вам нужен для вашего пакета.

Пользовательский макет видеоизображения:

```
<your.packagename.VideoView
    android:id="@+id/video_view"
    android:layout_width="300dp"
    android:layout_height="300dp" />
```

Код для настраиваемого оптимизированного `VideoView` :

```
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);
    }
}
```

Справка из этого репозитория gitub . Хотя у Него есть некоторые проблемы, как это было написано 3 года назад, мне удалось исправить их самостоятельно, как написано выше.

Прочитайте Оптимизированный видеообзор онлайн:
https://riptutorial.com/ru/android/topic/10638/оптимизированный-видеообзор

глава 192: Опубликовать в Play Store

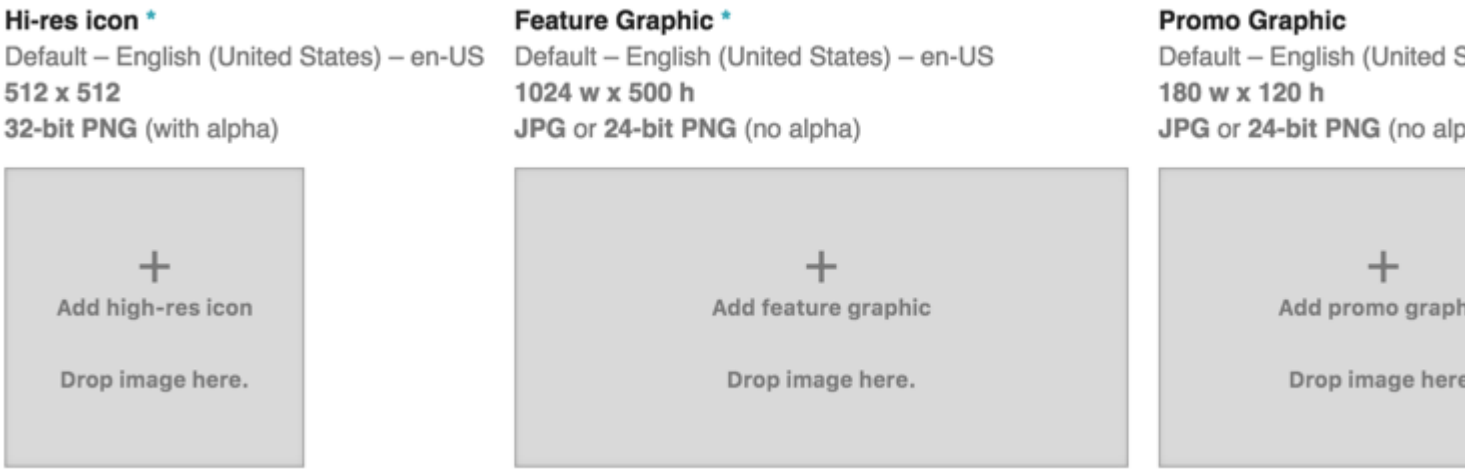
Examples

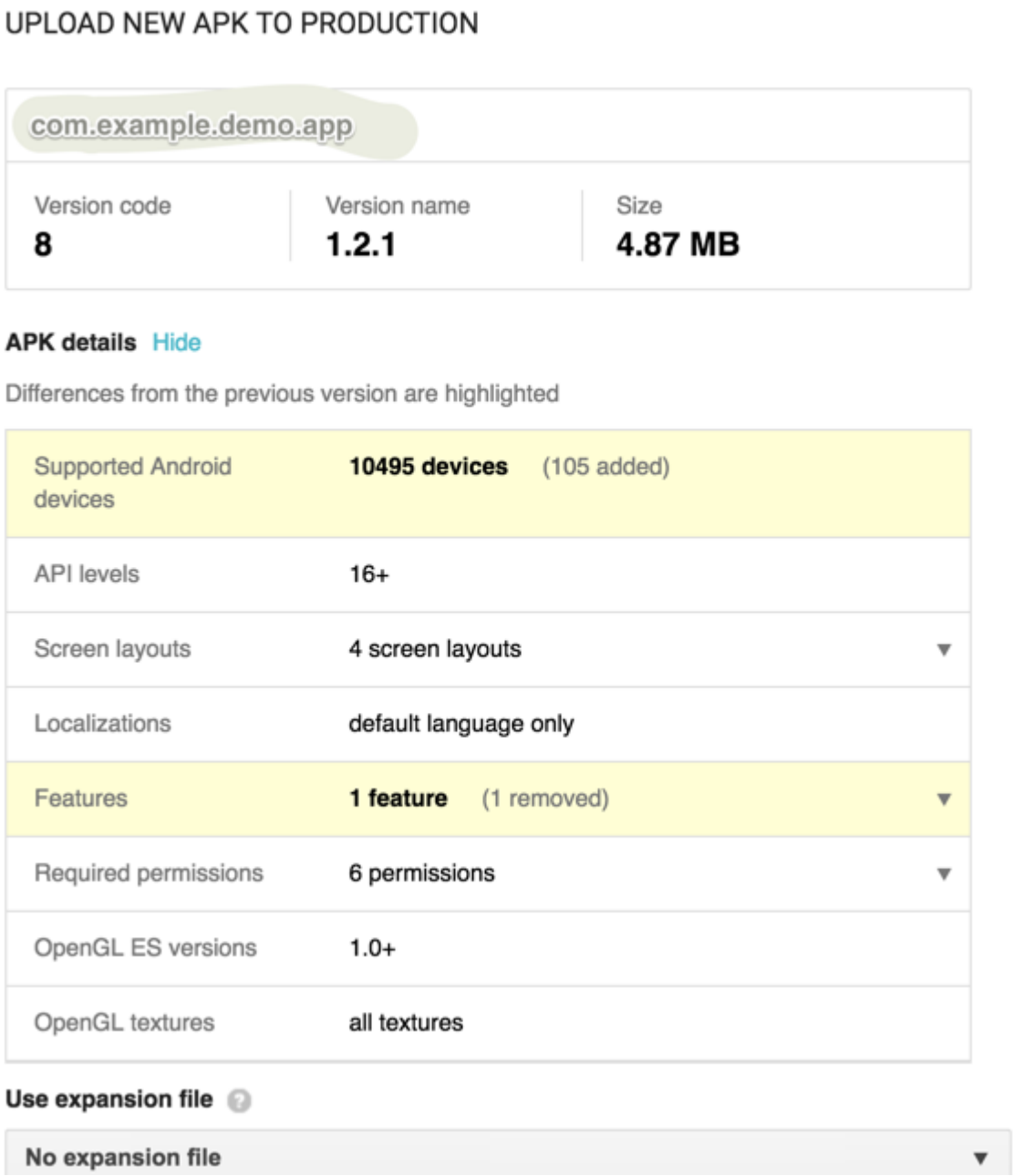
Минимальное руководство по представлению приложений

Требования:

- Счет разработчика
- Apk, уже построенный и подписанный с не отладочным ключом
- Бесплатное приложение, которое не имеет биллинга в приложении
- нет облачных сообщений Firebase или игровых сервисов

1. Перейдите на страницу https://play.google.com/apps/publish/.
   1a) Создайте свою учетную запись разработчика, если у вас ее нет.
2. Нажмите кнопку « `Create new Application`
3. Нажмите кнопку APK
4. Заполните все обязательные поля в форме, включая некоторые активы, которые будут отображаться в Play Маркете (см. Рисунок ниже)
5. Когда вы удовлетворены, нажмите кнопку « `Publish app`





Подробнее о входе в Настройка параметров подписи

Прочитайте Опубликовать в Play Store онлайн: https://riptutorial.com/ru/android/topic/5369/опубликовать-в-play-store

глава 193: Отображение портов с использованием библиотеки Cling на Android

Examples

Добавление поддержки Cling в ваш Android-проект

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'

}
```

Отображение порта 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;
    }
```

Прочитайте Отображение портов с использованием библиотеки Cling на Android онлайн: https://riptutorial.com/ru/android/topic/6208/отображение-портов-с-использованием-библиотеки-cling-на-android

глава 194: Переходы общего элемента

Вступление

Здесь вы найдете примеры перехода между `Activities` или `Fragments` с использованием общего элемента. Примером такого поведения является приложение Google Play Store, которое переводит значок приложения из списка в представление деталей приложения.

Синтаксис

- transaction.addSharedElement (sharedElementView, "targetTransitionName");
- fragment.setSharedElementEnterTransition (новый CustomTransaction ());

Examples

Переход между двумя фрагментами

В этом примере один из двух разных `ImageViews` должен быть переведен из `ChooserFragment` в `DetailFragment` .

В макете `ChooserFragment` нам нужны уникальные атрибуты `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" />
```

В классе `ChooserFragments` нам нужно передать `View` который был нажат, и идентификатор родительского `Activity` которое обрабатывает замену фрагментов (нам нужен идентификатор, чтобы узнать, какой ресурс изображения будет отображаться в `DetailFragment`). Как подробно передавать информацию в родительскую деятельность, безусловно, рассматривается в другой документации.

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

В `DetailFragment` для `ImageView` общего элемента также нужен уникальный атрибут `transitionName` .

```
<ImageView
    android:id="@+id/image_shared"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:transitionName="sharedImage" />
```

В `onCreateView() DetailFragment` мы должны решить, какой ресурс изображения должен быть показан (если мы этого не сделаем, общий элемент исчезнет после перехода).

```
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;
}
```

Родительская `Activity` получает обратные вызовы и обрабатывает замену фрагментов.

```
@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();
}
```

Не забыть - сам `Transition` . Этот пример перемещает и масштабирует общий элемент.

```
@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class DetailsTransition extends TransitionSet {

    public DetailsTransition() {
        setOrdering(ORDERING_TOGETHER);
        addTransition(new ChangeBounds()).
            addTransition(new ChangeTransform()).
            addTransition(new ChangeImageTransform());
    }

}
```

Прочитайте Переходы общего элемента онлайн: https://riptutorial.com/ru/android/topic/8933/переходы-общего-элемента

глава 195: Пикассо

Вступление

Picasso - это библиотека изображений для Android. Он создан и поддерживается Square . Это упрощает процесс отображения изображений из внешних местоположений. Библиотека обрабатывает каждый этап процесса, начиная с исходного HTTP-запроса и заканчивая кэшированием изображения. Во многих случаях для реализации этой аккуратной библиотеки требуется всего несколько строк кода.

замечания

Picasso - это мощная библиотека для загрузки и кэширования изображений для Android. Следуйте этому примеру, чтобы добавить библиотеку в свой проект.

Веб-сайты:

- Источник
- доктор
- Журнал изменений

Examples

Добавление библиотеки Picasso в ваш Android-проект

Из официальной документации :

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>
```

Захват и обработка ошибок

Picasso поддерживает как загрузочные, так и заполнитель ошибок в качестве дополнительных функций. Он также обеспечивает обратные вызовы для обработки результата загрузки.

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

Запрос будет повторен три раза, пока не появится место для замещения ошибки.

Повторная калибровка и поворот

```
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);
```

Круговые аватары с Пикассо

Вот пример класса Picasso Circle Transform на основе оригинала , с добавлением тонкой границы, а также включает функциональность для дополнительного разделителя для стекирования:

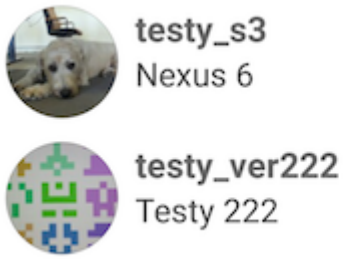
```
import android.graphics.Bitmap;
import android.graphics.BitmapShader;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Style;

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";
    }
}
```

Вот как использовать его при загрузке изображения (при условии, что `this` контекст активности, а `url` - строка с URL-адресом загружаемого изображения):

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform())
    .into(ivAvatar);
```

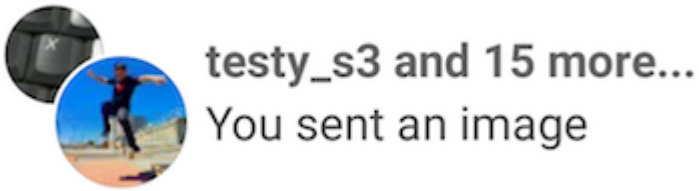
Результат:



Для использования с разделителем `true` конструктор для верхнего изображения:

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform(true))
    .into(ivAvatar);
```

Результат (два ImageViews в FrameLayout):



Отключить кеш в Picasso

```
Picasso.with(context)
.load(uri)
.networkPolicy(NetworkPolicy.NO_CACHE)
.memoryPolicy(MemoryPolicy.NO_CACHE)
.placeholder(R.drawable.placeholder)
.into(imageView);
```

Загрузка изображения из внешнего хранилища

```
String filename = "image.png";
String imagePath = getExternalFilesDir() + "/" + filename;

Picasso.with(context)
    .load(new File(imagePath))
    .into(imageView);
```

Загрузка изображения в виде растрового изображения с использованием Picasso

Если вы хотите загрузить изображение в виде `Bitmap` с помощью `Picasso` следующий код поможет вам:

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

Отмена запросов изображения с использованием Picasso

В некоторых случаях нам нужно отменить загрузку изображения в Picasso до завершения загрузки.

Это может произойти по разным причинам, например, если родительское представление перешло к другому представлению до того, как загрузка изображения может быть завершена.

В этом случае вы можете отменить запрос загрузки изображения с помощью `cancelRequest()` :

```
ImageView imageView;

//......

Picasso.with(imageView.getContext()).cancelRequest(imageView);
```

Использование Picasso в качестве ImageGetter для Html.fromHtml

Использование Picasso в качестве ImageGetter для 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);
    }

    //------------------------------------------------------------------//

}
}
```

Использование прост:

```
Html.fromHtml(textToParse, new PicassoImageGetter(picasso, textViewTarget), null);
```

Сначала попробуйте автономный кеш диска, затем перейдите в Интернет и выберите изображение

сначала добавьте OkHttp в файл сборки градиента модуля приложения

```
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'
```

Затем сделайте расширение класса

```
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);

    }
```

```
}
```

добавьте его в файл манифеста следующим образом:

```
<application
        android:name=".Global"
        .. >

</application>
```

Нормальное использование

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

Ссылка на оригинальный ответ

Прочитайте Пикассо онлайн: https://riptutorial.com/ru/android/topic/2172/пикассо

глава 196: Планирование работы

замечания

Остерегайтесь запускать много кода или выполнять тяжелую работу внутри вашего `JobService` , например, в `onStartJob()` . Код будет работать в потоке **основного** / **пользовательского интерфейса** и, следовательно, может привести к блокированному пользовательскому интерфейсу, более не реагирующему приложению или даже краху вашего приложения!

Из-за этого вы должны разгружать работу, например, используя `Thread` или `AsyncTask` .

Examples

Основное использование

Создать новый JobService

Это делается путем расширения класса `JobService` и реализации / переопределения необходимых методов `onStartJob()` и `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;
    }
}
```

Добавьте новый JobService в свой

AndroidManifest.xml

Следующий шаг является *обязательным* , иначе вы не сможете выполнять свою работу:

`MyJobService` свой класс `MyJobService` как новый элемент `<service>` между `<application>` `</application>` в вашем *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>
```

Настройка и запуск задания

После того, как вы внедрили новый JobService и добавили его в свой *AndroidManifest.xml* , вы можете продолжить выполнение последних шагов.

- `onButtonClick_startJob()` готовит и запускает периодическую работу. Помимо периодических заданий `JobInfo.Builder` позволяет указать многие другие настройки и ограничения. Например, вы можете определить, что для запуска задания требуется *подключенное зарядное устройство* или *сетевое соединение* .
- `onButtonClick_stopJob()` отменяет все текущие задания

```
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
    }
}
```

После вызова `onButtonClick_startJob()` задание будет выполняться с интервалом в 10 секунд, даже если приложение находится в состоянии *паузы* (кнопка нажата на кнопку дома и приложение больше не отображается).

Вместо отмены всех выполняемых заданий внутри `onButtonClick_stopJob()` вы также можете вызвать `jobScheduler.cancel()` чтобы отменить определенное задание на основе его идентификатора задания.

Прочитайте Планирование работы онлайн: https://riptutorial.com/ru/android/topic/6907/планирование-работы

глава 197: погрузчик

Вступление

Loader - хороший выбор для предотвращения утечки памяти, если вы хотите загрузить данные в фоновом режиме при вызове метода oncreate. Например, когда мы выполняем Asynctask в методе oncreate, и мы поворачиваем экран так, чтобы активность заново воссоздала, что снова запустит еще одну AsyncTask, так что, вероятно, две параллельные Asyntask работают параллельно, а не как загрузчик, которые будут продолжать фоновый процесс, который мы выполнили раньше.

параметры

Учебный класс	Описание
LoaderManager	Абстрактный класс, связанный с Activity или Fragment для управления одним или несколькими экземплярами Loader.
LoaderManager.LoaderCallbacks	Интерфейс обратного вызова для взаимодействия клиента с LoaderManager.
погрузчик	Абстрактный класс, который выполняет асинхронную загрузку данных.
AsyncTaskLoader	Абстрактный загрузчик, который предоставляет AsyncTask для выполнения работы.
CursorLoader	Подкласс AsyncTaskLoader, который запрашивает ContentResolver и возвращает курсор.

замечания

Представленные в Android 3.0 загрузчики упрощают асинхронную загрузку данных в виде активности или фрагмента. Погрузчики имеют следующие характеристики:

- Они доступны для каждой деятельности и фрагмента .
- Они обеспечивают асинхронную загрузку данных.
- Они отслеживают источник своих данных и приносят новые результаты при изменении содержимого.
- Они автоматически подключаются к курсору последнего загрузчика при воссоздании

после изменения конфигурации. Таким образом, им не нужно повторно запрашивать свои данные.

Когда не использовать Loaders

Вы не должны использовать Loaders, если вам нужны фоновые задачи для завершения. Android уничтожает Loaders вместе с действиями / фрагментами, к которым они принадлежат. Если вы хотите выполнить некоторые задачи, которые должны выполняться до завершения, не используйте Loaders. Вместо этого вы должны использовать сервисы для такого рода вещей.

Examples

Основной AsyncTaskLoader

`AsyncTaskLoader` - абстрактный `Loader` который предоставляет `AsyncTask` для выполнения этой работы.

Вот некоторые основные реализации:

```
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();
    }
}
```

Обычно `Loader` инициализируется в `onCreate()` активности или внутри `onActivityCreated()` . Также обычно активность или фрагмент реализует интерфейс `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) {
    }
}
```

В этом примере, когда загрузчик завершен, будет показан тост с результатом.

AsyncTaskLoader с кешем

Хорошая практика кэширования загруженного результата, чтобы избежать многократной загрузки одних и тех же данных.
Чтобы `onContentChanged()` недействительность кеша `onContentChanged()` необходимо вызвать. Если загрузчик уже запущен, `forceLoad()` , в противном случае (если загрузчик в состоянии остановки) загрузчик сможет понять изменение содержимого с `takeContentChanged()` проверки `takeContentChanged()` .

Примечание: `onContentChanged()` необходимо вызвать из основного потока процесса.

Javadocs говорит о takeContentChanged ():

> Возьмите текущий флаг, указывающий, изменилось ли содержимое загрузчика во время его остановки. Если это так, возвращается true и флаг очищается.

```
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
    }
}
```

перезарядка

Чтобы аннулировать старые данные и перезапустить существующий загрузчик, вы можете использовать `restartLoader()` :

```
private void reload() {
    getLoaderManager().reastartLoader(LOADER_ID, Bundle.EMPTY, this);
}
```

Передача параметров с помощью Bundle

Вы можете передавать параметры через Bundle:

```
Bundle myBundle = new Bundle();
myBundle.putString(MY_KEY, myValue);
```

Получите значение в onCreateLoader:

```
@Override
public Loader<String> onCreateLoader(int id, final Bundle args) {
    final String myParam = args.getString(MY_KEY);
    ...
}
```

Прочитайте погрузчик онлайн: https://riptutorial.com/ru/android/topic/4390/погрузчик

глава 198: Поддержка экранов с различными разрешениями, размерами

замечания

Условия и понятия

Размер экрана

Фактический физический размер, измеренный как диагональ экрана. Для простоты Android группирует все фактические размеры экрана в четыре обобщенных размера: маленький, обычный, большой и очень большой.

Плотность экрана

Количество пикселей в физической области экрана; обычно называемый dpi (точек на дюйм). Например, экран с низкой плотностью имеет меньше пикселей в заданной физической области по сравнению с экраном с «нормальной» или «высокой» плотностью. Для простоты Android группирует все фактические плотности экрана в шесть обобщенных плотностей: низкое, среднее, высокое, сверхвысокое, сверх-экстра-высокое и сверх-экстра-экстра-высокое.

ориентация

Ориентация экрана с точки зрения пользователя. Это либо пейзаж, либо портрет, что означает, что соотношение сторон экрана является либо широким, либо высоким, соответственно. Помните, что по умолчанию разные устройства работают не только в разных ориентациях, но и во время работы, когда пользователь поворачивает устройство, ориентация может измениться. Разрешение Общее количество физических пикселей на экране. При добавлении поддержки нескольких экранов приложения не работают напрямую с разрешением; приложения должны касаться только размера экрана и плотности, как это определено обобщенными группами размеров и плотности. Независимый от плотности пиксель (dp) Элемент виртуального пикселя, который следует использовать при определении компоновки пользовательского интерфейса, чтобы выразить размеры или положение макета независимо от плотности. Независимый от плотности пиксель эквивалентен одному физическому пикселю на экране с разрешением 160 точек на дюйм, который представляет собой базовую плотность, принимаемую системой для экрана

средней плотности. Во время работы система прозрачно обрабатывает любое масштабирование блоков dp по мере необходимости, исходя из фактической плотности используемого экрана. Преобразование блоков dp в пиксели экрана просто: px = dp * (dpi / 160). Например, на экране с разрешением 240 точек на дюйм 1 дп равен 1,5 физическим пикселям. Вы всегда должны использовать модули dp при определении пользовательского интерфейса вашего приложения, чтобы обеспечить правильное отображение вашего интерфейса на экранах с различной плотностью.

Единицы

- **ПВ**

  Пиксели - соответствуют фактическим пикселям на экране.

- **в**

  Дюймы - на основе физического размера экрана. 1 дюйм = 2,54 сантиметра

- **мм**

  Миллиметры - в зависимости от физических размеров экрана.

- **пт**

  Очки - 1/72 дюйма в зависимости от физического размера экрана.

- **dp или dip**

  Плотно-независимые пиксели - абстрактный блок, основанный на физической плотности экрана. Эти единицы относятся к экрану с разрешением 160 точек на дюйм, поэтому один пиксель составляет один пиксель на экране с разрешением 160 точек на дюйм. Отношение dp-to-pixel будет меняться с плотностью экрана, но не обязательно в прямой пропорции. Примечание. Компилятор принимает как «dip», так и «dp», хотя «dp» более соответствует «sp».

- **зр**

  Масштабируемые пиксели - это похоже на блок dp, но он также

масштабируется по предпочтению размера шрифта пользователя. Рекомендуется использовать этот аппарат при задании размеров шрифта, поэтому они будут настроены как по плотности экрана, так и по предпочтениям пользователя. От понимания независимости плотности в Android:

Единица измерения	Описание	Единицы на физический дюйм	Независимость от плотности	Один и тот же физический размер на каждом экране
ПВ	Пиксели	Различная	нет	нет
в	дюймов	1	да	да
мм	миллиметры	+25,4	да	да
пт	Точки	72	да	да
дп	Плотность независимых пикселей	~ 160	да	нет
зр	Масштабировать независимые пиксели	~ 160	да	нет

Рекомендации:

- https://developer.android.com/guide/practices/screens_support.html
- http://developer.android.com/guide/topics/resources/more-resources.html

Examples

Использование спецификаторов конфигурации

Android поддерживает несколько спецификаторов конфигурации, которые позволяют вам контролировать, как система выбирает ваши альтернативные ресурсы на основе характеристик текущего экрана устройства. Спецификатор конфигурации - это строка, которую вы можете добавить в каталог ресурсов в своем проекте Android и указать конфигурацию, для которой сконструированы ресурсы внутри.

Чтобы использовать квалификатор конфигурации:

1. Создайте новый каталог в директории res / проекта и назовите его в формате:

`<resources_name>-<qualifier>` . `<resources_name>` - это стандартное имя ресурса (например, drawable или layout).

2. `<qualifier>` - это спецификатор конфигурации, определяющий конфигурацию экрана, для которой эти ресурсы должны использоваться (например, hdpi или xlarge).

Например, следующие каталоги ресурсов приложений предоставляют различные схемы компоновки для разных размеров экрана и различных чертежей. Используйте `mipmap/` folders для значков запуска.

```
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
```

Преобразование dp и sp в пиксели

Когда вам нужно установить значение пикселя для чего-то вроде `Paint.setTextSize` но все же хотите, чтобы оно масштабировалось на основе устройства, вы можете преобразовать значения dp и 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);
```

Кроме того, вы можете преобразовать ресурс измерения в пиксели, если у вас есть контекст для загрузки ресурса.

```
<?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);
```

Размер текста и различные размеры экрана для Android

Иногда лучше иметь только три варианта

```
 style="@android:style/TextAppearance.Small"
 style="@android:style/TextAppearance.Medium"
 style="@android:style/TextAppearance.Large"
```

Используйте маленькие и большие, чтобы отличать нормальный размер экрана.

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="@android:style/TextAppearance.Small"/>
```

Для нормального вам ничего не нужно указывать.

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

Используя это, вы можете избежать тестирования и указания размеров для разных размеров экрана.

Прочитайте Поддержка экранов с различными разрешениями, размерами онлайн: https://riptutorial.com/ru/android/topic/1086/поддержка-экранов-с-различными-разрешениями--размерами

глава 199: Подключение Wi-Fi

Examples

Подключение к WEP-шифрованию

В этом примере подключается точка доступа Wi-Fi с использованием WEP-шифрования с учетом SSID и пароля.

```
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;
    }
}
```

Подключение к шифрованию WPA2

В этом примере подключается точка доступа Wi-Fi с использованием шифрования 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;
  }
}
```

Сканирование точек доступа

В этом примере просматриваются доступные точки доступа и сети ad hoc. `btnScan` активирует сканирование, инициированное с помощью `WifiManager.startScan()` . После сканирования `WifiManager` вызывает `SCAN_RESULTS_AVAILABLE_ACTION` намерение и `WifiScanReceiver` класс обрабатывает результат сканирования. Результаты отображаются в `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");
            }
        }
    }
}
```

права доступа

В *AndroidManifest.xml* должны быть определены следующие разрешения:

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

`android.permission.ACCESS_WIFI_STATE` необходимо для вызова `WifiManager.getScanResults()` . Без `android.permission.CHANGE_WIFI_STATE` вы не можете инициировать сканирование с помощью `WifiManager.startScan()` .

При компиляции проекта для уровня api 23 или выше (Android 6.0 и выше) необходимо установить либо `android.permission.ACCESS_FINE_LOCATION` либо `android.permission.ACCESS_COARSE_LOCATION` . Кроме того, необходимо получить разрешение, например, в методе `onCreate` вашего основного вида деятельности:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    String[] PERMS_INITIAL={
            Manifest.permission.ACCESS_FINE_LOCATION,
    };
    ActivityCompat.requestPermissions(this, PERMS_INITIAL, 127);
}
```

Прочитайте Подключение Wi-Fi онлайн: https://riptutorial.com/ru/android/topic/3288/подключение-wi-fi

глава 200: Подпишите свое приложение для Android

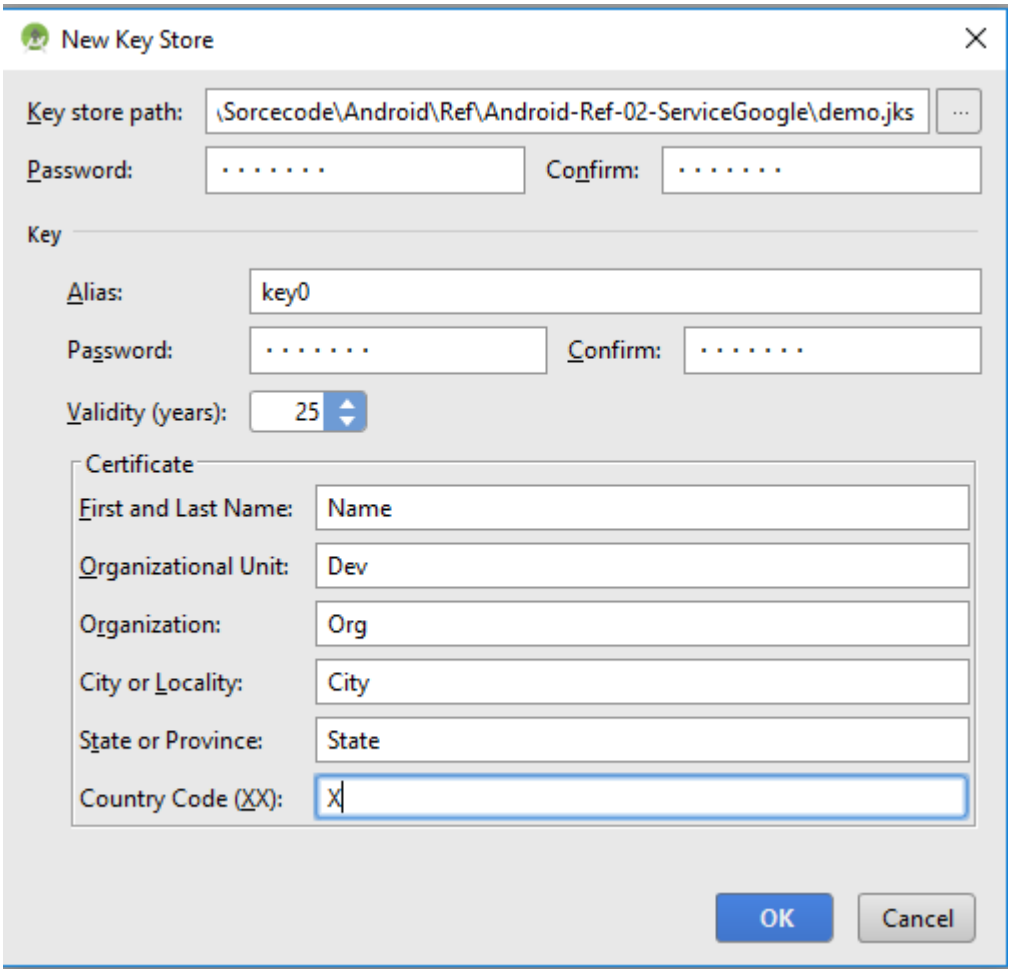
Вступление

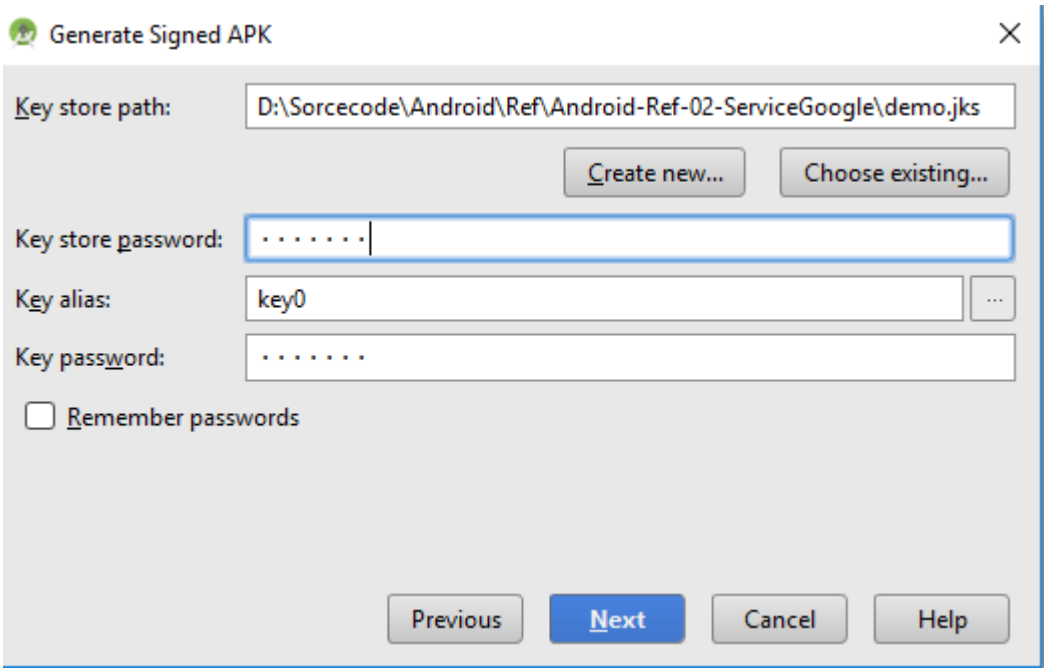
Android требует, чтобы все APK были подписаны для выпуска.

Examples

Подпишите свое приложение

1. В строке меню нажмите «Создать»> «Создать подписанный APK».
2. Выберите модуль, который вы хотите выпустить, и нажмите «Далее».
3. Чтобы создать новое хранилище ключей, нажмите «Создать новый». Теперь заполните необходимую информацию и нажмите ok в New Key Store.





4. В полях Generate Signed APK Wizard уже заполнены поля, если вы только что создали новый хранилище ключей, иначе заполните его и нажмите «Далее».

5. В следующем окне выберите пункт назначения для подписанного APK, выберите тип сборки и нажмите «Готово».

Настроить build.gradle с настройкой подписи

Вы можете определить конфигурацию подписи, чтобы подписать apk в файле `build.gradle` .

Вы можете определить:

- `storeFile` : файл хранилища ключей
- `storePassword` : пароль хранилища ключей
- `keyAlias` : ключевое имя псевдонима
- `keyPassword` : пароль псевдонима ключа

Вы должны **определить** блок `signingConfigs` для создания конфигурации подписи:

```
android {
    signingConfigs {

        myConfig {
            storeFile file("myFile.keystore")
            storePassword "xxxx"
            keyAlias "xxxx"
            keyPassword "xxxx"
        }
    }
    //....
}
```

Затем вы можете **назначить** его одному или нескольким типам сборки.

```
android {

    buildTypes {
        release {
            signingConfig signingConfigs.myConfig
        }
    }
}
```

Прочитайте Подпишите свое приложение для Android онлайн:
https://riptutorial.com/ru/android/topic/9721/подпишите-свое-приложение-для-android

глава 201: Пожарная авария

Examples

Как добавить отчет о сбоях Firebase в приложение

Чтобы добавить *Firebase Crash Reporting* в ваше приложение, выполните следующие действия:

- Создайте приложение в *консоли Firebase* здесь .
- Скопируйте файл `google-services.json` из вашего проекта в свой каталог `app/` .
- Добавьте следующие правила в файл *build.gradle на* уровне *root* , чтобы включить плагин `google-services` :

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.0.0'
    }
}
```

- В своем модульном файле Gradle добавьте строку `apply plugin` в нижней части файла, чтобы включить плагин Gradle:

```
apply plugin: 'com.google.gms.google-services'
```

- Добавьте зависимость для *Crash Reporting* в файл *build.gradle на* уровне *приложения* :

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

- Затем вы можете запустить специальное исключение из своего приложения, используя следующую строку:

```
FirebaseCrash.report(new Exception("Non Fatal Error logging"));
```

  Все ваши фатальные исключения будут отправлены на вашу *Firebase Console* .

- Если вы хотите добавить пользовательские журналы в консоль, вы можете использовать следующий код:

```
FirebaseCrash.log("Level 2 completed.");
```

Для получения дополнительной информации, пожалуйста, посетите:

- Официальная документация
- Тема переполнения стека

Как сообщить об ошибке

Firebase Crash Reporting автоматически генерирует отчеты о фатальных ошибках (или исключенных исключениях).

Вы можете создать свой собственный отчет, используя:

```
FirebaseCrash.report(new Exception("My first Android non-fatal error"));
```

Вы можете проверить журнал, когда FirebaseCrash инициализировал модуль:

> 07-20 08: 57: 24.442 D / FirebaseCrashApiImpl: **API отчетов FirebaseCrash инициализирован** 07-20 08: 57: 24.442 I / FirebaseCrash: **отчет FirebaseCrash инициализирует** d com.google.firebase.crash.internal.zzg@3333d325 07-20 08: 57: 24.442 D / FirebaseApp: **инициализирован класс com.google.firebase.crash.FirebaseCrash.**

И затем, когда он отправил исключение:

> 07-20 08: 57: 47.052 D / FirebaseCrashApiImpl: **throwable java.lang.Exception: Моя первая нефатальная ошибка Android** 07-20 08: 58: 18.822 D / FirebaseCrashSenderServiceImpl: **Код ответа: 200** 07-20 08: 58: 18.822 D / FirebaseCrashSenderServiceImpl: **отправлено сообщение**

Вы можете добавить собственные отчеты в свой отчет с помощью

```
FirebaseCrash.log("Activity created");
```

Прочитайте Пожарная авария онлайн: https://riptutorial.com/ru/android/topic/5965/пожарная-авария

глава 202: Пожарная безопасность Firebase

Вступление

Firebase Cloud Messaging (FCM) - это межплатформенное решение для обмена сообщениями, которое позволяет надежно доставлять сообщения без каких-либо затрат.

Используя FCM, вы можете уведомить клиентское приложение о том, что для синхронизации доступны новые электронные или другие данные. Вы можете отправлять уведомляющие сообщения для вовлечения пользователей и их удержания. Для таких случаев, как обмен мгновенными сообщениями, сообщение может передавать полезную нагрузку до `4KB` в клиентское приложение.

Examples

Настройка клиентского приложения CloudBase Firebase на Android

1. Заполните раздел « Установка и настройка», чтобы подключить ваше приложение к Firebase.
   Это создаст проект в Firebase.

2. Добавьте зависимость для облачных сообщений Firebase к файлу `build.gradle` уровне `build.gradle` :

```
dependencies {
     compile 'com.google.firebase:firebase-messaging:10.2.1'
}
```

Теперь вы готовы работать с FCM в Android.

Для клиентов FCM требуются устройства под управлением `Android 2.3` или более поздней версии, в которых также установлено приложение Google Play Store или эмулятор под управлением Android 2.3 с API Google.

Измените файл `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>
```

Регистрационный токен

При первом запуске вашего приложения FCM SDK генерирует регистрационный токен для экземпляра клиентского приложения.
Если вы хотите настроить таргетинг на отдельные устройства или создать группы устройств, вам необходимо получить доступ к этому токену, расширив `FirebaseInstanceIdService` .

`onTokenRefresh` вызов `onTokenRefresh` срабатывает всякий раз, когда генерируется новый токен, и вы можете использовать метод `FirebaseInstanceID.getToken()` для извлечения текущего токена.

Пример:

```
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);

    }

}
```

Этот код, который я применил в своем приложении для нажатия изображения, сообщения, а также ссылки для открытия в вашем веб-браузере

Это мой сервис 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;

    }
}}
```

И это MainActivity, чтобы открывать ссылку в моем WebView или другом браузере на ваше требование по намерениям.

```
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();
    }}
```

Получать сообщения

Чтобы получать сообщения, используйте службу, которая расширяет `FirebaseMessagingService` и переопределяет метод `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());
        }

        //.....
    }
```

Когда приложение находится в фоновом режиме, Android направляет уведомления на системный трей. Пользователь, нажав на уведомление, открывает по умолчанию программу запуска приложений.

Сюда входят сообщения, содержащие как полезную нагрузку для уведомлений, так и данные (и все сообщения, отправленные с консоли уведомлений). В этих случаях уведомление доставляется на системный лоток устройства, а полезная нагрузка данных доставляется в дополнение к назначению вашей активности запуска.

Вот краткое описание:

Состояние приложения	уведомление	Данные	И то и другое
передний план	`onMessageReceived`	`onMessageReceived`	`onMessageReceived`

Состояние приложения	уведомление	Данные	И то и другое
Фон	Системный лоток	`onMessageReceived`	Уведомление: системный трей
			Данные: в дополнение к намерениям.

Подписаться на тему

Приложения для клиентов могут подписаться на любую существующую тему или создать новую тему. Когда клиентское приложение подписывается на новое имя темы, в FCM создается новый раздел этого имени, и любой клиент может впоследствии подписаться на него.

Чтобы подписаться на тему, используйте метод `subscribeToTopic()` определяющий имя темы:

```
FirebaseMessaging.getInstance().subscribeToTopic("myTopic");
```

Прочитайте Пожарная безопасность Firebase онлайн:
https://riptutorial.com/ru/android/topic/8826/пожарная-безопасность-firebase

глава 203: Показатели отображения устройства

Examples

Получить размеры пикселей экрана

Чтобы увеличить ширину и высоту экранов в пикселях, мы можем использовать показатели отображения WindowManagers .

```
// Get display metrics
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
```

Эти DisplayMetrics содержат серию информации о экране устройств, например, о ее плотности или размере:

```
// Get width and height in pixel
Integer heightPixels = metrics.heightPixels;
Integer widthPixels = metrics.widthPixels;
```

Получить плотность экрана

Чтобы получить плотность экранов, мы также можем использовать DisplayMetrics Windowmanagers . Это быстрый пример:

```
// Get density in dpi
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
int densityInDpi =  metrics.densityDpi;
```

Формула px для dp, dp to px для разговора

DP для пикселя:

```
private int dpToPx(int dp)
{
    return (int) (dp * Resources.getSystem().getDisplayMetrics().density);
}
```

Pixel to DP:

```
private int pxToDp(int px)
{
    return (int) (px / Resources.getSystem().getDisplayMetrics().density);
```

```
}
```

Прочитайте Показатели отображения устройства онлайн:
https://riptutorial.com/ru/android/topic/4207/показатели-отображения-устройства

глава 204: Покрасить

Вступление

Краска - это один из четырех объектов, необходимых для рисования, а также холст (содержит призывы рисования), битмап (содержит пиксели) и примитив рисования (Rect, Path, Bitmap ...)

Examples

Создание краски

Вы можете создать новую краску с одним из этих трех конструкторов:

- `new Paint()` Создать с настройками по умолчанию
- `new Paint(int flags)` Создать с флагами
- `new Paint(Paint from)` Параметры копирования из другой краски

Обычно рекомендуется никогда не создавать объект рисования или любой другой объект в onDraw (), поскольку это может привести к проблемам с производительностью. (Android Studio, вероятно, предупредит вас) Вместо этого сделайте его глобальным и инициализируйте его в своем конструкторе классов следующим образом:

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

Настройка Paint для текста

Настройки текстового чертежа

- `setTypeface(Typeface typeface)` Установите шрифт. См. Шрифт
- `setTextSize(int size)`

Установите размер шрифта в пикселях.

- `setColor(int color)` Установите цвет рисования рисунка, включая цвет текста. Вы также можете использовать `setARGB(int a, int r, int g, int b` и `setAlpha(int alpha)`
- `setLetterSpacing(float size)` Установите расстояние между символами в ems. Значение по умолчанию равно 0, отрицательное значение будет затягивать текст, а положительный - расширяет его.
- `setTextAlign(Paint.Align align)` Установите выравнивание текста относительно его начала. `Paint.Align.LEFT` будет рисовать его справа от начала координат, `RIGHT` будет рисовать его влево, а `CENTER` будет рисовать его по центру (по горизонтали)
- `setTextSkewX(float skewX)` Это можно считать поддельным курсивом. SkewX представляет горизонтальное смещение нижней части текста. (используйте -0.25 для курсива)
- `setStyle(Paint.Style style)` Наполните текст `FILL` , Stroke текст `STROKE` , или оба `FILL_AND_STROKE`

Обратите внимание, что вы можете использовать `TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, size, getResources().getDisplayMetrics())` для преобразования из SP или DP в пиксели.

Измерительный текст

- `float width = paint.measureText(String text)` Измерьте ширину текста
- `float height = paint.ascent()` Измерение высоты текста
- `paint.getTextBounds(String text, int start, int end, Rect bounds` Сохраняет размеры текста. Вы назначили Rect, оно не может быть null:

```
String text = "Hello world!";
Rect bounds = new Rect();
paint.getTextBounds(text, 0, text.length(), bounds);
```

Существуют и другие методы измерения, однако эти три должны соответствовать большинству целей.

Настройка Paint для рисования фигур

- `setStyle(Paint.Style style)` Заполненная форма `FILL` , форма Stroke `STROKE` , или оба `FILL_AND_STROKE`
- `setColor(int color)` Установите цвет рисования рисунка. Вы также можете использовать `setARGB(int a, int r, int g, int b` и `setAlpha(int alpha)`
- `setStrokeCap(Paint.Cap cap)` Установите ограничения на линию, либо `ROUND` , `SQUARE` , либо `BUTT` (нет). См. это .
- `setStrokeJoin(Paint.Join join)` Установите линии, соединяющие `MITER` (pointy), `ROUND` или `BEVEL` . Смотрите это .
- `setStrokeMiter(float miter)` Установите ограничение на объединение митра. Это может препятствовать тому, чтобы объединение титров продолжалось неопределенно,

превратив его в скошенное соединение после x пикселей. Смотрите это .

- `setStrokeWidth(float width)` Установите ширину хода. `0` будет рисовать в режиме волоса, независимо от матрицы холста. (всегда 1 пиксель)

Установка флагов

Вы можете установить следующие флаги в конструкторе или с помощью `setFlags(int flags)`

- `Paint.ANTI_ALIAS_FLAG` Включить сглаживание, `Paint.ANTI_ALIAS_FLAG` рисунок.
- `Paint.DITHER_FLAG` Включить сглаживание. Если точность цвета выше, чем у устройства, это произойдет .
- `Paint.EMBEDDED_BITMAP_TEXT_FLAG` Позволяет использовать растровые шрифты.
- `Paint.FAKE_BOLD_TEXT_FLAG` будет рисовать текст с поддельным жирным эффектом, который можно использовать вместо жирного шрифта. Некоторые шрифты имеют жирный шрифт, поддельные смелые не будут
- `Paint.FILTER_BITMAP_FLAG` Влияет на выборку растровых изображений при преобразовании.
- `Paint.HINTING_OFF` , `Paint.HINTING_ON` Переключает `Paint.HINTING_ON` шрифтов, см. Это
- `Paint.LINEAR_TEXT_FLAG` Отключает масштабирование шрифтов, вместо этого выполняются операции рисования
- `Paint.SUBPIXEL_TEXT_FLAG` Текст будет вычисляться с использованием субпиксельной точности.
- `Paint.STRIKE_THRU_TEXT_FLAG` Текст будет нарисован
- `Paint.UNDERLINE_TEXT_FLAG` Текст будет `Paint.UNDERLINE_TEXT_FLAG`

Вы можете добавить флаг и удалить флаги следующим образом:

```
Paint paint = new Paint();
paint.setFlags(paint.getFlags() | Paint.FLAG);   // Add flag
paint.setFlags(paint.getFlags() & ~Paint.FLAG);  // Remove flag
```

Пытаться удалить флаг, которого нет или добавить флаг, который уже там, ничего не изменит. Также обратите внимание, что большинство флагов также можно установить с помощью `set<Flag>(boolean enabled)` , например `setAntialias(true)` .

Вы можете использовать `paint.reset()` для сброса краски по умолчанию. Единственный флаг по умолчанию - `EMBEDDED_BITMAP_TEXT_FLAG` . Он будет установлен, даже если вы используете `new Paint(0)` , у вас будет

Прочитайте Покрасить онлайн: https://riptutorial.com/ru/android/topic/9141/покрасить

глава 205: Получение имен системных шрифтов и использование шрифтов

Вступление

В следующих примерах показано, как получить имена по умолчанию системных шрифтов, хранящихся в каталоге */ system / fonts /,* и как использовать системный шрифт для установки шрифта элемента `TextView` .

Examples

Получение имен системных шрифтов

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

Применение системного шрифта к TextView

В следующем коде вам нужно заменить имя `fontsname` на имя шрифта, который вы хотите использовать:

```
TextView lblexample = (TextView) findViewById(R.id.lblexample);
lblexample.setTypeface(Typeface.createFromFile("/system/fonts/" + "fontsname" + ".ttf"));
```

Прочитайте Получение имен системных шрифтов и использование шрифтов онлайн: https://riptutorial.com/ru/android/topic/10930/получение-имен-системных-шрифтов-и-использование-шрифтов

глава 206: Пользовательские шрифты

Examples

Внесение пользовательского шрифта в ваше приложение

1. Перейдите в папку (папку проекта)
2. Затем app -> src -> main.
3. Создать папку 'assets -> fonts' в основной папке.
4. Поместите ваш 'fontfile.ttf' в папку шрифтов.

Инициализация шрифта

```
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");
}
```

Использование пользовательского шрифта в TextView

```
public void setFont(TextView textView) {
    textView.setTypeface(myFont);
}
```

Применить шрифт в TextView по xml (не требуется код Java)

TextViewPlus.java:

```
public class TextViewPlus extends TextView {
    private static final String TAG = "TextView";

    public TextViewPlus(Context context) {
        super(context);
    }

    public TextViewPlus(Context context, AttributeSet attrs) {
        super(context, attrs);
        setCustomFont(context, attrs);
    }

    public TextViewPlus(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        setCustomFont(context, attrs);
    }
```

```
    private void setCustomFont(Context ctx, AttributeSet attrs) {
        TypedArray a = ctx.obtainStyledAttributes(attrs, R.styleable.TextViewPlus);
        String customFont = a.getString(R.styleable.TextViewPlus_customFont);
        setCustomFont(ctx, customFont);
        a.recycle();
    }

    public boolean setCustomFont(Context ctx, String asset) {
        Typeface typeface = null;
        try {
            typeface = Typeface.createFromAsset(ctx.getAssets(), asset);
        } catch (Exception e) {
            Log.e(TAG, "Unable to load typeface: "+e.getMessage());
            return false;
        }

        setTypeface(typeface);
        return true;
    }
}
```

attrs.xml: (Где разместить **res / values**)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="TextViewPlus">
        <attr name="customFont" format="string"/>
    </declare-styleable>
</resources>
```

Как пользоваться:

```
<?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>
```

Пользовательский шрифт в тексте холста

Рисование текста в холсте с вашим шрифтом из активов.

```
Typeface typeface = Typeface.createFromAsset(getAssets(), "fonts/SomeFont.ttf");
Paint textPaint = new Paint();
textPaint.setTypeface(typeface);
canvas.drawText("Your text here", x, y, textPaint);
```

Эффективная загрузка поверхностного слоя

Загрузка пользовательских шрифтов может привести к плохой производительности. Я настоятельно рекомендую использовать этот маленький помощник, который сохраняет / загружает ваши уже используемые шрифты в 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;
}
```

}

Использование:

```
Typeface typeface = TypefaceUtils.getTypeface(context, "RobotoSlab-Bold.ttf");
setTypeface(typeface);
```

Пользовательский шрифт для всей деятельности

```
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();
    }
}
```

Затем в вашей деятельности, в `onCreate()` :

```
// Put your font to assets folder...

ReplaceFont.changeDefaultFont(getApplication(), "DEFAULT", "LinLibertine.ttf");
```

Работа с шрифтами в Android O

Android O изменяет способ работы со шрифтами.

Android O представляет новую функцию, называемую *Fonts in XML* , которая позволяет использовать шрифты в качестве ресурсов. Это означает, что нет необходимости связывать шрифты как активы. Шрифты теперь скомпилированы в *R*- файле и автоматически доступны в системе как ресурс.

Чтобы добавить новый **шрифт** , вам необходимо сделать следующее:

- Создайте новый каталог ресурсов: `res/font` .
- Добавьте файлы шрифтов в эту папку шрифтов. Например, добавив `myfont.ttf` , вы сможете использовать этот шрифт через `R.font.myfont` .

Вы также можете создать собственное **семейство шрифтов** , добавив следующий файл XML в каталог `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>
```

Вы можете использовать как файл **шрифта, так и** файл семейства **шрифтов** таким же образом:

- **В XML-файле,** используя атрибут `android:fontFamily` , например, вот так:

  ```
  <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:fontFamily="@font/myfont"/>
  ```

  Или вот так:

  ```
  <style name="customfontstyle" parent="@android:style/TextAppearance.Small">
      <item name="android:fontFamily">@font/myfont</item>
  ```

```
</style>
```

- **В вашем коде** , используя следующие строки кода:

```
Typeface typeface = getResources().getFont(R.font.myfont);
textView.setTypeface(typeface);
```

Прочитайте Пользовательские шрифты онлайн: https://riptutorial.com/ru/android/topic/3358/пользовательские-шрифты

глава 207: Посмотреть список

Вступление

ListView - это группа представлений, которая группирует несколько элементов из источника данных, например массива или базы данных, и отображает их в списке прокрутки. Данные привязаны к списку с использованием класса Adapter.

замечания

`ListView` - это группа представлений, которая отображает список прокручиваемых элементов.
Элементы списка автоматически вставляются в список с помощью `Adapter` который извлекает контент из источника, такого как массив или запрос базы данных, и преобразует результат каждого элемента в представление, которое помещено в список.

Когда содержимое вашего макета будет динамическим или не определено заранее, вы можете использовать макет, который подклассифицирует `AdapterView` для заполнения макета с помощью представлений во время выполнения. В подклассе класса `AdapterView` используется `Adapter` для привязки данных к его компоновке.

Перед использованием `ListView` вы также должны проверить примеры `RecyclerView` .

Examples

Фильтрация с помощью 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);
     }
});
```

Скажем, ваш запрос будет запускаться каждый раз, когда пользователь вводит в `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) {

        }
    });
```

Пользовательский массив ArrayAdapter

По умолчанию класс ArrayAdapter создает представление для каждого элемента массива, вызывая `toString()` для каждого элемента и помещая содержимое в TextView.

Чтобы создать сложный вид для каждого элемента (например, если вы хотите ImageView для каждого элемента массива), расширьте класс ArrayAdapter и переопределите метод `getView()` чтобы вернуть тип представления, который вы хотите для каждого элемента.

Например:

```
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);
         }
    }
}
```

Основной ListView с ArrayAdapter

По умолчанию `ArrayAdapter` создает представление для каждого элемента массива, вызывая `toString()` для каждого элемента и помещая содержимое в `TextView` .

Пример:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, myStringArray);
```

где `android.R.layout.simple_list_item_1` - это макет, содержащий `TextView` для каждой строки массива.

Затем просто вызовите `setAdapter()` в `ListView` :

```
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

Чтобы использовать нечто, отличное от TextView для отображения массива, например ImageViews, или для получения некоторых данных, кроме того, что результаты `toString()` заполняют представления, переопределяют `getView(int, View, ViewGroup)` чтобы вернуть `getView(int, View, ViewGroup)` тип представления. Проверьте этот пример .

Прочитайте Посмотреть список онлайн: https://riptutorial.com/ru/android/topic/4226/посмотреть-список

глава 208: Поставщик услуг

замечания

Поставщики контента управляют доступом к структурированному набору данных. Они инкапсулируют данные и предоставляют механизмы для определения безопасности данных. Поставщики контента - это стандартный интерфейс, который соединяет данные в одном процессе с кодом, запущенным в другом процессе.

Когда вы хотите получить доступ к данным в поставщике контента, вы используете объект `ContentResolver` в `Context` вашего приложения для связи с поставщиком как с клиентом. Объект `ContentResolver` связывается с объектом-провайдером, экземпляром класса, который реализует `ContentProvider` . Объект поставщика получает запросы данных от клиентов, выполняет запрошенное действие и возвращает результаты.

Вам не нужно разрабатывать собственный провайдер, если вы не собираетесь делиться своими данными с другими приложениями. Тем не менее, вам нужен собственный провайдер для предоставления пользовательских предложений по поиску в вашем собственном приложении. Вам также нужен собственный провайдер, если вы хотите скопировать и вставить сложные данные или файлы из вашего приложения в другие приложения.

Сам Android включает поставщиков контента, которые управляют данными, такими как аудио, видео, изображения и личные контактные данные. Вы можете увидеть некоторые из них, перечисленные в справочной документации для пакета `android.provider` . С некоторыми ограничениями эти поставщики доступны для любого приложения Android.

Examples

Внедрение базового класса поставщика контента

1) Создать класс контракта

Класс контракта определяет константы, которые помогают приложениям работать с URI содержимого, именами столбцов, действиями намерений и другими функциями поставщика контента. Классы договоров не включаются автоматически с поставщиком; разработчик провайдера должен определить их, а затем сделать их доступными для других разработчиков.

У провайдера обычно есть единый орган, который служит его внутренним именем Android. Чтобы избежать конфликтов с другими поставщиками, используйте уникальный контент. Поскольку эта рекомендация верна и для имен пакетов Android, вы можете определить полномочия своего поставщика как расширение имени пакета, содержащего поставщика.

Например, если ваше имя пакета Android является `com.example.appname` , вы должны предоставить вашему провайдеру полномочия `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";
}
```

URI контента - это URI, который идентифицирует данные в провайдере. Контексты содержимого включают символическое имя всего поставщика (его полномочия) и имя, которое указывает на таблицу или файл (путь). Дополнительная часть id указывает на отдельную строку в таблице. Каждый метод доступа к данным ContentProvider имеет URI контента в качестве аргумента; это позволяет определить таблицу, строку или файл для доступа. Определите их в классе контракта.

```
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) Создайте класс помощника

Класс-помощник управляет созданием базы данных и управлением версиями.

```
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) Создайте класс, расширяющий класс 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;
```

UriMatcher отображает полномочия и путь к целочисленному значению. Метод `match()` возвращает уникальное целочисленное значение для URI (это может быть любое произвольное число, если оно уникально). Оператор switch выбирает между запросом всей таблицы и запросом для одной записи. Наш UriMatcher возвращает 100, если URI является URI содержимого таблицы и 101, если URI указывает на определенную строку внутри этой таблицы. Вы можете использовать подстановочный символ `#` чтобы соответствовать любому числу и `*` чтобы соответствовать любой строке.

```
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;
}
```

ВАЖНО: порядок `addURI()` имеет значение! UriMatcher будет выглядеть в последовательном порядке от первого добавленного к последнему. Так как подстановочные знаки типа `#` и `*` являются жадными, вам нужно убедиться, что вы правильно заказали свои URI. Например:

```
uriMatcher.addURI(CONTENT_AUTHORITY, "/example", 1);
uriMatcher.addURI(CONTENT_AUTHORITY, "/*",       2);
```

это правильный порядок, так как совпадение будет искать `/example` сначала, прежде чем прибегать к совпадению `/*` . Если эти вызовы методов были отменены, и вы вызвали `uriMatcher.match("/example")` , UriMatcher перестанет искать совпадения, как только встретит путь `/*` и вернет неправильный результат!

Затем вам необходимо переопределить эти функции:

onCreate () : Инициализировать вашего провайдера. Система Android вызывает этот метод сразу же после его создания. Обратите внимание, что ваш провайдер не создается до тех пор, пока объект ContentResolver не попытается получить к нему доступ.

```
@Override
public boolean onCreate() {
    dbhelper = new DatabaseHelper(getContext());
    return true;
}
```

getType () : возвращает тип MIME, соответствующий URI содержимого

```
@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 () : получение данных у вашего провайдера. Используйте аргументы, чтобы выбрать таблицу для запроса, строки и столбцы для возврата, а также порядок сортировки результата. Верните данные как объект Cursor.

```
@Override
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
String sortOrder) {
    Cursor retCursor = dbHelper.getReadableDatabase().query(
        MyContract.TABLE_NAME, projection, selection, selectionArgs, null, null, sortOrder);
    retCursor.setNotificationUri(getContext().getContentResolver(), uri);
    return retCursor;
}
```

Вставьте новую строку в свой провайдер. Используйте аргументы для выбора таблицы назначения и получения значений столбцов. Верните URI содержимого для вновь вставленной строки.

```
@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 () : удаление строк у вашего провайдера. Используйте аргументы для выбора таблицы и строк для удаления. Верните количество удаленных строк.

```
@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 () : обновить существующие строки в вашем провайдере. Используйте аргументы для выбора таблицы и строк для обновления и получения новых значений столбца.

Верните количество обновленных строк.

```
@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) Обновить файл манифеста

```
<provider
        android:authorities="com.example.myApp"
        android:name=".DatabaseProvider"/>
```

Прочитайте Поставщик услуг онлайн: https://riptutorial.com/ru/android/topic/3075/поставщик-услуг

глава 209: Преобразование vietnamese строки в английскую строку Android

Examples

пример:

```
String myStr = convert("Lê Minh Thoại là người Việt Nam");
```

конвертирована:

```
"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;
}
```

Прочитайте Преобразование vietnamese строки в английскую строку Android онлайн: https://riptutorial.com/ru/android/topic/10946/преобразование-vietnamese-строки-в-английскую-строку-android

глава 210: Преобразование речи в текст

Examples

Речь в текст со стандартным диалогом подсказок Google Google

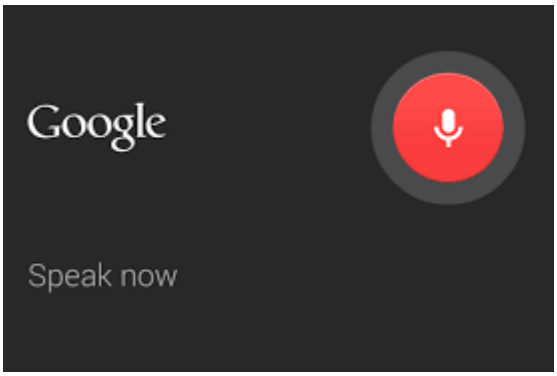
Триггерная речь для перевода текста

```
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();
    }
}
```

Получать переведенные результаты в onActivityResult

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    switch (requestCode) {
    case REQ_CODE_SPEECH_INPUT: {
        if (resultCode == RESULT_OK && null != data) {

            ArrayList<String> result = data
                    .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
            txtSpeechInput.setText(result.get(0));
        }
        break;
    }

    }
}
```

Выход



Речь в текст без диалога

Следующий код может использоваться для запуска преобразования речи в текст без отображения диалогового окна:

```
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);
}
```

Пользовательский класс слушателя `CustomRecognitionListener` используемый в приведенном выше коде, реализован следующим образом:

```
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);
    }
}
```

Прочитайте Преобразование речи в текст онлайн: https://riptutorial.com/ru/android/topic/6252/преобразование-речи-в-текст

глава 211: Проведите по экрану

Синтаксис

1. **setColorSchemeResources** устанавливает цвета **индикатора** SwipeToRefreshLayout
2. **setOnRefreshListener** устанавливает, что делать, когда макет
3. **app: layout_behavior = "@ string / appbar_scrolling_view_behavior",** если у вас есть панель инструментов с вашим макетом, добавьте это с помощью scrollFlags на панели инструментов, и панель инструментов будет скользить во время прокрутки вниз и снова входить во время прокрутки вверх.

Examples

Проведите по экрану с помощью RecyclerView

Чтобы добавить макет **Swipe To Refresh** с помощью **RecyclerView,** добавьте следующее в файл макета Activity / Fragment:

```
<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>
```

В вашей деятельности / фрагменте добавьте следующее для инициализации **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
        }
    });
```

Как добавить Swipe-to-Refresh к вашему приложению

Убедитесь, что в файл `build.gradle` вашего приложения добавлена `build.gradle` зависимость:

```
compile 'com.android.support:support-core-ui:24.2.0'
```

Затем добавьте `SwipeRefreshLayout` в ваш макет:

```
<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>
```

Наконец, реализуем прослушиватель `SwipeRefreshLayout.OnRefreshListener` .

```
mSwipeRefreshLayout = (SwipeRefreshLayout) findViewById(R.id.swipe_refresh_layout);
mSwipeRefreshLayout.setOnRefreshListener(new OnRefreshListener() {
    @Override
    public void onRefresh() {
         // your code
    }
});
```

Прочитайте Проведите по экрану онлайн: https://riptutorial.com/ru/android/topic/5241/проведите-по-экрану

глава 212: Проверить подключение к данным

Examples

Проверьте подключение к данным

Этот метод предназначен для проверки соединения данных с помощью ping определенного IP или имени домена.

```
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;
}
```

Проверьте соединение с помощью ConnectivityManager

```
    public static boolean isConnectedNetwork (Context context) {

        ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
        return cm.getActiveNetworkInfo () != null && cm.getActiveNetworkInfo
().isConnectedOrConnecting ();

    }
```

Использовать сетевые намерения для выполнения задач, пока данные разрешены

Когда ваше устройство подключается к сети, отправляется намерение. Многие приложения не проверяют эти намерения, но чтобы ваше приложение работало правильно, вы можете слушать сетевые изменения, которые расскажут вам, когда возможно общение. Чтобы проверить подключение к сети, вы можете, например, использовать следующее предложение:

```
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
}
```

Прочитайте Проверить подключение к данным онлайн:
https://riptutorial.com/ru/android/topic/8670/проверить-подключение-к-данным

глава 213: Проверка подключения к Интернету

Вступление

Этот метод используется для проверки погоды. WI-Fi подключен или нет.

Синтаксис

- isNetworkAvailable (): проверить наличие Интернета на устройстве

параметры

параметр	подробность
контекст	Ссылка на контекст действия

замечания

Если интернет подключен, метод вернет true или false.

Examples

Проверьте, имеет ли устройство интернет-соединение

Добавьте необходимые сетевые разрешения в файл манифеста приложения:

```
<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;
}
```

Как проверить силу сети в 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
            }
        }
```

Как проверить силу сети

Чтобы проверить точную прочность в децибелах,

```
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
        }
    }
```

Чтобы проверить тип сети, используйте этот класс-

```
 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 "";
        }
    }

}
```

Прочитайте Проверка подключения к Интернету онлайн:
https://riptutorial.com/ru/android/topic/3918/проверка-подключения-к-интернету

глава 214: Проверка электронной почты

Examples

Проверка адреса электронной почты

Добавьте следующий метод проверки правильности адреса электронной почты:

```
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();
}
```

Вышеуказанный метод может быть легко проверен путем преобразования текста виджета `EditText` в `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();
}
```

Проверка адреса электронной почты с использованием шаблонов

```
  if (Patterns.EMAIL_ADDRESS.matcher(email).matches()){
       Log.i("EmailCheck","It is valid");
  }
```

Прочитайте Проверка электронной почты онлайн: https://riptutorial.com/ru/android/topic/5605/проверка-электронной-почты

глава 215: Программирование на Android с Kotlin

Вступление

Использование Kotlin с Android Studio - легкая задача, так как Kotlin разработан JetBrains. Это та самая компания, которая стоит за IntelliJ IDEA - базовой IDE для Android Studio. Вот почему почти нет проблем с совместимостью.

замечания

Если вы хотите узнать больше о языке программирования Kotlin, ознакомьтесь с документацией .

Examples

Установка плагина Kotlin

Во-первых, вам нужно установить плагин Kotlin.

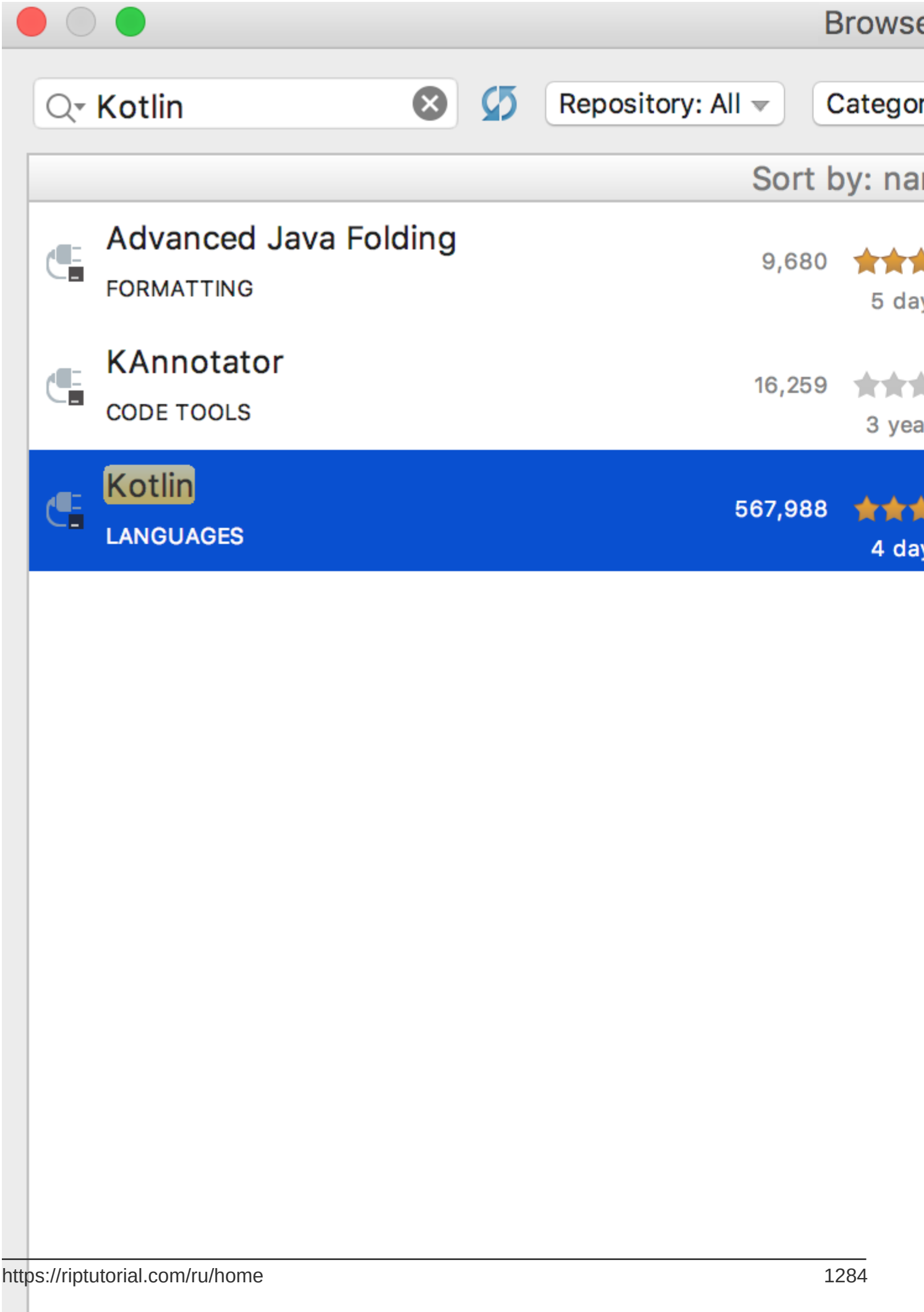
Для Windows:

- Перейдите к `File` → `Settings` → `Plugins` → `Install JetBrains plugin`

Для Mac:

- Перейдите в `Android Studio` → `Preferences` → `Plugins` → `Install JetBrains plugin`

И затем найдите и установите Kotlin. После этого вам нужно будет перезапустить среду IDE.


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

Kotlin или изменить существующий проект. Для этого вам необходимо:

1. **Добавьте зависимость от корневого файла градиента** - вам нужно добавить зависимость для `kotlin-android` к корневому файлу `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. **Применить Kotlin Android Plugin** - просто добавьте `apply plugin: 'kotlin-android'` в файл `build.gradle` модуля.

3. **Добавьте зависимость от Kotlin stdlib** - добавьте зависимость к `'org.jetbrains.kotlin:kotlin-stdlib:1.1.2'` в раздел зависимости в файле `build.gradle` модуля.

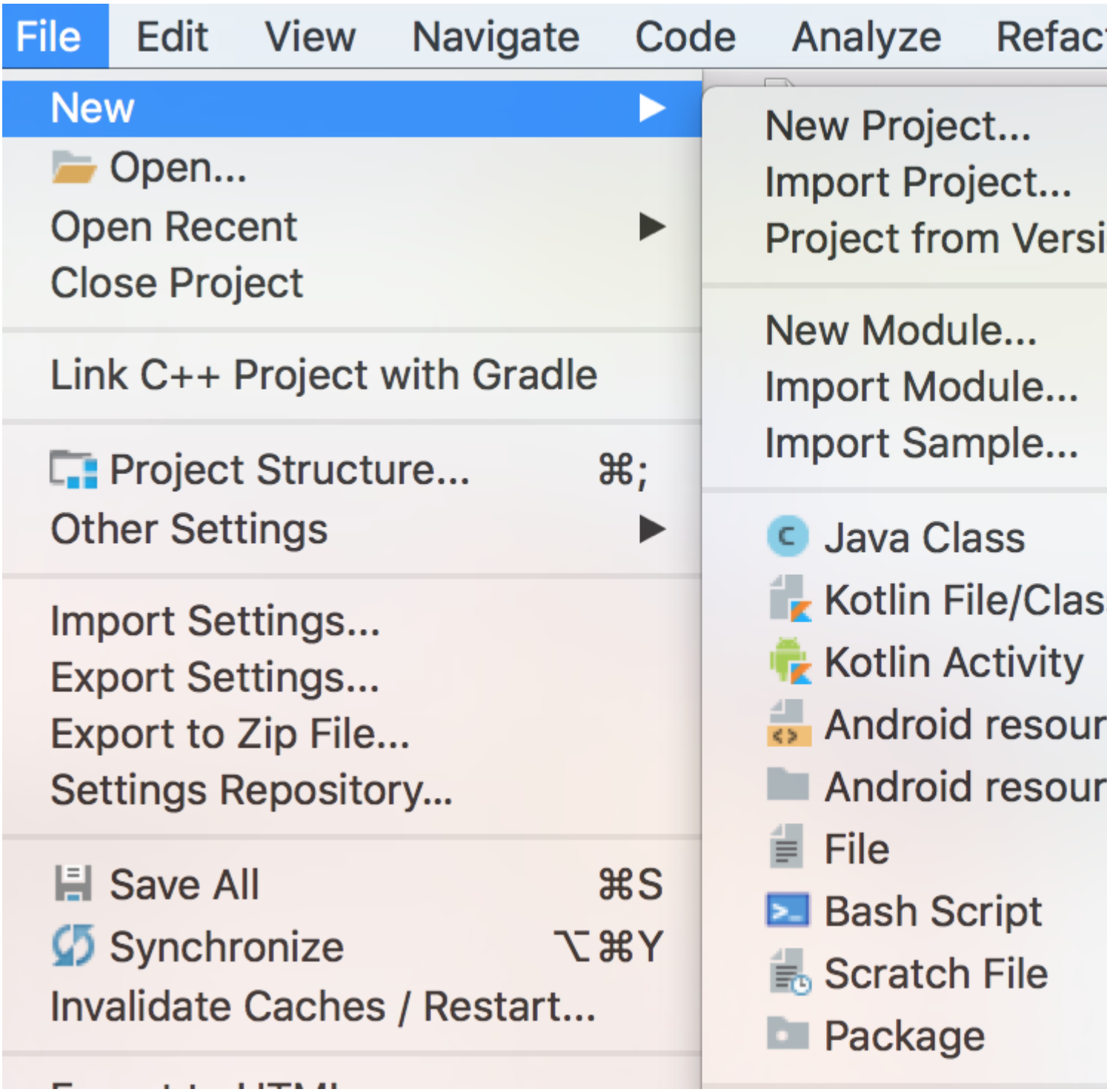
Для нового проекта файл `build.gradle` может выглядеть так:

```
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'
}
```

Создание новой деятельности Kotlin

1. Нажмите « `File` → `New` → « `Kotlin Activity` .
2. Выберите тип действия.
3. Выберите имя и другие параметры для Activity.
4. Конец.

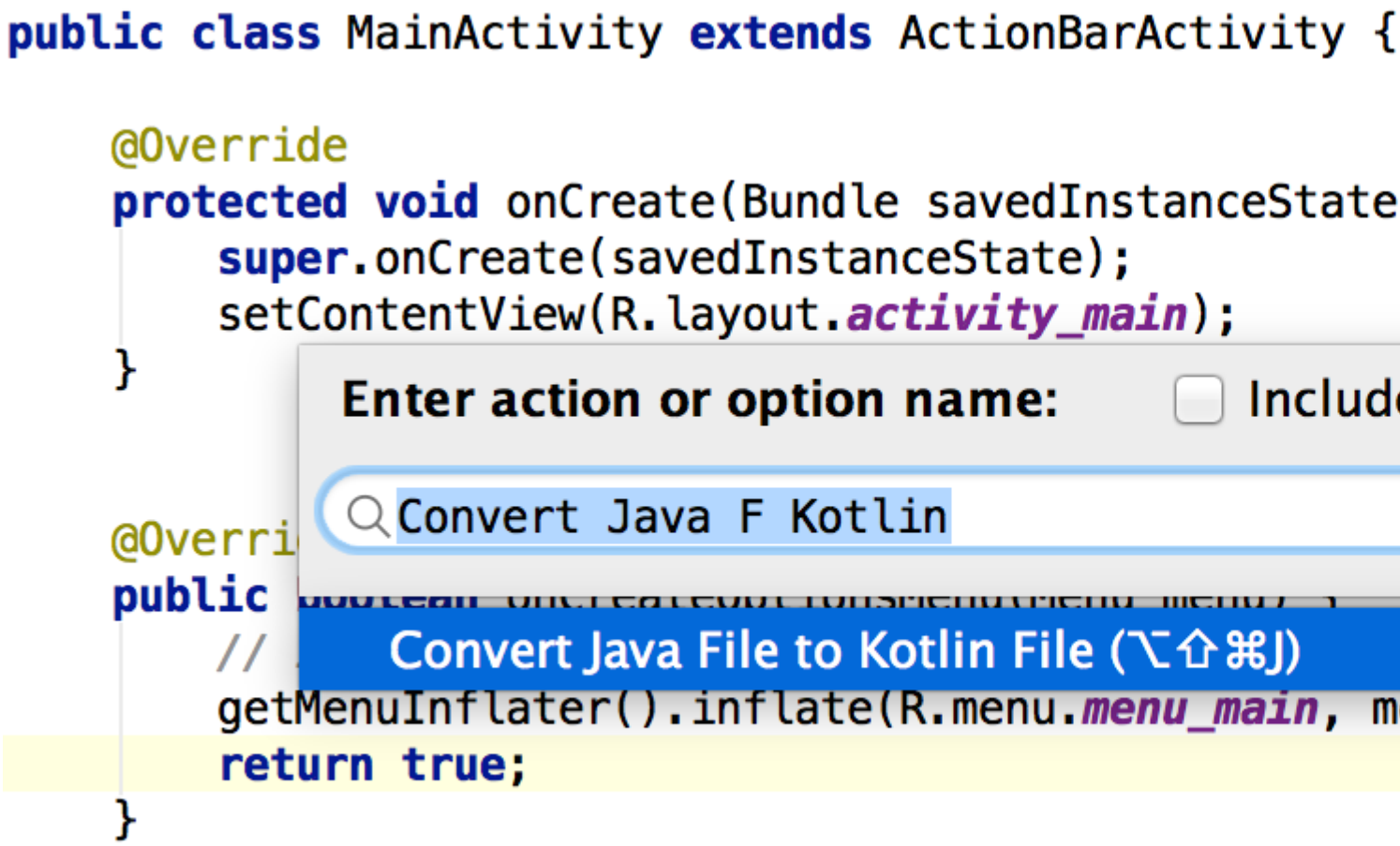


Конечный класс может выглядеть следующим образом:

```
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)
    }
}
```

Преобразование существующего Java-кода в Kotlin

Плагин Kotlin для Android Studio поддерживает преобразование существующих файлов Java в файлы Kotlin. Выберите файл Java и вызовите действие Преобразовать файл Java в файл Kotlin:



Запуск новой операции

```
fun startNewActivity(){
    val intent: Intent = Intent(context, Activity::class.java)
    startActivity(intent)
}
```

Вы можете добавить дополнительные функции к цели, как в Java.

```
fun startNewActivityWithIntents(){
    val intent: Intent = Intent(context, Activity::class.java)
    intent.putExtra(KEY_NAME, KEY_VALUE)
    startActivity(intent)
}
```

Прочитайте Программирование на Android с Kotlin онлайн:
https://riptutorial.com/ru/android/topic/9623/программирование-на-android-с-kotlin

глава 216: Просмотр объявлений Google

Examples

Основная настройка объявлений

Вам нужно добавить следующие зависимости:

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

а затем поместите это в тот же файл.

```
apply plugin: 'com.google.gms.google-services'
```

Затем вам нужно добавить соответствующую информацию в ваш файл strings.xml.

```
<string name="banner_ad_unit_id">ca-app-pub-####/####</string>
```

Затем разместите рекламный блок везде, где хотите, и создайте его так же, как и любое другое представление.

```
    <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>
```

И последнее, но не менее важное: бросьте это в свой 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);
```

Если вы скопировали именно то, у вас должно быть небольшое рекламное объявление. Просто разместите больше AdView, где бы вы ни нуждались в них больше.

Добавление межстраничного объявления

Межстраничные объявления - это полноэкранные объявления, которые охватывают интерфейс их хост-приложения. Они обычно отображаются в естественных точках перехода в потоке приложения, например между действиями или во время паузы между уровнями в игре.

Убедитесь, что у вас есть необходимые разрешения в файле `Manifest` :

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

1. Перейдите в свою учетную запись AdMob .
2. Перейдите на вкладку « **Монетизация** ».
3. Выберите или создайте приложение и выберите платформу.
4. Выберите Interstitial и укажите название рекламного блока.
5. После создания рекламного блока вы можете заметить идентификатор рекламного блока на панели управления. Например: ca-app-pub-00000000000/000000000
6. Добавить зависимости

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

Это должно быть на дне.

```
apply plugin: 'com.google.gms.google-services'
```

Добавьте свой **идентификатор рекламного блока** в файл `strings.xml`

```
<string name="interstitial_full_screen">ca-app-pub-00000000/00000000</string>
```

Добавьте в манифест конфигурационные переменные и метаданные:

```
<activity
            android:name="com.google.android.gms.ads.AdActivity"

android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScree

            android:theme="@android:style/Theme.Translucent" />
```

а также

```
<meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
```

Деятельность:

```
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();
        }
    }

}
```

Теперь AdActivity покажет полноэкранное объявление.

Прочитайте Просмотр объявлений Google онлайн: https://riptutorial.com/ru/android/topic/5984/просмотр-объявлений-google

глава 217: Публикация библиотеки в репозитории Maven

Examples

Опубликовать файл .aar в Maven

Чтобы публиковать в репозитории в формате Maven, можно использовать плагин «maven-publish» для градиента.

Плагин должен быть добавлен в файл `build.gradle` в библиотечном модуле.

```
apply plugin: 'maven-publish'
```

Вы должны определить публикацию и ее атрибуты идентификации в файле `build.gradle` . Эти атрибуты идентификации будут показаны в сгенерированном файле pom и в будущем для импорта этой публикации вы будете использовать их. Вам также необходимо определить, какие артефакты вы хотите опубликовать, например, я просто хочу опубликовать сгенерированный файл .aar после создания библиотеки ,

```
publishing {
    publications {
        myPulication(MavenPublication) {
            groupId 'com.example.project'
            version '1.0.2'
            artifactId 'myProject'
            artifact("$buildDir/outputs/aar/myProject.aar")
        }
    }
}
```

Вам также нужно будет указать URL своего репозитория

```
publishing{
    repositories {
        maven {
            url "http://www.myrepository.com"
        }
    }
}
```

Вот полный файл `build.gradle` библиотеки

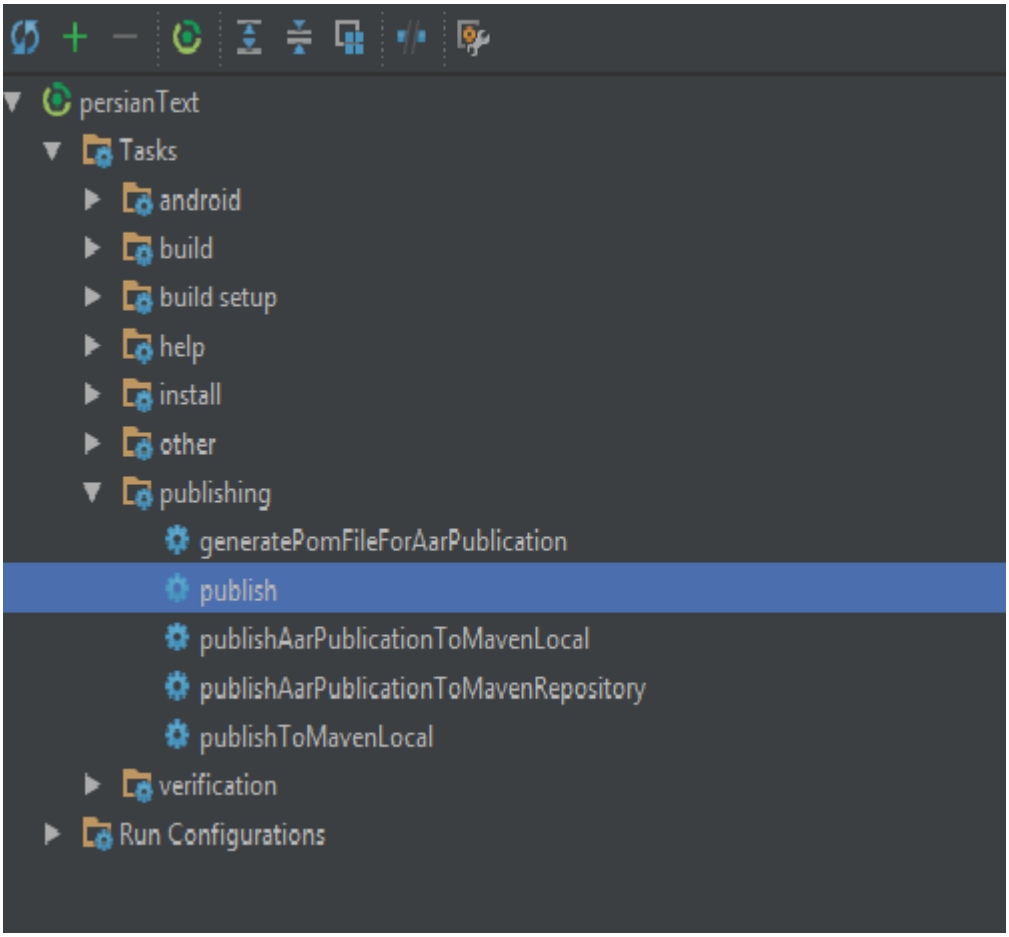
```
apply plugin: 'com.android.library'
apply plugin: 'maven-publish'

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"
        }
    }
}
```

Для публикации вы можете запустить команду консоли gradle

```
gradle publish
```

или вы можете перейти с панели задач градации



Прочитайте Публикация библиотеки в репозитории Maven онлайн:
https://riptutorial.com/ru/android/topic/9359/публикация-библиотеки-в-репозитории-maven

глава 218: Публиковать файл .aar для Apache Archiva с Gradle

Examples

Пример простой реализации

```
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'
            }
        }
    }

}
```

Прочитайте Публиковать файл .aar для Apache Archiva с Gradle онлайн:
https://riptutorial.com/ru/android/topic/6453/публиковать-файл--aar-для-apache-archiva-c-gradle

глава 219: Разархивировать файл в Android

Examples

Разархивировать файл

```
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;}
```

Прочитайте Разархивировать файл в Android онлайн:
https://riptutorial.com/ru/android/topic/3927/разархивировать-файл-в-android

глава 220: Разбиение страницы в RecyclerView

Вступление

Пагинация - обычная проблема для множества мобильных приложений, которые нуждаются в обработке списков данных. Большинство мобильных приложений теперь начинают использовать модель «бесконечной страницы», где прокрутка автоматически загружается в новом контенте. CWAC Endless Adapter упрощает использование этого шаблона в приложениях 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

открытый интерфейс 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);
        }
    }

}
```

Прочитайте Разбиение страницы в RecyclerView онлайн:
https://riptutorial.com/ru/android/topic/9243/разбиение-страницы-в-recyclerview

глава 221: Разработка Android-игр

Вступление

Краткое введение в создание игры на платформе Android с использованием Java

замечания

- Первый пример охватывает основы: целей нет, но он показывает вам, как вы создаете основную часть 2D-игры с использованием SurfaceView.
- Обязательно сохраняйте важные данные при создании игры; все остальное будет потеряно

Examples

Игра с использованием Canvas и SurfaceView

Это описывает, как вы можете создать базовую 2D-игру, используя SurfaceView.

Во-первых, нам нужна деятельность:

```
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.
    }

}
```

Активность также должна быть объявлена в манифесте Android.

Теперь для самой игры. Сначала мы начинаем с реализации игрового потока:

```
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");
    }
}
```

Это основная часть. Теперь у вас есть возможность рисовать на экране.

Теперь давайте начнем с добавления к целым числам:

```
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;
```

Для этой следующей части вам понадобится изображение. Он должен быть около 100x100, но он может быть больше или меньше. Для обучения также может использоваться Rect (но это требует изменения кода немного вниз)

Теперь мы объявляем Bitmap:

```
private Bitmap PLAYER_BMP = BitmapFactory.decodeResource(getResources(),
R.drawable.my_player_drawable);
```

В рендеринге нам нужно нарисовать это растровое изображение.

```
...
c.drawBitmap(PLAYER_BMP, x, y, null);
...
```

ПЕРЕД НАЧАЛОМ РАБОТЫ еще предстоит кое-что сделать

Сначала нам нужно булево:

```
boolean up = false;
```

в onTouchEvent добавим:

```
if(ev.getAction() == MotionEvent.ACTION_DOWN){
    up = true;
}else if(ev.getAction() == MotionEvent.ACTION_UP){
    up = false;
}
```

И в тике нам нужно это, чтобы переместить плеер:

```
if(up){
    velY -=1;
}
else{
    velY +=1;
}
if(velY >14)velY = 14;
if(velY <-14)velY = -14;
y += velY *2;
```

и теперь нам это нужно в 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();
```

И нам нужны эти переменные:

```
public static int WIDTH, HEIGHT;
```

На этом этапе игра будет запущена. Это означает, что вы можете запустить его и протестировать.

Теперь у вас должно быть изображение игрока или прямоугольник, идущий вверх и вниз по экрану. Игрок может быть создан как пользовательский класс, если это необходимо. Затем все связанные с игроком вещи могут быть перемещены в этот класс и использовать экземпляр этого класса для перемещения, визуализации и выполнения другой логики.

Теперь, как вы, вероятно, видели под тестированием, он улетает с экрана. Поэтому нам нужно ограничить это.

Во-первых, нам нужно объявить Rect:

```
private Rect screen;
```

В init, после инициализации ширины и высоты, мы создаем новый прямоугольник, который является экраном.

```
screen = new Rect(0,0,WIDTH,HEIGHT);
```

Теперь нам нужен другой прямоугольник в виде метода:

```
private Rect getPlayerBound(){
    return new Rect(x, y, x + PLAYER_BMP.getWidth(), y + PLAYER_BMP.getHeight();
}
```

и в тике:

```
if(!getPlayerBound().intersects(screen){
    gameOver = true;
}
```

Реализация gameOVer также может быть использована для показа начала игры.

Другие аспекты игры стоит отметить:

Сохранение (в настоящее время отсутствует в документации)

Прочитайте Разработка Android-игр онлайн: https://riptutorial.com/ru/android/topic/10011/разработка-android-игр

глава 222: Разрешения времени выполнения в API-23 +

Вступление

Android Marshmallow представила модель разрешения Runtime Permission . Разрешения подразделяются на две категории: нормальные и опасные . где опасные разрешения теперь предоставляются пользователем во время выполнения.

замечания

Из sdk 23 Android требуется разрешение на запуск для разрешений на устройствах под управлением Android 6.0 и выше, в том, что классифицируется как опасные группы разрешений. Опасные группы разрешений - это те, которые, как считается, ставят под угрозу конфиденциальность и / или безопасность пользователя.

Ниже приведен список опасных групп разрешений:

Опасные группы разрешений

Группа разрешений
КАЛЕНДАРЬ
КАМЕРЫ
КОНТАКТЫ
МЕСТО НАХОЖДЕНИЯ
МИКРОФОН
ТЕЛЕФОН
ДАТЧИКИ
смс
МЕСТО ХРАНЕНИЯ

Любые разрешения этих групп требуют управления разрешениями времени выполнения для устройств на Android 6.0 и выше с целевым sdk 23 или выше.

Обычные разрешения

Ниже приведен список обычных разрешений. Они не считаются опасными для конфиденциальности или безопасности пользователя и поэтому не требуют разрешений времени выполнения для sdk 23 и выше.

ACCESS_LOCATION_EXTRA_COMMANDS
ACCESS_NETWORK_STATE

ACCESS_NOTIFICATION_POLICY
ACCESS_WIFI_STATE
БЛЮТУЗ
BLUETOOTH_ADMIN
BROADCAST_STICKY
CHANGE_NETWORK_STATE
CHANGE_WIFI_MULTICAST_STATE
CHANGE_WIFI_STATE
DISABLE_KEYGUARD
EXPAND_STATUS_BAR
GET_PACKAGE_SIZE
INSTALL_SHORTCUT
ИНТЕРНЕТ
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
УСТАНОВИТЬ БУДИЛЬНИК
SET_TIME_ZONE
УСТАНОВКА ОБОЕВ
SET_WALLPAPER_HINTS
TRANSMIT_IR
UNINSTALL_SHORTCUT
USE_FINGERPRINT
VIBRATE
WAKE_LOCK
WRITE_SYNC_SETTINGS

Examples

Множественные разрешения для Android 6.0

В этом примере показано, как проверять разрешения во время выполнения в Android 6 и более поздних версиях.

```
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;
        }
    }
}
```

Применение разрешений в трансляции, URI

Вы можете выполнить проверку разрешений при отправке намерения зарегистрированному широковещательному приемнику. Отправленные разрешения перекрестно проверены с теми, которые зарегистрированы в теге. Они ограничивают, кто может отправлять трансляции соответствующему получателю.

Чтобы отправить запрос широковещательной передачи с разрешениями, укажите разрешение как строку в `Context.sendBroadcast(Intent intent, String permission)` , но имейте в виду, что приложение получателя **ДОЛЖНО** иметь это разрешение для получения вашей трансляции. Приемник должен быть установлен первым перед отправителем.

Подпись метода:

```
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");
```

и вы можете указать в своем манифесте, что отправитель вещания должен включать запрашиваемое разрешение, отправленное через sendBroadcast:

```
<!--  Your special permission -->
<permission android:name="org.quadcore.mypermission"
   android:label="my_permission"
   android:protectionLevel="dangerous"></permission>
```

Также объявите разрешение в манифесте приложения, которое должно получить эту трансляцию:

```
<!--  I use the permission ! -->
<uses-permission android:name="org.quadcore.mypermission"/>
<!-- along with the receiver -->
<receiver android:name="Bcastreceiver" android:exported="true" />
```

Примечание. Как приемник, так и вещатель могут потребовать разрешения, и когда это произойдет, обе проверки разрешений должны пройти для того, чтобы Intent был доставлен связанному целевому объекту. Сначала необходимо установить приложение, определяющее разрешение.

Найдите полную документацию здесь, в разделе «Разрешения».

Несколько разрешений времени выполнения от одинаковых групп разрешений

В манифесте у нас есть четыре опасных разрешения во время выполнения из двух групп.

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

В том случае, когда требуются разрешения. Обратите внимание, что важно проверить разрешения в любой деятельности, требующей разрешений, поскольку разрешения могут быть отменены, когда приложение находится в фоновом режиме, и приложение затем выйдет из строя.

```
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();
    }
```

Нам нужно только запрашивать разрешение для одного из них из каждой группы, и все другие разрешения из этой группы предоставляются, если пользователь не отменяет разрешение.

```
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;
}
```

Это связано с тем, что пользователь разрешает или не разрешает разрешения. В этом примере, если разрешения не разрешены, приложение будет убито.

```
@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);
    }
}
```

Дополнительная информация https://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en

Использование PermissionUtil

PermissionUtil - простой и удобный способ запроса разрешений в контексте. Вы можете легко обеспечить, что должно произойти в случае всех разрешенных разрешений (`onAllGranted()`), любой запрос был отклонен (`onAnyDenied()`) или в случае необходимости рационального (`onRational()`).

В любом месте вашего AppCompatActivity или Fragment, который вы хотите запросить для разрешения пользователя

```
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);
```

И добавьте это в `onRequestPermissionsResult`

```
if(mRequestObject!=null){
    mRequestObject.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

Добавьте запрошенное разрешение на ваш AndroidManifest.xml, а также

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

Включите весь код, связанный с правами доступа, в абстрактный базовый класс и расширьте активность этого базового класса, чтобы получить более чистый / многоразовый код

```
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);
    }
}
```

Пример использования в деятельности

Действие должно расширять абстрактный базовый класс, определенный выше, следующим образом:

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

Прочитайте Разрешения времени выполнения в API-23 + онлайн:
https://riptutorial.com/ru/android/topic/1525/разрешения-времени-выполнения-в-api-23-plus

глава 223: Распознавание активности

Вступление

Распознавание активности - это обнаружение физической активности пользователя для выполнения определенных действий на устройстве, таких как определение точек при обнаружении диска, отключение Wi-Fi при потере телефона или ограничение громкости звонка, когда пользователь ходьба.

Examples

Активность Google PlayRecognitionAPI

Это простой пример того, как использовать ActivityRecognitionApi службы GooglePlay. Хотя это отличная библиотека, она не работает на устройствах, на которых не установлены службы Google Play.

Документы для API ActivityRecognition

манифест

```
<!-- 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"));
    }
}
```

Распознавание активности PathSense

Распознавание активности PathSense является еще одной хорошей библиотекой для устройств, не имеющих сервисов Google Play, поскольку они создали свою собственную модель распознавания активности, но требуют, чтобы разработчики регистрировались на http://developer.pathsense.com, чтобы получить ключ API и идентификатор клиента ,

манифест

```
<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);
    }
}
```

Прочитайте Распознавание активности онлайн: https://riptutorial.com/ru/android/topic/9831/распознавание-активности

глава 224: Регистрация и использование Logcat

Синтаксис

- `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)`

параметры

вариант	Описание
-b (буфер)	Загружает альтернативный буфер журнала для просмотра, например, события или радио. Основной буфер используется по умолчанию. См. Просмотр альтернативных буферов журнала.
-c	Очищает (очищает) весь журнал и выходит.
-d	Сбрасывает журнал на экран и выходит.
-f (имя файла)	Записывает вывод сообщения журнала в (имя файла). По умолчанию используется стандартный вывод.
-r	Распечатывает размер указанного буфера журнала и завершает работу.
-n (count)	Устанавливает максимальное количество повернутых журналов (количество). Значение по умолчанию - 4. Требуется опция -r.
-r (кбайты)	Поворачивает файл журнала каждый (кбайт) вывода. Значение по умолчанию - 16. Требуется опция -f.
-s	Устанавливает спецификацию фильтра по умолчанию для молчания.
-v (формат)	Устанавливает формат вывода сообщений журнала. По умолчанию используется короткий формат.

замечания

Определение

Logcat - это инструмент командной строки, который удаляет журнал системных сообщений, включая трассировки стека, когда устройство выдает сообщение об ошибке и сообщения, которые вы написали из вашего приложения, с классом журнала .

Когда использовать

Если вы планируете использовать методы System.out Java для печати на консоли вместо использования одного из методов журнала журнала Google, то вы должны знать, что они в основном работают одинаково. Однако лучше избегать использования методов Java, потому что дополнительная информация и форматирование, предоставляемые методами журнала Android, более выгодны. Кроме того, методы печати System.out перенаправляются на метод `Log.i()` .

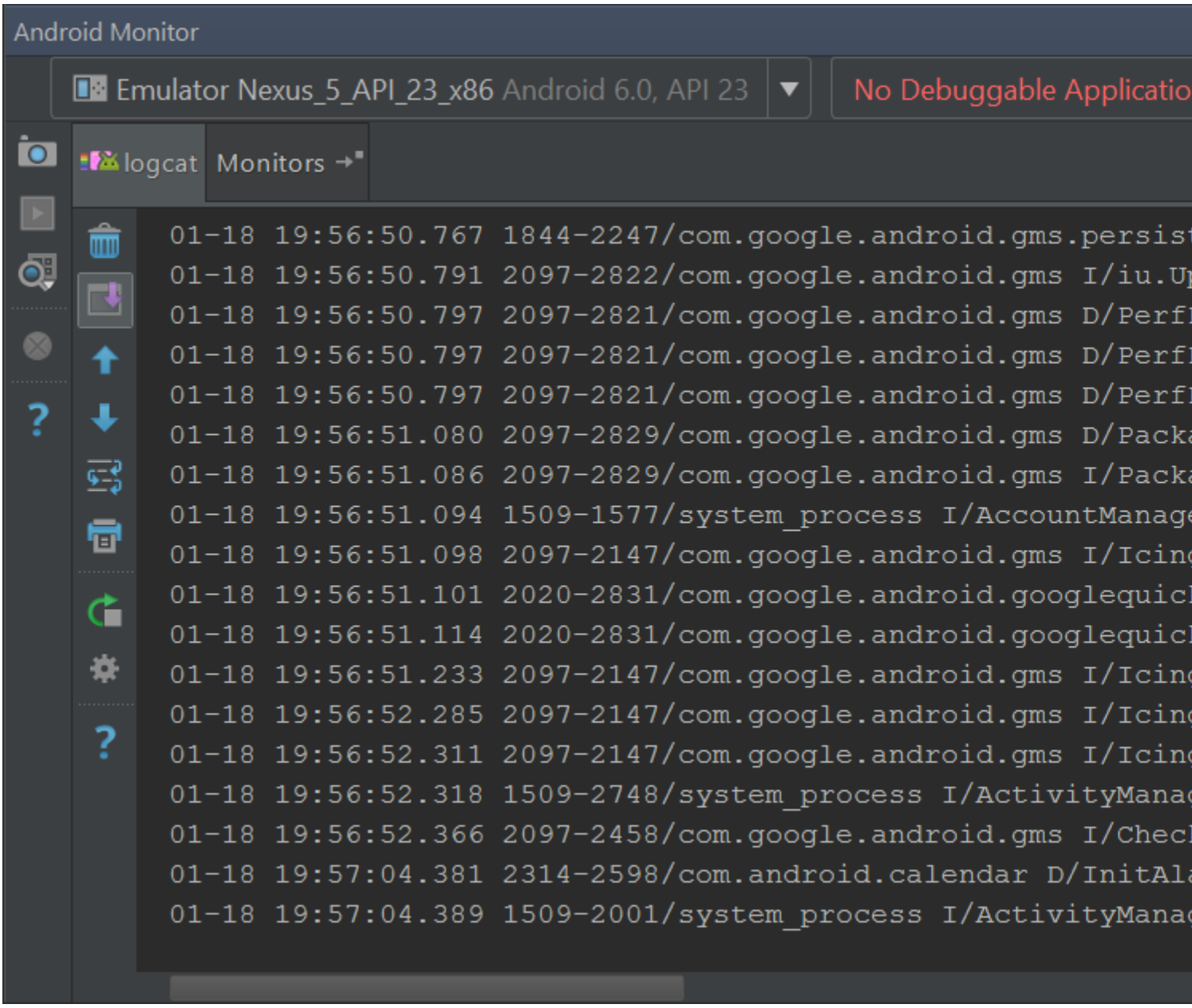
Полезные ссылки

- Официальная документация для разработчиков Android для Log и logcat .
- Вопрос Stackoveflow: Android Log.v (), Log.d (), Log.i (), Log.w (), Log.e () - Когда использовать их?

Examples

Фильтрация вывода logcat

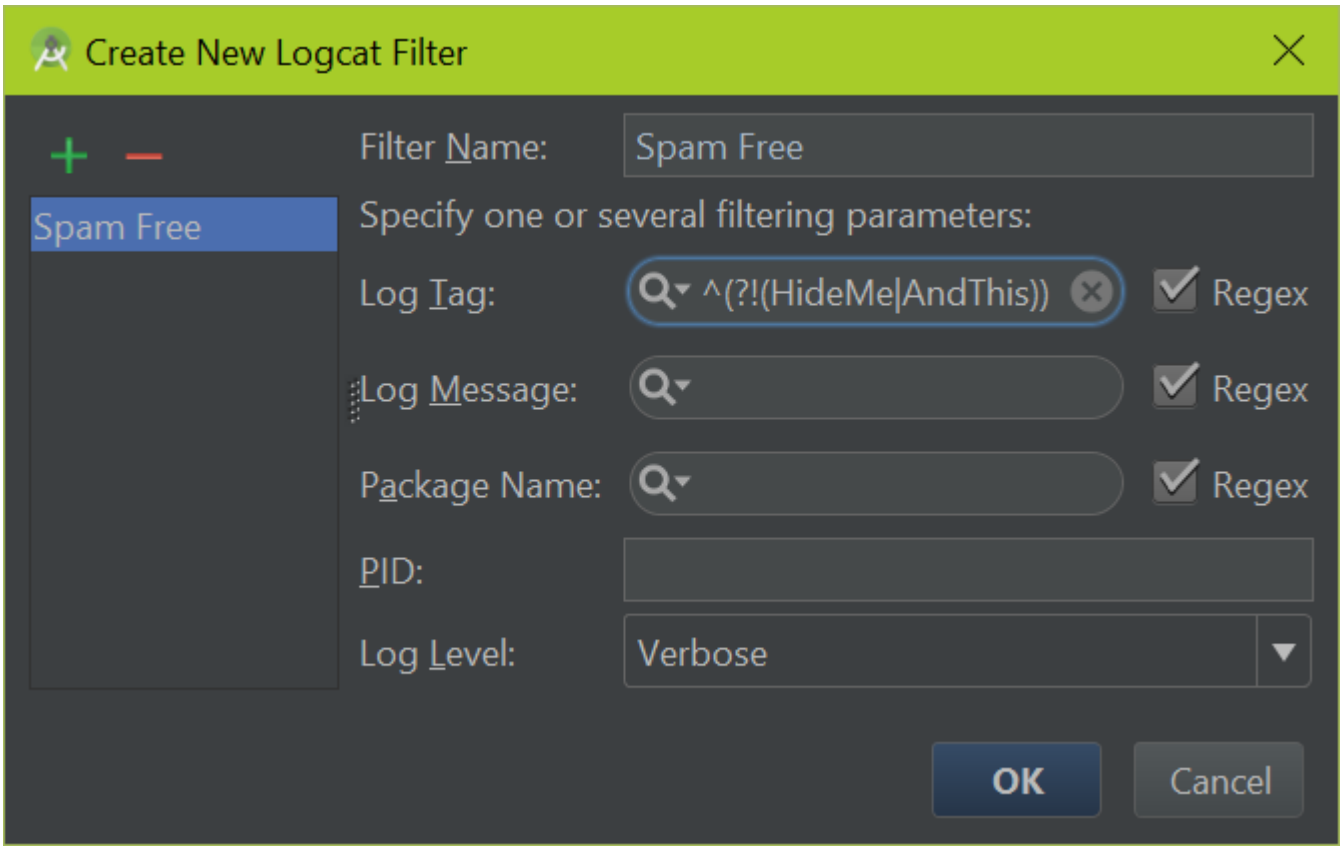
Полезно отфильтровать вывод logcat, потому что есть много сообщений, которые не представляют интереса. Чтобы отфильтровать выход, откройте «Android Monitor» и нажмите на раскрывающийся список в правом верхнем углу и выберите « *Редактировать настройку фильтра»*



Теперь вы можете добавить настраиваемые фильтры, чтобы отображать сообщения, которые представляют интерес, а также отфильтровывать известные строки журнала, которые можно безопасно игнорировать. Чтобы игнорировать часть вывода, вы можете определить регулярное выражение . Ниже приведен пример исключения совпадающих тегов:

```
^(?!(HideMe|AndThis))
```

Это можно ввести, следуя этому примеру:



Вышеприведенное является регулярным выражением, которое исключает входы. Если вы хотите добавить еще один тег в *черный список* , добавьте его после трубы `|` персонаж. Например, если вы хотите сделать черный список «GC», вы должны использовать такой фильтр:

```
^(?!(HideMe|AndThis|GC))
```

Для получения дополнительной документации и примеров посетите журнал и использование Logcat

логирование

Любое качественное приложение для Android будет отслеживать, что он делает с помощью журналов приложений. Эти журналы позволяют легко отлаживать помощь разработчика для диагностики того, что происходит с приложением. Полная документация на Android можно найти здесь , но следующее резюме:

Основная регистрация

Класс `Log` является основным источником написания журналов разработчиков, указав `tag` и `message` . Тег - это то, что вы можете использовать для фильтрации сообщений журнала, чтобы определить, какие строки относятся к вашей конкретной Деятельности. Просто позвоните

```
Log.v(String tag, String msg);
```

И система Android напишет сообщение 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
```

> СОВЕТ:
> Обратите внимание на идентификатор процесса и идентификатор потока. Если они одинаковые - журнал поступает из основного / пользовательского потока!

Любой тег можно использовать, но обычно используется имя класса как тег:

```
public static final String tag = MyAnimator.class.getSimpleName();
```

Уровни журнала

Регистратор Android имеет 6 разных уровней, каждый из которых служит определенной цели:

- `ERROR` : `Log.e()`
  - Используется для указания критического сбоя, это уровень, напечатанный при бросании `Exception` .
- `WARN` : `Log.w()`
  - Используется для указания предупреждения, в основном для восстановления отказов
- `INFO` : `Log.i()`
  - Используется для указания информации более высокого уровня о состоянии приложения
- `DEBUG` : `Log.d()`
  - Используется для записи информации, которая была бы полезной при отладке приложения, но будет мешать при запуске приложения
- `VERBOSE` : `Log.v()`
  - Используется для регистрации информации, которая отражает мелкие детали о состоянии приложения
- `ASSERT` : `Log.wtf()`
  - Используется для регистрации информации о состоянии, которое никогда не должно происходить.
  - *wtf* означает «Что такое ужасная неудача».

Мотивация для регистрации

Мотивация для ведения журнала - это легко найти ошибки, предупреждения и другую информацию, взглянув на цепочку событий из приложения. Например, представьте приложение, которое читает строки из текстового файла, но неправильно предполагает, что файл никогда не будет пустым. Трассировка журнала (приложения, которое не регистрируется) выглядит примерно так:

```
E/MyApplication: Process: com.example.myapplication, PID: 25788
                         com.example.SomeRandomException: Expected string, got 'null' instead
```

Вслед за кучей следов стека, которые в конечном итоге приведут к нарушительной линии, где переход с помощью отладчика в конечном итоге приведет к проблеме

Тем не менее, трассировка журнала приложения с включенным протоколированием может выглядеть примерно так:

```
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
```

Быстрый взгляд на журналы, и очевидно, что файл был пуст.

Что нужно учитывать при регистрации:

Хотя ведение журнала - это мощный инструмент, позволяющий разработчикам Android лучше понять внутреннюю работу своего приложения, у журналов есть некоторые недостатки.

Чтение журналов:

В Android-приложениях обычно используется несколько журналов, работающих синхронно. Таким образом, очень важно, чтобы каждый журнал был легко читаемым и содержал только необходимую, необходимую информацию.

Спектакль:

Для ведения журнала требуется небольшое количество системных ресурсов. В общем, это не требует внимания, однако, если чрезмерное использование, регистрация может негативно повлиять на производительность приложения.

Безопасность:

Недавно на рынок Google Play были добавлены несколько приложений для Android, которые позволяют пользователю просматривать журналы всех запущенных приложений. Это непреднамеренное отображение данных может позволить пользователям просматривать конфиденциальную информацию. Как правило, всегда удаляйте журналы, которые содержат непубличные данные, *прежде чем* публиковать свое приложение на рынке.

Заключение:

Ведение журнала является неотъемлемой частью приложения для Android, поскольку оно дает разработчикам. Возможность создания полезной трассировки журнала является одним из самых сложных аспектов разработки программного обеспечения, но класс журнала Android помогает сделать его намного проще.

Для получения дополнительной документации и примеров посетите журнал и использование Logcat

Вход со ссылкой на источник непосредственно с Logcat

Это хороший трюк, чтобы добавить ссылку на код, поэтому легко перейти к коду, который выдает журнал.

В следующем коде этот вызов:

```
MyLogger.logWithLink("MyTag","param="+param);
```

Это приведет к:

```
07-26...012/com.myapp D/MyTag: MyFrag:onStart(param=3)  (MyFrag.java:2366) // << logcat
converts this to a link to source!
```

Это код (внутри класса 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());
}
```

Это базовый пример, его можно легко расширить, чтобы выпустить ссылку на вызывающего абонента (подсказка: стек будет [4] вместо [3]), и вы также можете добавить другую соответствующую информацию.

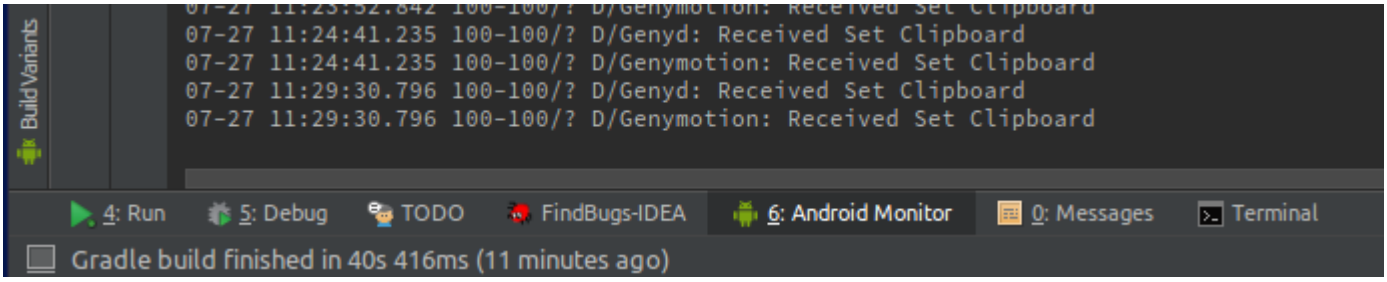
Использование Logcat

Logcat - это инструмент командной строки, который сбрасывает журнал системных сообщений, включая трассировки стека, когда устройство выдает сообщение об ошибке и сообщения, которые вы написали из вашего приложения, с классом журнала.

Выход Logcat можно отобразить в Android Monitor Android Monitor или с помощью командной строки adb.

В Android Studio

Показать, нажав значок «Android Monitor»:



Или нажав `Alt` + `6` на Windows / Linux или `CMD` + `6` на Mac.

через командную строку:

Простое использование:

```
$ adb logcat
```

С отметками времени:

```
$ adb logcat -v time
```

Фильтр по определенному тексту:

```
$ adb logcat -v time | grep 'searchtext'
```

Здесь доступно множество опций и фильтров для *logcat командной строки* , описанных здесь .
Простым, но полезным примером является следующее выражение фильтра, которое

отображает все сообщения журнала с «ошибкой» уровня приоритета по всем тегам:

```
$ adb logcat *:E
```

Создание регистрационного кода

`Live templates` `Android Studio` `Live templates` могут предлагать довольно много ярлыков для быстрого ведения журнала.
Чтобы использовать Live-шаблоны, все, что вам нужно сделать, это начать вводить имя шаблона и нажать `TAB` или ввести, чтобы вставить инструкцию.

Примеры:

- `logi` → превращается в → `android.util.Log.i(TAG, "$METHOD_NAME$: $content$");`
  - `$METHOD_NAME$` будет автоматически заменено вашим именем метода, и курсор будет ждать заполнения содержимого.
- `loge` → тот же, для ошибки
- и т. д. для остальных уровней регистрации.

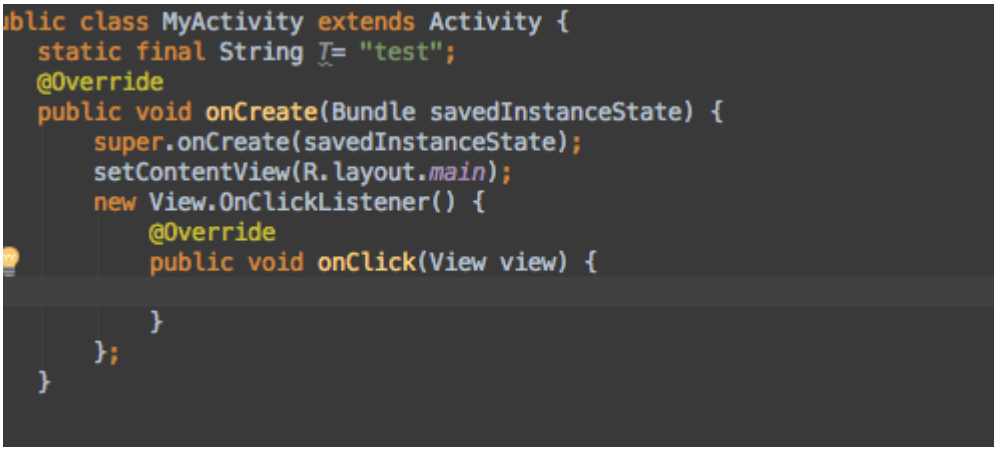
Полный список шаблонов можно найти в настройках `Android Studio` (`ALT` + `s` и введите « live»). Также можно добавить свои собственные шаблоны.

Если вы обнаружите, что `Live templates` для `Android Studio` недостаточно для ваших нужд, вы можете рассмотреть плагин Android Postfix

Это очень полезная библиотека, которая поможет вам избежать записи строки журнала вручную.

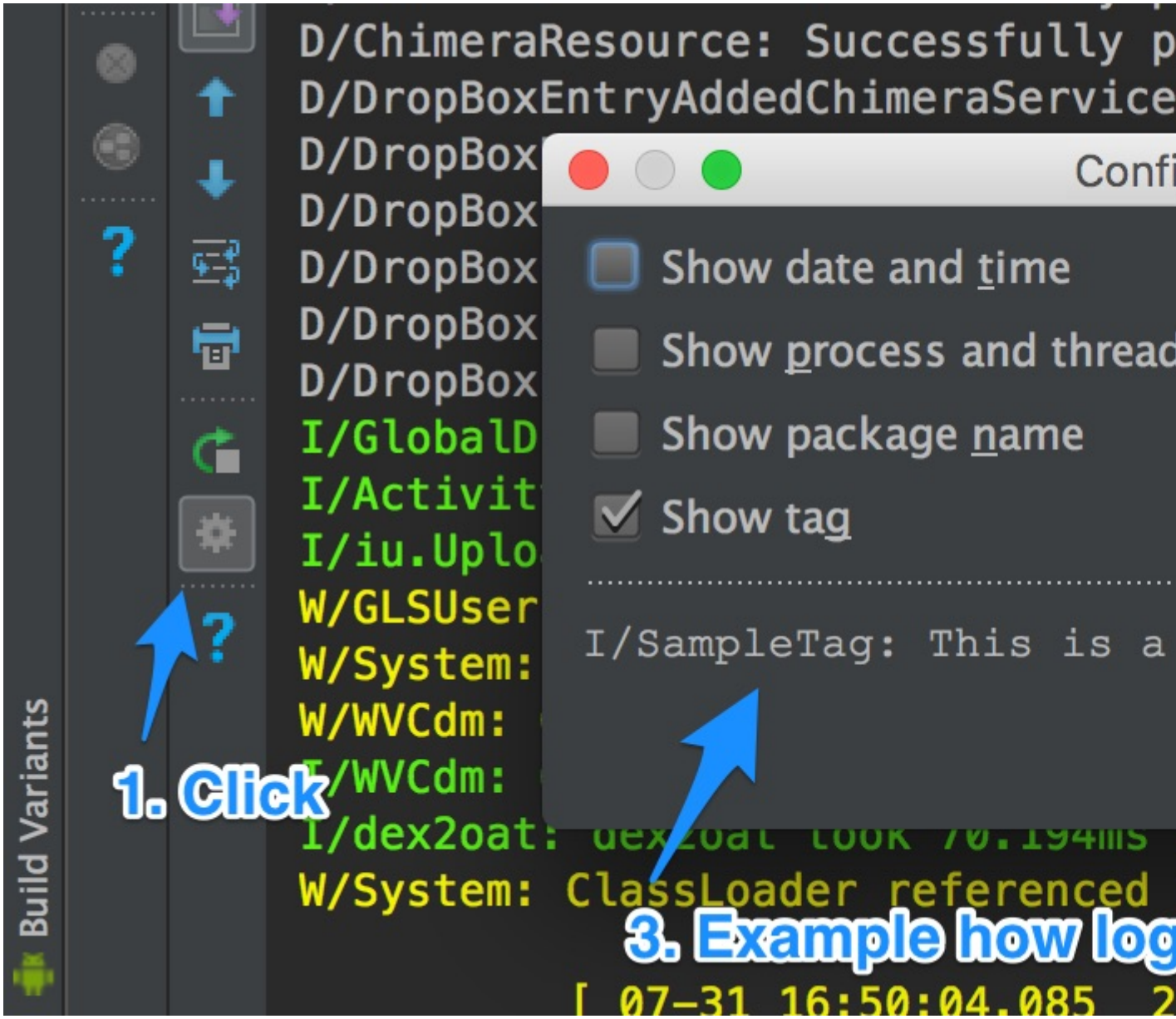
Синтаксис абсолютно прост:

.log - Регистрация. Если существует постоянная переменная «TAG», она использует « TAG». Кроме того, он использует имя класса.

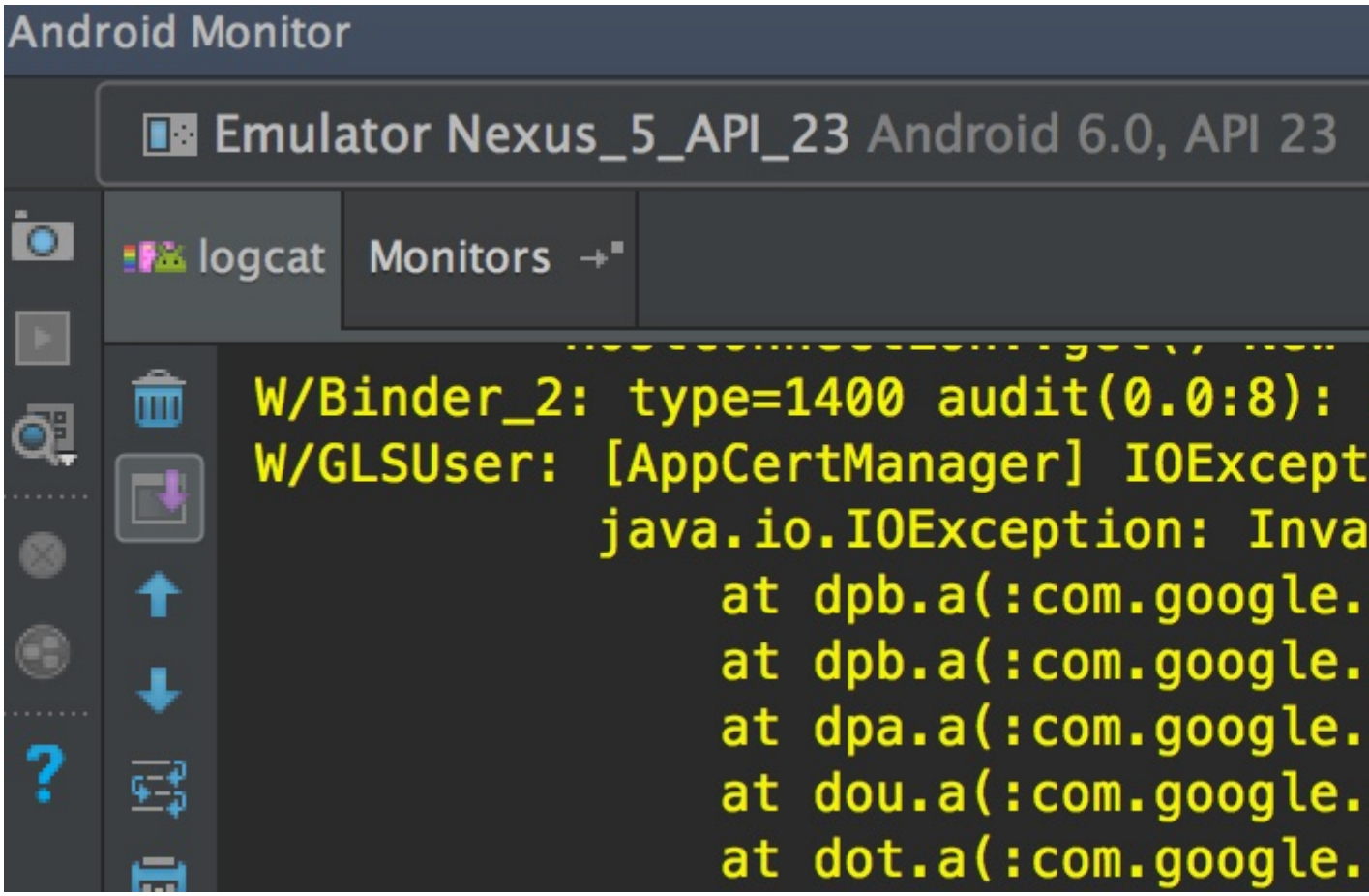


Использование Android Studio

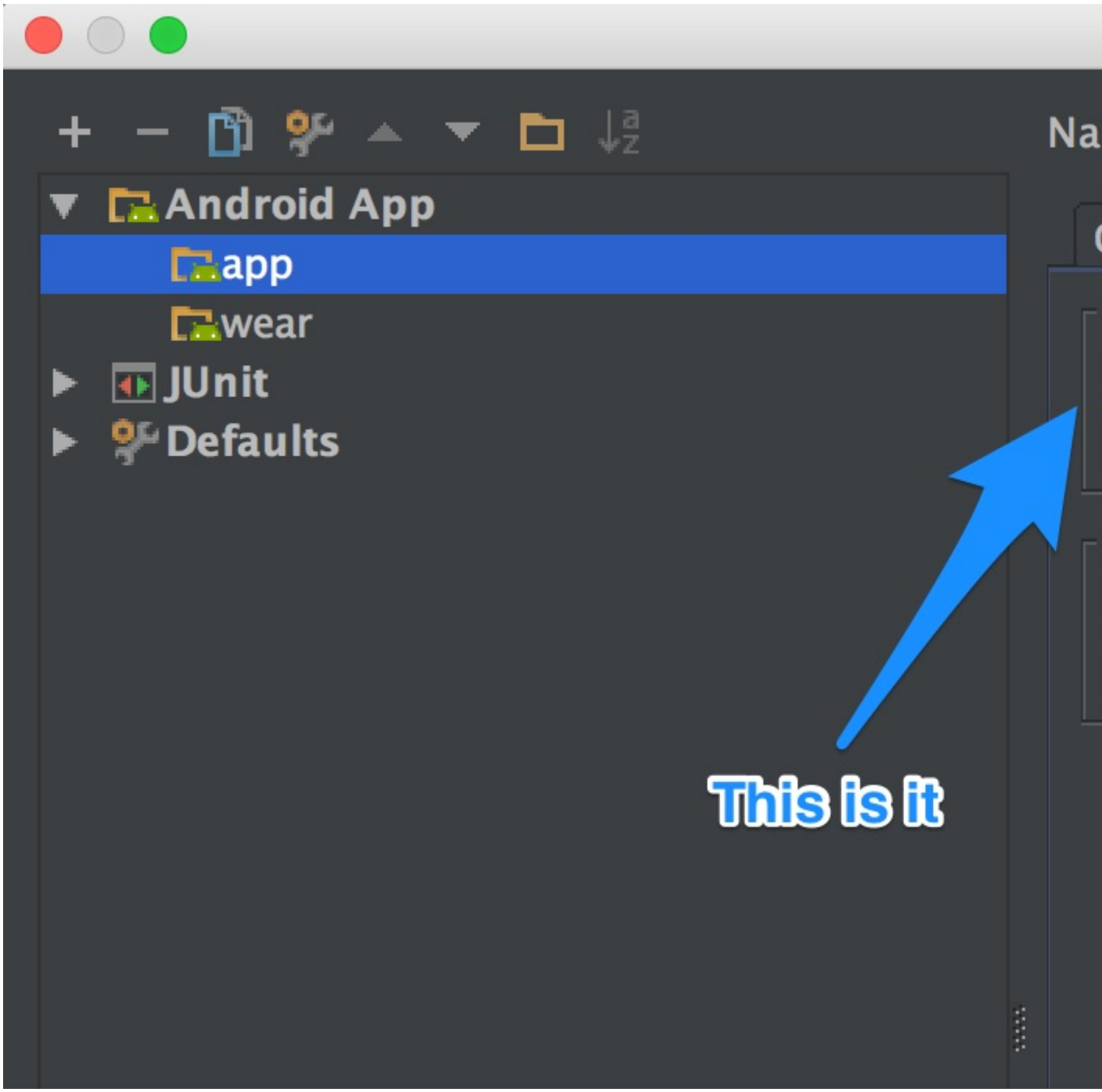
1. Скрыть / показать распечатанную информацию:



2. Контрольная многословия ведения журнала:



3. Отключить / включить окно открытия журнала при запуске приложения запуска / отладки



Очистить журналы

Чтобы очистить (очистить) весь журнал:

```
adb logcat -c
```

Прочитайте Регистрация и использование Logcat онлайн:
https://riptutorial.com/ru/android/topic/1552/регистрация-и-использование-logcat

глава 225: Редактировать текст

Examples

Работа с EditTexts

EditText - это стандартный виджет ввода текста в приложениях Android. Если пользователю нужно вводить текст в приложение, это основной способ сделать это.

Редактировать текст

Существует множество важных свойств, которые можно настроить для настройки поведения EditText. Некоторые из них перечислены ниже. Ознакомьтесь с официальным руководством по текстовым полям для получения более подробной информации о полевых условиях.

использование

EditText добавляется в макет со всеми стандартными поведением со следующим XML:

```
<EditText
    android:id="@+id/et_simple"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
</EditText>
```

Обратите внимание, что EditText - это просто тонкое расширение TextView и наследует все те же свойства.

Получение значения

Получение значения текста, введенного в EditText, выглядит следующим образом:

```
EditText simpleEditText = (EditText) findViewById(R.id.et_simple);
String strValue = simpleEditText.getText().toString();
```

Дальнейшая настройка входа

Возможно, мы захотим ограничить запись одной строкой текста (избегайте новых строк):

```
<EditText
  android:singleLine="true"
  android:lines="1"
/>
```

Вы можете ограничить символы, которые можно ввести в поле, используя атрибут цифр:

```
<EditText
  android:inputType="number"
  android:digits="01"
/>
```

Это ограничило бы введенные цифры только «0» и «1». Мы могли бы ограничить общее количество символов:

```
<EditText
  android:maxLength="5"
/>
```

Используя эти свойства, мы можем определить ожидаемое поведение ввода для текстовых полей.

Регулировка цветов

Вы можете отрегулировать цвет фона выделения выделенного текста в EditText с помощью свойства `android:textColorHighlight` :

```
<EditText
    android:textColorHighlight="#7cff88"
/>
```

Отображение подсказок для закладок

Возможно, вы захотите установить подсказку для элемента управления EditText, чтобы пригласить пользователя для определенного ввода с помощью:

```
<EditText
    ...
    android:hint="@string/my_hint">
</EditText>
```

Советы

Изменение цвета нижней строки

Предполагая, что вы используете библиотеку AppCompat, вы можете переопределить стили colorControlNormal, colorControlActivated и 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>
```

Если вы не видите эти стили, применяемые в диалоговом окне DialogFragment, существует известная ошибка при использовании LayoutInflater, переданного в метод onCreateView ().

Эта проблема уже исправлена в библиотеке AppCompat v23. См. Это руководство о том, как обновить. Другим временным обходным решением является использование надувного устройства макета Activity вместо того, которое было передано в метод onCreateView ():

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = getActivity().getLayoutInflater().inflate(R.layout.dialog_fragment,
container);
  }
```

Прослушивание ввода EditText

Посмотрите на базовые прослушиватели событий, чтобы посмотреть, как слушать изменения в EditText и выполнять действие, когда происходят эти изменения.

Отображение информации о плавающей этикетке

Традиционно EditText скрывает сообщение подсказки (объяснено выше) после того, как пользователь начинает вводить текст. Кроме того, все сообщения об ошибках проверки должны были управляться разработчиком вручную.

С помощью `TextInputLayout` вы можете настроить плавающую метку для отображения подсказок и сообщений об ошибках. Здесь вы можете найти более подробную информацию .

Настройка InputType

Текстовые поля могут иметь разные типы ввода, такие как число, дата, пароль или адрес электронной почты. Тип определяет, какие символы разрешены внутри поля, и может предложить виртуальной клавиатуре оптимизировать свой макет для часто используемых символов.

По умолчанию любое текстовое содержимое в элементе управления `EditText` отображается как обычный текст. `inputType` атрибут `inputType` , мы можем облегчить ввод различных типов информации, таких как номера телефонов и пароли:

```
<EditText
    ...
    android:inputType="phone">
</EditText>
```

Наиболее распространенные типы входных данных включают:

Тип	Описание
textUri	Текст, который будет использоваться как URI
textEmailAddress	Текст, который будет использоваться в качестве адреса электронной почты
textPersonName	Текст, который является именем человека
textPassword	Текст, который является паролем, который должен быть закрыт
число	Только числовое поле
Телефон	Для ввода номера телефона
Дата	Для ввода даты
время	Для ввода времени
textMultiLine	Разрешить несколько строк текста в поле

`android:inputType` также позволяет указать некоторые поведения клавиатуры, например, использовать ли все новые слова или использовать такие функции, как автозаполнение и орфографические предложения.
Вот некоторые из общих значений типа ввода, которые определяют поведение клавиатуры:

Тип	Описание
textCapSentences	Обычная текстовая клавиатура, которая заглавная буква для каждого нового предложения
textCapWords	Обычная текстовая клавиатура, которая заглаживает каждое слово. Хорошо подходит для названий или имен людей
textAutoCorrect	Обычная текстовая клавиатура, которая исправляет часто ошибочные слова

Вы можете установить несколько атрибутов `inputType` если это необходимо (в `inputType` «|»).
Пример:

```
<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" />
```

Вы можете просмотреть список всех доступных типов входных здесь .

атрибут `inputype`

атрибут `inputype` в `inputype EditText` : *(проверен на Android 4.4.3 и 2.3.3)*

```
<EditText android:id="@+id/et_test" android:inputType="?????"/>
```

textLongMessage = Клавиатура: алфавит / значение по умолчанию. Введите кнопку: Отправить / Далее. Эмоция: да. Корпус: строчный. Предложение: да. Добавлять. обугливается: **и.** и все

textFilter = Клавиатура: алфавит / по умолчанию. Введите кнопку: Отправить / Далее. Эмоция: да. Корпус: строчный. **Предложение: нет** . Добавлять. обугливается: **и.** и все

textCapWords = Клавиатура: алфавит / значение по умолчанию. Введите кнопку: Отправить / Далее. Эмоция: да. **Случай: Случай верблюда** . Предложение: да. Добавлять. обугливается: **и.** и все

textCapSentences = Клавиатура: алфавит / значение по умолчанию. Введите кнопку: Отправить / Далее. Эмоция: да. **Случай: случай предложения** . Предложение: да. Добавлять. обугливается: **и.** и все

time = Клавиатура: числовая. Введите кнопку: Отправить / Далее. Эмоции: нет. Случай: -. **Предложение: нет** . Добавлять. символы::

textMultiLine = Клавиатура: алфавит / значение по умолчанию. **Введите кнопку: nextline** . Эмоция: да. Корпус: строчный. Предложение: да. Добавлять. обугливается: **и.** и все

number = Клавиатура: числовая . Введите кнопку: Отправить / Далее. Эмоции: нет. Случай: -. Предложение: нет. **Добавлять. символы: ничего**

textEmailAddress = Клавиатура: алфавит / значение по умолчанию. Введите кнопку: Отправить / Далее. **Эмоции: нет** . Корпус: строчный. **Предложение: нет** . Добавлять. символы: @ и **.** и все

(Нет типа) = Клавиатура: алфавит / по умолчанию. **Введите кнопку: nextline** . Эмоция: да. Корпус: строчный. Предложение: да. Добавлять. обугливается: **и.** и все

textPassword = Клавиатура: алфавит / значение по умолчанию. Введите кнопку: Отправить / Далее. Эмоции: нет. Корпус: строчный. **Предложение: нет** . Добавлять. обугливается: **и.** и все

text = Клавиатура: Клавиатура: алфавит / по умолчанию. Введите кнопку: Отправить / Далее. Эмоция: да. Корпус: строчный. Предложение: да. Добавлять. обугливается: **и.** и все

textShortMessage = Клавиатура: алфавит / значение по умолчанию. **Введите кнопку: эмоции** . Эмоция: да. Корпус: строчный. Предложение: да. Добавлять. обугливается: **и.** и все

textUri = Клавиатура: алфавит / значение по умолчанию. Введите кнопку: Отправить / Далее. Эмоции: нет. Корпус: строчный. **Предложение: нет** . Добавлять. символы: **/** и **.** и все

textCapCharacters = Клавиатура: алфавит / по умолчанию. Введите кнопку: Отправить / Далее. Эмоция: да. **Случай: ВЕРХНИЙ** . Предложение: да. Добавлять. обугливается: **и.** и все

phone = Клавиатура: числовая . Введите кнопку: Отправить / Далее. Эмоции: нет. Случай: -. **Предложение: нет** . Добавлять. chars: *** #. - / () WPN, + **

textPersonName = Клавиатура: алфавит / значение по умолчанию. Введите кнопку: Отправить / Далее. Эмоция: да. Корпус: строчный. Предложение: да. Добавлять. обугливается: **и.** и все

Примечание. `Auto-capitalization` изменяет поведение по умолчанию.

Примечание 2: На `Numeric keyboard` ВСЕ номера указаны на английском языке 1234567890.

Примечание 3: Настройка `Correction/Suggestion` изменит поведение по умолчанию.

Скрытие SoftKeyboard

Скрытие Softkeyboard является **основным требованием,** обычно при работе с EditText. По *умолчанию программная* клавиатура может быть закрыта только нажатием кнопки «Назад», поэтому большинство разработчиков используют InputMethodManager, чтобы заставить Android скрыть виртуальную клавиатуру, вызывающую hideSoftInputFromWindow, и передать в токене окна, содержащего ваше сфокусированное представление. Код, чтобы сделать следующее:

```
public void hideSoftKeyboard()
{
        InputMethodManager inputMethodManager = (InputMethodManager)
getSystemService(Activity.INPUT_METHOD_SERVICE);
        inputMethodManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(), 0);
}
```

Код является прямым, но возникают другие серьезные проблемы, связанные с тем, что функция hide должна вызываться при возникновении какого-либо события. Что делать, если вам нужна Softkeyboard, скрытая при нажатии в любом месте, кроме вашего EditText? Следующий код дает аккуратную функцию, которую нужно вызвать в методе onCreate () только один раз.

```
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);
            }
        }
}
```

Значок или кнопка внутри Custom Edit Text и его действие и нажмите кнопку прослушивания.

Этот пример поможет редактировать текст с помощью значка в правой части.

> Примечание: в этом я использую setCompoundDrawablesWithIntrinsicBounds, поэтому, если вы хотите изменить положение значка, вы можете достичь этого, используя setCompoundDrawablesWithIntrinsicBounds в 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);
    }
}
```

Если вы хотите изменить сенсорную область, вы можете изменить значения EXTRA_TOUCH_AREA по умолчанию, которые я дал как 50.

И для параметра «Включить кнопку» и «прослушиватель кликов» вы можете позвонить из

своей деятельности или фрагмента, как это,

```
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.

            }
        });
```

Прочитайте Редактировать текст онлайн: https://riptutorial.com/ru/android/topic/5843/редактировать-текст

глава 226: Режим PorterDuff

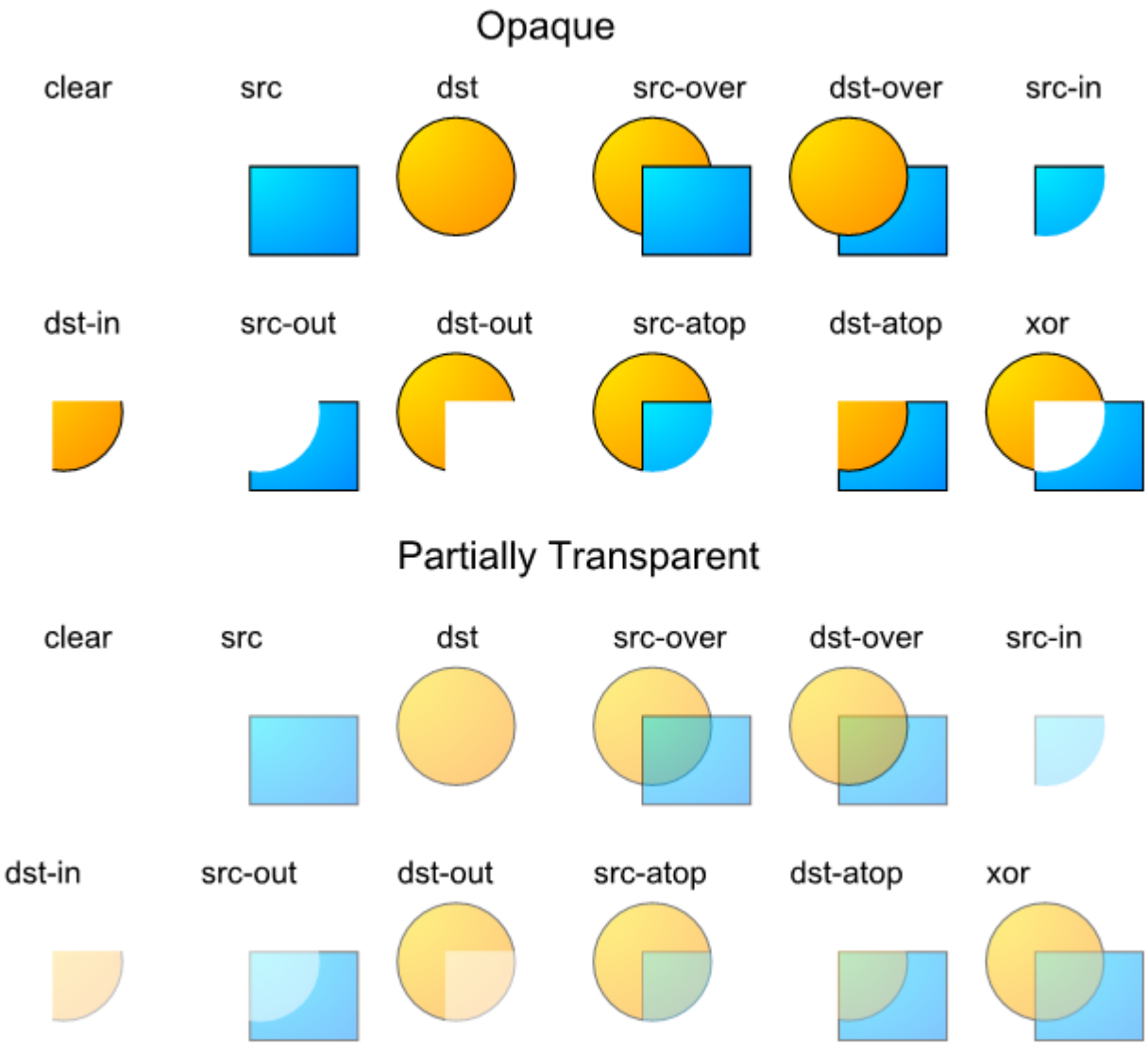
Вступление

PorterDuff описывается как способ комбинирования изображений, как если бы они были «фигурами неправильной формы картона», наложенными друг на друга, а также схемой смешивания перекрывающихся частей

замечания

«Портер Дафф» сам по себе является альфа-композиционной техникой, названной в честь работы Томаса Портера и Тома Даффа.

Подводя итог, техника берет два изображения с альфа-каналом и генерирует выходное изображение, комбинируя значения пикселей двух изображений. Различные режимы комбинирования приводят к разному выходному изображению. Например, на следующем изображении синяя форма (источник, существующие пиксели) объединяется с желтой формой (назначение, новые пиксели) в разных режимах:



Examples

Создание фильтра PorterDuff ColorFilter

`PorterDuff.Mode` используется для создания `PorterDuffColorFilter` . Цветовой фильтр изменяет цвет каждого пикселя визуального ресурса.

```
ColorFilter filter = new PorterDuffColorFilter(Color.BLUE, PorterDuff.Mode.SRC_IN);
```

Вышеуказанный фильтр будет оттенять непрозрачные пиксели до синего цвета.

Цветовой фильтр можно применить к `Drawable` :

```
drawable.setColorFilter(filter);
```

Его можно применять к `ImageView` :

```
imageView.setColorFilter(filter);
```

Кроме того, он может быть применен к `Paint` , так что цвет, который рисуется с использованием этой краски, изменяется фильтром:

```
paint.setColorFilter(filter);
```

Создание PorterDuff XferMode

`Xfermode` (думаю, «передача») работает как шаг передачи в чертежном конвейере. Когда `Xfermode` применяется к `Paint` , пиксели, нарисованные краской, объединяются с базовыми пикселями (уже нарисованными) в соответствии с режимом:

```
paint.setColor(Color.BLUE);
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
```

Теперь у нас есть синяя краска. Любая обрамленная фигура будет оттенять уже существующие, непрозрачные синие пиксели в области формы.

Примените радиальную маску (виньетку) к растровому изображению, используя 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);
}
```

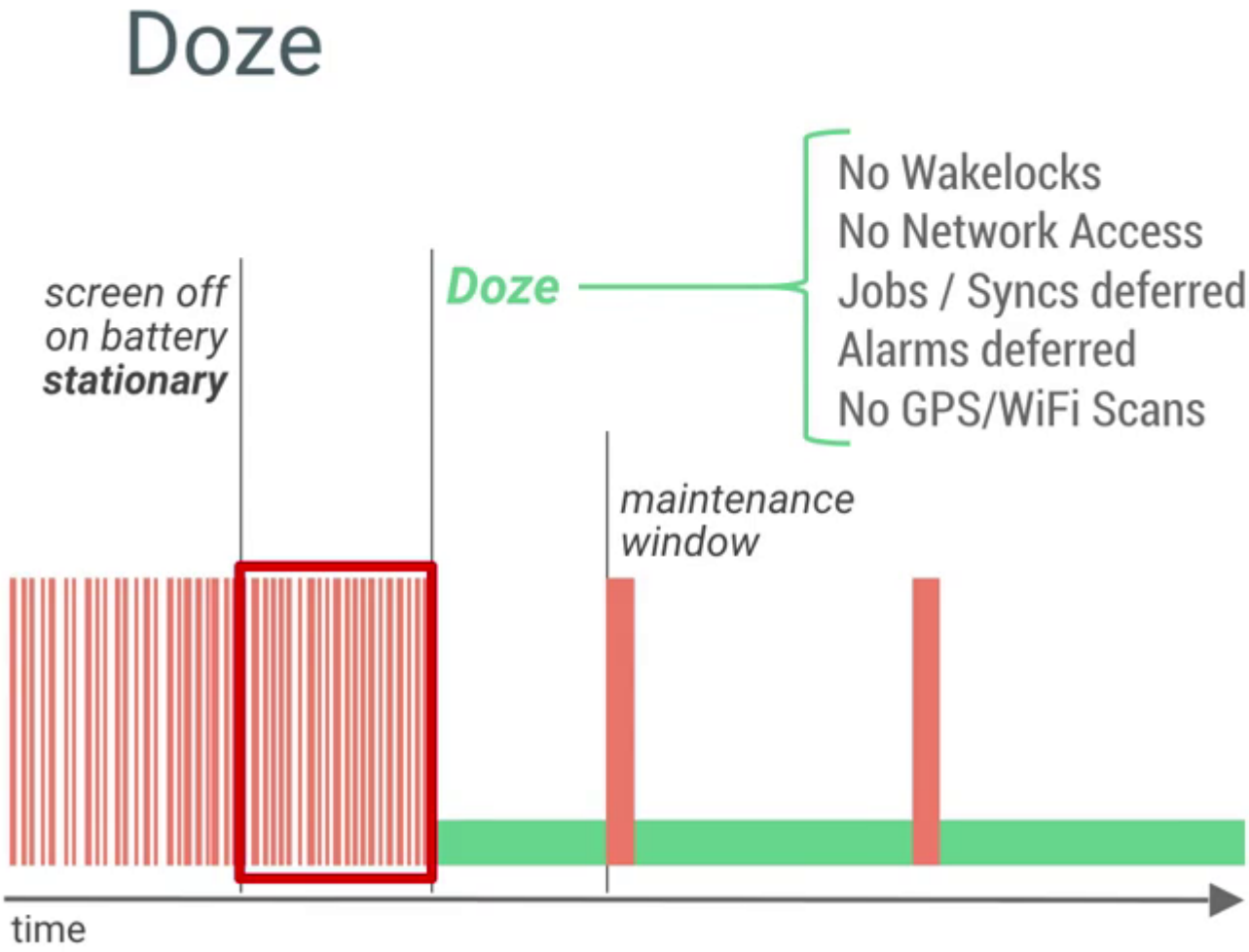
Прочитайте Режим PorterDuff онлайн: https://riptutorial.com/ru/android/topic/377/режим-porterduff

глава 227: Режим дозировки

замечания

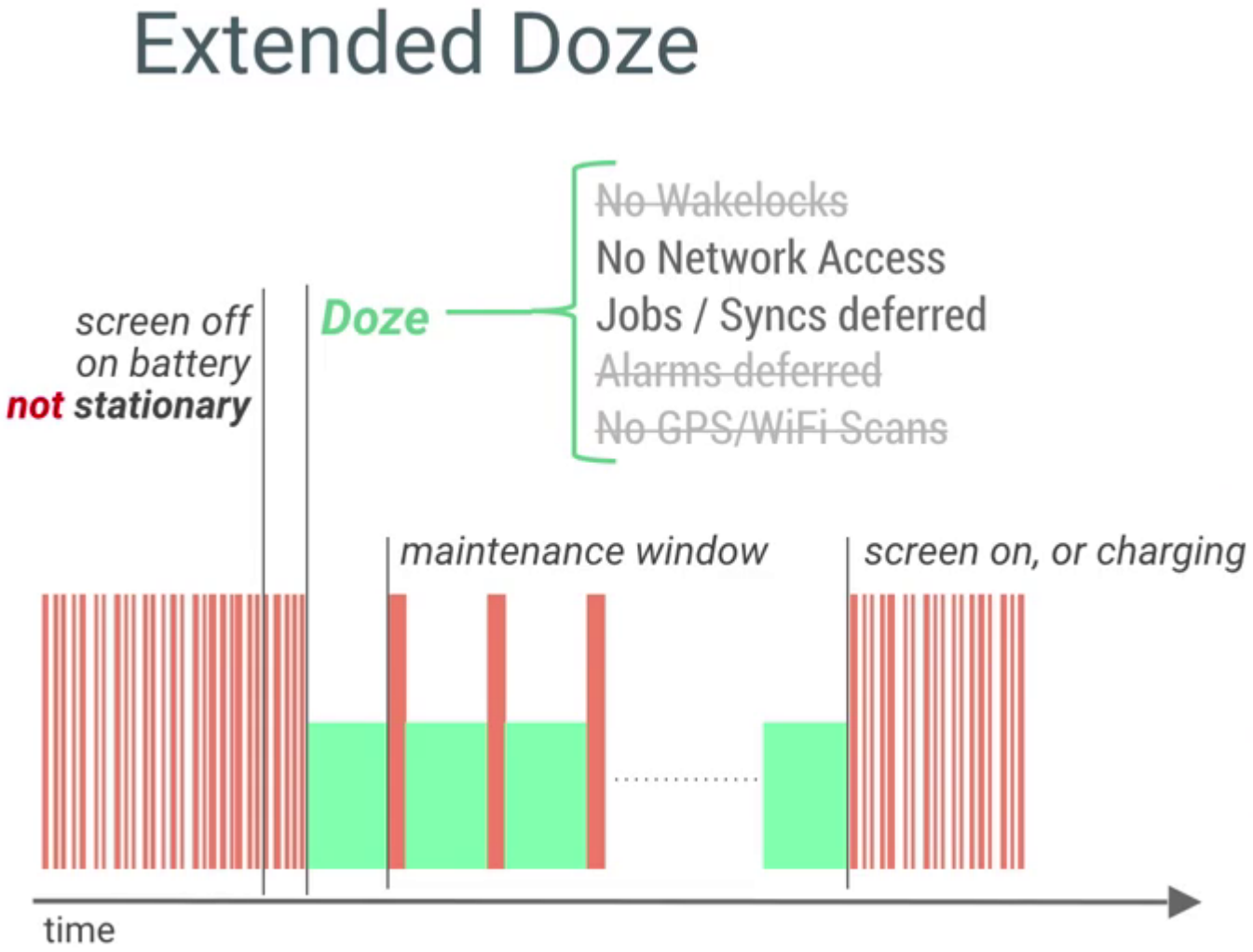
Режим Doze - это набор изменений и правил, которые заставляют ваш телефон спать при простоях.

На Android 6.0 Marshmallow: режим Doz активируется через некоторое время, когда экран выключен, устройство остается неподвижным и работает от аккумулятора.



Как вы можете видеть на диаграмме выше, когда активируется Doze Mode, устройство не получает никаких вакелов, сетевого доступа, заданий / синхронизаций, сигналов тревоги, сканирования GPS / Wi-Fi.

На Android 7.0 Nougat: представьте, если ваш телефон находится в кармане (экран выключен, он работает от батареи, но он не является стационарным), вы также можете получить функции режима Doz Mode, верно? Вот почему Google объявил о расширенном режиме «Доза»: он работает, когда экран выключен, но не стационарный.



Как вы можете видеть на этой диаграмме, отключены только сетевой доступ и задания / синхронизация. Обратите внимание, что Extended Doze не заменяет первый режим Doze. Они работают вместе, в зависимости от состояния телефона (стационарного или нет). Вот отличия:

	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

Разработчики должны знать, что:

- Doze может поддерживать временный wakelock и сетевой доступ для сообщений с высоким приоритетом GCM (Google Cloud Messaging) (для случаев, когда пользователю требуется немедленное уведомление);
- Работа на переднем плане (например, воспроизведение музыки) будет продолжать работать.

Вы можете найти более подробную информацию здесь:
https://developer.android.com/training/monitoring-device-state/doze-standby.html

Examples

Исключение приложения из режима доз

1. Настройки открытого телефона
2. открытая батарея
3. откройте меню и выберите «оптимизация батареи»,
4. в раскрывающемся меню выберите «все приложения»,

5. выберите приложение, в которое хотите добавить белый список
6. выберите "не оптимизировать"

Теперь это приложение будет отображаться в не оптимизированных приложениях.

Приложение может проверить, `isIgnoringBatteryOptimizations()` ли оно в белый список, вызывая `isIgnoringBatteryOptimizations()`

Белый список приложений для Android

Белый список не отключает режим доз для вашего приложения, но вы можете сделать это, используя сетевые блокировки и блокировки ожидания.

Белым списком приложений для Android можно сделать следующее:

```
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);
}
```

Результат запуска вышеуказанного действия может быть подтвержден следующим кодом:

```
@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
        }
    }
}
```

Прочитайте Режим дозировки онлайн: https://riptutorial.com/ru/android/topic/4719/режим-дозировки

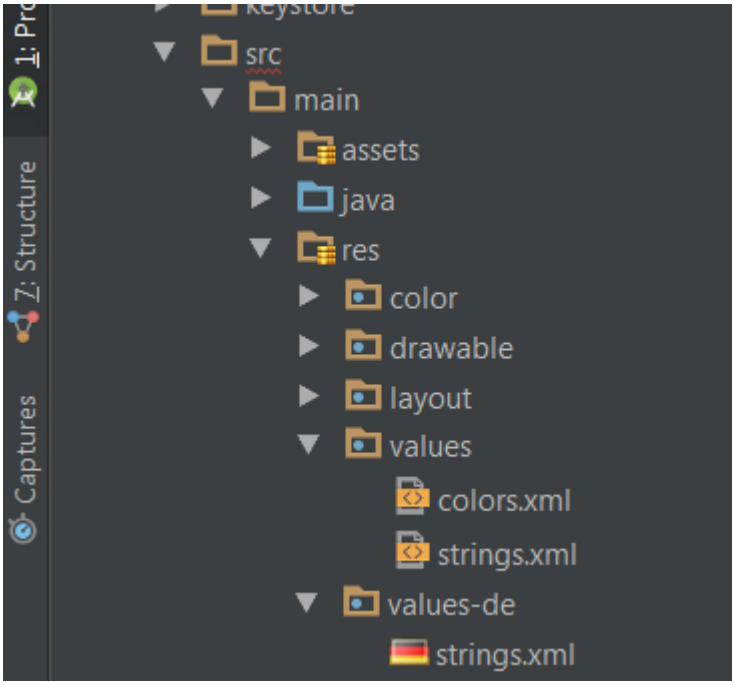
глава 228: Ресурсы

Examples

Перевести строку

Строки могут быть интернационализированы путем определения другого strings.xml для каждого поддерживаемого вами языка.

Вы добавляете новый язык, создавая новый каталог значений с кодом языка ISO в качестве суффикса. Например, при добавлении немецкого набора ваша структура может выглядеть следующим образом:



Когда система ищет запрашиваемую строку, она сначала проверяет специфический для языка xml, если он не найден, возвращается значение из файла strings.xml по умолчанию. Ключ остается неизменным для каждого языка, и изменяется только значение.

Пример содержимого:

/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>
```

Определить строки

Строки обычно хранятся в файле ресурсов `strings.xml` . Они определяются с помощью XML-элемента `<string>` .

Цель strings.xml - разрешить интернационализацию. Вы можете определить strings.xml для каждого языка iso-кода. Таким образом, когда система ищет строку «app_name», она сначала проверяет файл xml, соответствующий текущему языку, и если он не найден, ищет запись в файле strings.xml по умолчанию. Это означает, что вы можете выбрать локализовать некоторые из ваших строк, а не другие.

/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>
```

Как только строка определена в файле ресурсов XML, ее можно использовать другими частями приложения.

Файлы проекта проекта XML могут использовать элемент `<string>` , ссылаясь на `@string/string_name` . Например, файл манифеста приложения (/manifests/AndroidManifest.xml) включает в себя следующую строку по умолчанию в Android Studio:

```
android:label="@string/app_name"
```

Это говорит андроиду искать ресурс `<string>` называемый «имя приложения», для использования в качестве имени для приложения, когда он установлен или отображается в панели запуска.

В другой раз, когда вы используете ресурс `<string>` из файла XML в android, вы будете в файле макета. Например, следующее представляет TextView, который отображает строку `hello_world` мы определили ранее:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world"/>
```

Вы также можете получить доступ к ресурсам `<string>` из части java вашего приложения.

Чтобы отозвать нашу же `hello_world` сверху в классе Activity, используйте:

```
String helloWorld = getString(R.string.hello_world);
```

Определить массив строк

Чтобы определить строковый массив, напишите в файле ресурсов

RES / значения / filename.xml

```
<string-array name="string_array_name">
    <item>text_string</item>
    <item>@string/string_id</item>
</string-array>
```

например

RES / значения / 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>
```

и использовать его из java как

```
String[] strings = getResources().getStringArray(R.array.string_array_example;
Log.i("TAG",Arrays.toString(strings)));
```

Выход

```
I/TAG: [HelloWorld, Hello World!]
```

Определить размеры

Размеры обычно хранятся в именах файлов ресурсов `dimens.xml` . Они определяются с помощью элемента `<dimen>` .

RES / значения / 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>
```

Вы можете использовать разные единицы:

- **sp:** Масштабируемые пиксели. Для шрифтов.
- **dp:** независимые от плотности пиксели. Для всего остального.
- **pt:** Очки
- **px:** пиксели
- **мм:** миллиметры
- **im:** Дюймы

Теперь размеры могут быть `@dimen/name_of_the_dimension` в XML с синтаксисом `@dimen/name_of_the_dimension` .

Например:

```
<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>
```

Определить целые числа

Целые элементы обычно хранятся в файле ресурсов с именем `integers.xml` , но имя файла может быть выбрано произвольно. Каждое целое число определяется с помощью элемента `<integer>` , как показано в следующем файле:

RES / значения / integers.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer name="max">100</integer>
</resources>
```

Теперь целые числа можно ссылаться в XML с синтаксисом `@integer/name_of_the_integer` , как показано в следующем примере:

```
<ProgressBar
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:max="@integer/max"/>
```

Определить целочисленный массив

Чтобы определить целочисленный массив, напишите в файле ресурсов

RES / значения / filename.xml

```
<integer-array name="integer_array_name">
    <item>integer_value</item>
    <item>@integer/integer_id</item>
</integer-array>
```

например

RES / значения / 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>
```

и использовать его из java как

```
int[] values = getResources().getIntArray(R.array.fibo);
Log.i("TAG",Arrays.toString(values)));
```

Выход

```
I/TAG: [0, 1, 1, 2, 3, 5]
```

Определение цветов

Цвета обычно хранятся в файле ресурсов с именем `colors.xml` в `colors.xml /res/values/` .

Они определяются элементами `<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>
```

Цвета представлены шестнадцатеричными значениями цвета для каждого цветового канала `(0 - FF)` в одном из форматов:

- `#RGB`
- `#ARGB`
- `#RRGGBB`
- `#AARRGGBB`

легенда

- Значение A - alpha channel - 0 полностью прозрачно, значение FF непрозрачно
- R - красный канал
- G - зеленый канал
- B - синий канал

Определенные цвета могут использоваться в XML со следующим синтаксисом
`@color/name_of_the_color`

Например:

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/blackOverlay">
```

Использование цветов в коде

Эти примеры предполагают, что `this` ссылка на активность. Ссылка на контекст может использоваться и на своем месте.

1,6

```
int color = ContextCompat.getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

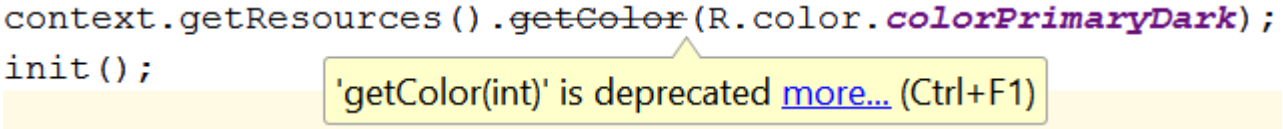
6,0

```
int color = this.getResources().getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

В вышеуказанной декларации `colorPrimary` , `colorPrimaryDark` и `colorAccent` используются для определения цветов дизайна материалов, которые будут использоваться при определении пользовательской темы Android в `styles.xml` . Они автоматически добавляются при создании нового проекта с помощью Android Studio.

Получение ресурсов без «устаревших» предупреждений

Используя Android API 23 или выше, очень часто можно увидеть такую ситуацию:



Эта ситуация вызвана структурными изменениями API Android в отношении получения ресурсов.
Теперь функция:

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

должен быть использован. Но в библиотеке `android.support.v4` есть другое решение.

Добавьте следующую зависимость в файл `build.gradle` :

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

Затем доступны все методы из библиотеки поддержки:

```
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));
```

Кроме того, можно использовать больше методов из библиотеки поддержки:

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

Определите ресурс меню и используйте его внутри Activity / Fragment

Определить меню в **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>
```

Дополнительные параметры конфигурации см. В разделе : Ресурс меню

Внутри **`Activity`** :

```
@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);
}
```

Для вызова методов выше при показе представления вызовите
`getActivity().invalidateOptionsMenu();`

Внутри **`Fragment`** дополнительный вызов:

```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    setHasOptionsMenu(true);
    super.onCreateView(inflater, container, savedInstanceState);
}
```

Форматирование строк в strings.xml

Определение строк в файле strings.xml также позволяет форматировать строки. Единственное предостережение в том, что String нужно будет обрабатывать в коде, как показано ниже, или просто привязывать его к макету.

```
<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);
```

В приведенном выше примере,

% 1 $ s
«%» отделяется от обычных символов,
'1' обозначает первый параметр,
'$' используется как разделитель между номером параметра и типом,
's' обозначает тип строки ('d' используется для целого числа)

Обратите внимание: `getString()` - это метод `Context` или `Resources` , т. `getActivity().getString()` Вы можете использовать его непосредственно в экземпляре `Activity` , иначе вы можете использовать `getActivity().getString()` или `getContext().getString()` соответственно.

Определить список состояний цвета

Списки состояния цвета могут использоваться как цвета, но будут меняться в зависимости от состояния вида, для которого они используются.

Чтобы определить один, создайте файл ресурсов в `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>
```

Элементы оцениваются в том порядке, в котором они определены, и используется первый элемент, чьи указанные состояния соответствуют текущему состоянию представления. Таким образом, хорошая практика заключается в том, чтобы указать конечную точку, без каких-либо определенных селекторов.

Каждый элемент может использовать цветной литерал или ссылаться на цвет, определенный в другом месте.

Определение строковых множеств

Чтобы различать множественные и сингулярные строки, вы можете определить множественное число в файле *strings.xml* и указать разные количества, как показано в следующем примере:

```
<?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>
```

Это определение можно получить из кода Java с помощью `getQuantityString()` класса `Resources` , как показано в следующем примере:

```
getResources().getQuantityString(R.plurals.hello_people, 3, 3);
```

Здесь первый параметр `R.plurals.hello_people` - это имя ресурса. Второй параметр (`3` в этом примере) используется для выбора правильной строки `quantity` . Третий параметр (также `3` в этом примере) - это аргумент формата, который будет использоваться для замены спецификатора формата `%d` .

Возможные значения количества (перечислены в алфавитном порядке):

```
few
many
one
other
two
zero
```

Важно отметить, что не все локали поддерживают каждую номинал `quantity` . Например, китайский язык не имеет понятия об `one` элементе. У английского языка нет `zero` элемента, так как он грамматически совпадает с `other` . Неподдерживаемые экземпляры `quantity` будут помечены IDE как предупреждения Lint, но не будут вызывать ошибки сложности, если они используются.

Импортировать массив объектов, определенных в ресурсах

Бывают случаи, когда пользовательские объекты должны быть созданы и определены в ресурсах приложения. Такие объекты могут состоять из простых типов `Java` , например `Integer` , `Float` , `String` .

Ниже приведен пример того, как импортировать объект, определенный в ресурсах приложения. Объект `Category` 3 свойства категории:

- Я БЫ
- цвет
- название

Этот `POJO` имеет эквивалент в файле `categories.xml` , где каждый из массива имеет те же свойства, которые определены для каждой категории.

1. **Создайте модель для объекта:**

```
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. **Создайте файл в папке `res/values` :**

```
categories.xml
```

3. **Составьте каждую модель, состоящую из ресурсов:**

```
<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. **Определить массив в файле ресурсов:**

```
<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. **Создайте функцию для их импорта:**

```
@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 патчей

9 Патч - это **растяжимые** изображения, в которых области, которые могут быть растянуты, определяются черными маркерами на прозрачной границе.

Существует большой учебник здесь .
Несмотря на то, что он настолько стар, он по-прежнему так ценен, и он помог многим из нас глубоко понять 9 патч-экипировку.

К сожалению, недавно эта страница была отложена на некоторое время (сейчас она снова включена).

Следовательно, необходимо иметь физическую копию этой страницы для разработчиков Android на наших надежных серверах.

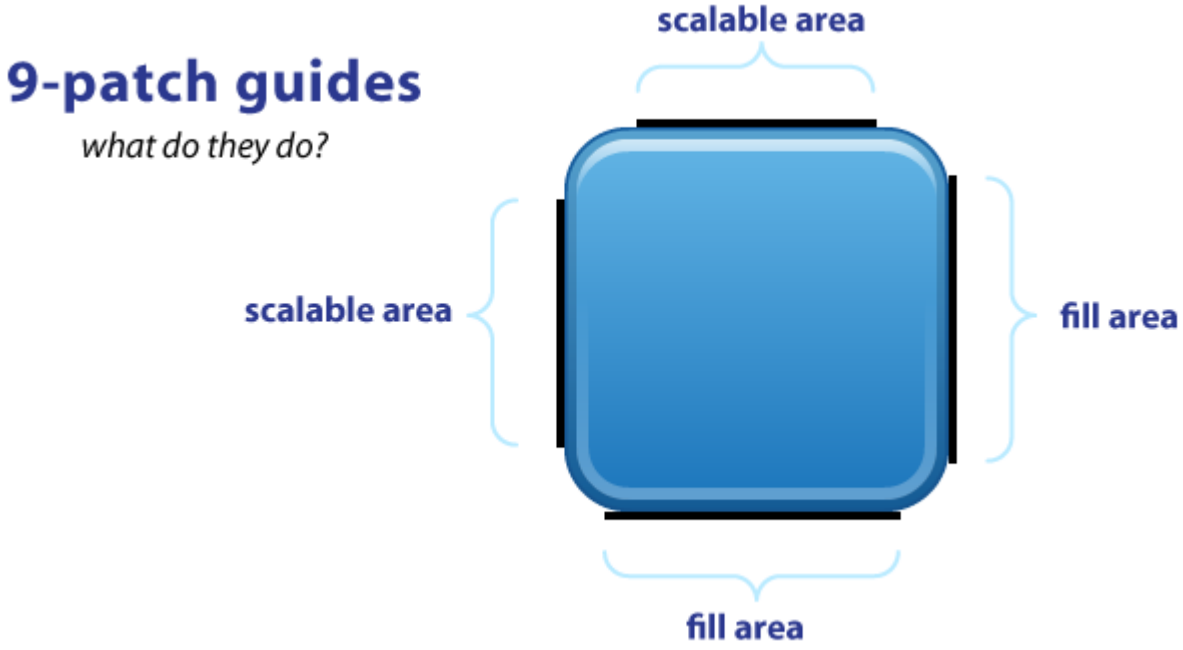
Вот.

ПРОСТОЕ РУКОВОДСТВО ДЛЯ 9-PATCH ДЛЯ ANDROID UI 18 мая 2011 г.

Пока я работал над своим первым Android-приложением, я нашел 9-patch (aka 9.png), чтобы запутать и плохо документировать. Через некоторое время я наконец понял, как это работает, и решил собрать что-то, чтобы помочь другим понять это.

В принципе, 9-patch использует прозрачность png, чтобы сделать расширенную форму 9-среза или scale9. Направляющие - прямые, 1-пиксельные черные линии, нарисованные на краю вашего изображения, которые определяют масштабирование и заливку вашего изображения. Назвав ваш файл изображения name.9.png, Android распознает формат 9. png и использует черные направляющие для масштабирования и заполнения ваших растровых изображений.

Вот базовая карта:



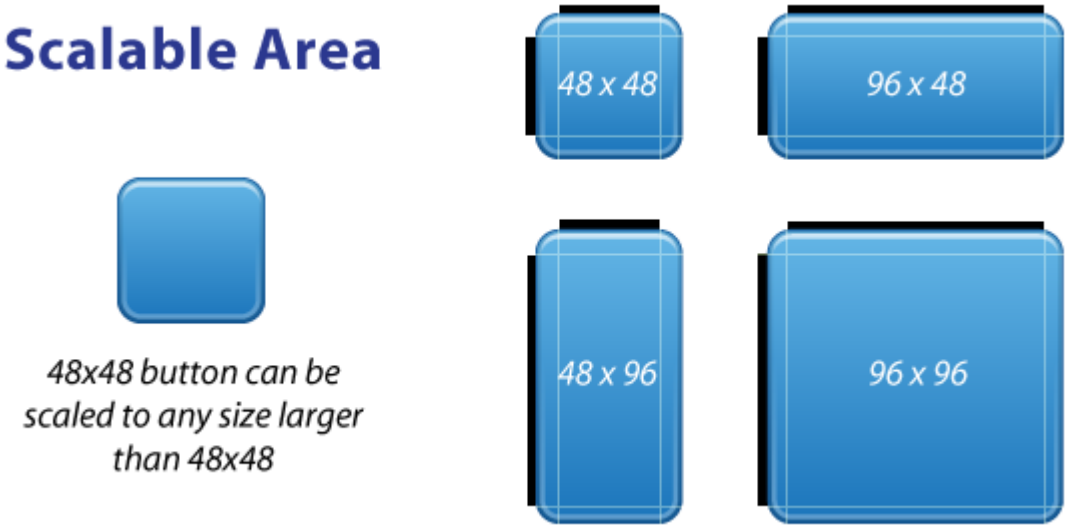
Как вы можете видеть, у вас есть руководства по каждой стороне вашего изображения. Направляющие TOP и LEFT предназначены для масштабирования вашего изображения (например, 9-среза), в то время как направляющие RIGHT и BOTTOM определяют область заполнения.

Черные направляющие линии отключаются / удаляются из вашего изображения - они не будут отображаться в приложении. Гиды должны быть только одного пикселя, поэтому, если вы хотите кнопку 48 × 48, ваш png будет фактически 50 × 50. Все, что толще, чем один пиксель, останется частью вашего изображения. (Мои примеры имеют направляющие шириной 4 пикселя для лучшей видимости. Они должны быть действительно только 1 пикселем).

Ваши направляющие должны быть сплошными черными (# 000000). Даже небольшое различие в цвете (# 000001) или альфа приведет к сбою и нормальному растяжению. Этот провал не будет очевидным *, он терпит неудачу! Да. В самом деле. Теперь ты знаешь.

Также вы должны иметь в виду, что оставшаяся область однопиксельного контура должна быть полностью прозрачной. Это включает в себя четыре угла изображения - они всегда должны быть четкими. Это может быть большой проблемой, чем вы понимаете. Например, если вы масштабируете изображение в Photoshop, оно добавит сглаженные пиксели, которые могут содержать почти невидимые пиксели, которые также могут привести к его сбою *. Если вы хотите масштабировать в Photoshop, используйте параметр Nearest Neighbor в раскрывающемся меню Resample Image (в нижней части всплывающего меню «Размер изображения»), чтобы сохранить острые края ваших направляющих.

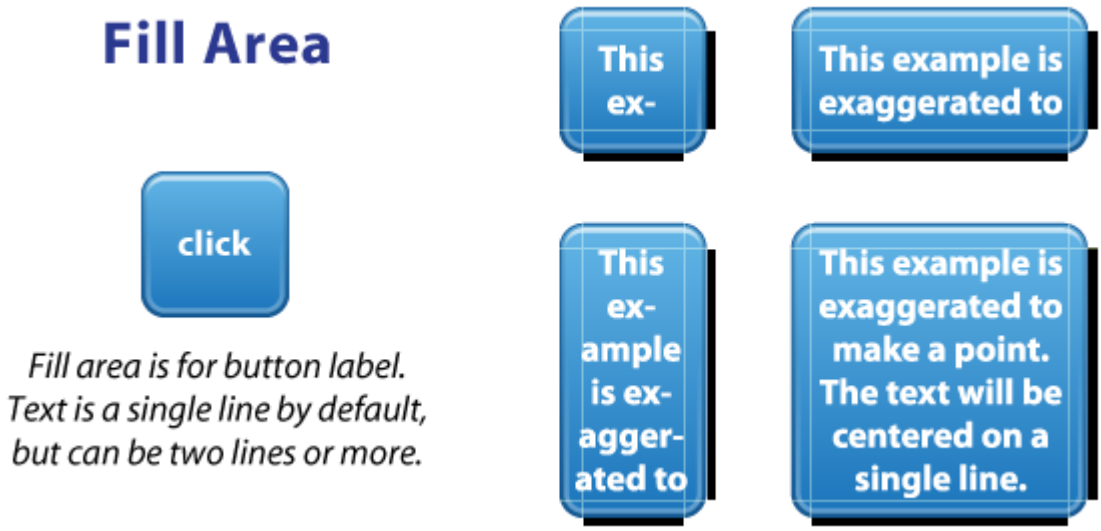
* (обновлено 1/2012) Это фактически «исправление» в последнем наборе для разработчиков. Раньше это проявлялось бы, когда все ваши образы и ресурсы внезапно ломались, а не фактически сломанное изображение с 9 патчами.



Направляющие TOP и LEFT используются для определения масштабируемой части вашего изображения - LEFT для масштабирования высоты, TOP для масштабирования ширины. Используя образ кнопки в качестве примера, это означает, что кнопка может растягиваться горизонтально и вертикально внутри черной части, а все остальное, например, углы, будет оставаться того же размера. Позволяет иметь кнопки, которые могут масштабироваться до любого размера и поддерживать единообразный внешний вид.

Важно отметить, что 9-патч-изображения не уменьшаются - они только расширяются. Поэтому лучше всего начинать как можно меньше.

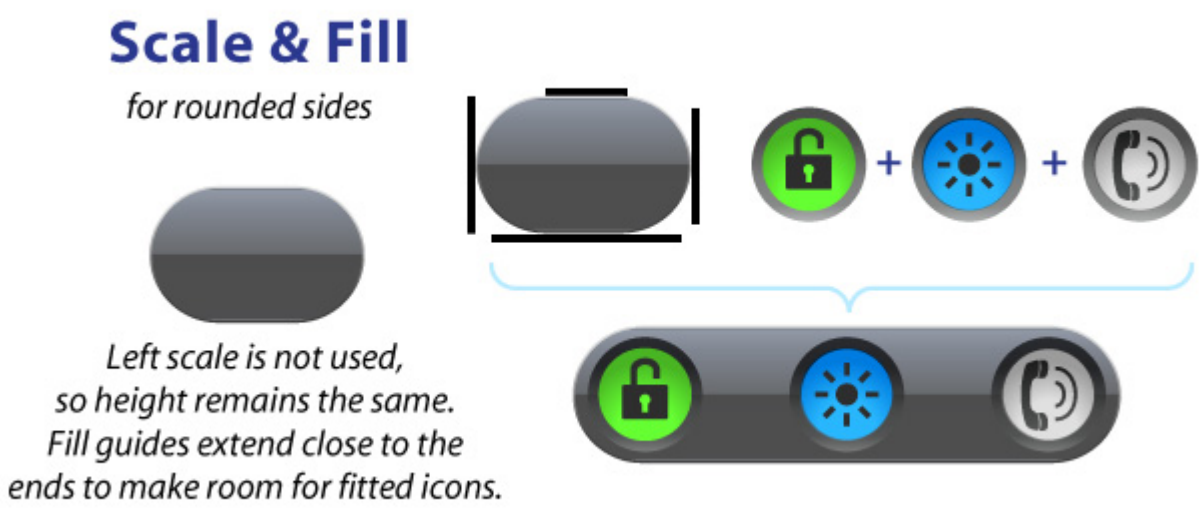
Кроме того, вы можете оставить части в середине линии шкалы. Например, если у вас есть кнопка с резким глянцевым краем по середине, вы можете оставить несколько пикселей в середине руководства LEFT. Центральная горизонтальная ось вашего изображения не будет масштабироваться, только части выше и ниже нее, поэтому ваш острый блеск не будет получать сглаживание или нечеткость.



Гиды области заполнения являются необязательными и обеспечивают способ определения области для таких вещей, как текстовая метка. Заполнение определяет, сколько места в вашем изображении помещается для размещения текста или значка или других вещей. 9-патч предназначен не только для кнопок, но и для фоновых изображений.

Приведенный выше пример кнопки & label преувеличен просто для объяснения идеи заполнения - ярлык не совсем точен. Честно говоря, я не знал, как Android делает многострочные метки, поскольку ярлык кнопки обычно представляет собой одну строку текста.

Наконец, вот хорошая демонстрация того, как гиды масштабирования и заливки могут меняться, например LinearLayout с фоновым изображением и полностью закругленными сторонами:



В этом примере руководство LEFT не используется, но мы по-прежнему должны иметь руководство. Фоновое изображение не масштабируется вертикально; он просто масштабируется горизонтально (на основе TOP-ориентира). Глядя на направляющие для заполнения, направляющие RIGHT и BOTTOM выходят за пределы, где они соответствуют изогнутым краям изображения. Это позволяет мне размещать круглые кнопки близко к краям фона для плотного, подогнанного вида.

Итак, это все. 9-patch очень просто, как только вы его получите. Это не идеальный способ масштабирования, но масштабные и многострочные шкалы-наборы обеспечивают большую гибкость, чем традиционный 9-срез и масштабирование9. Попробуйте, и вы быстро это выясните.

Уровень прозрачности (альфа)

Значения непрозрачности шестнадцатеричных

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

```
|   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        |
 -----------------------------
```

Если вы хотите установить 45% на красный цвет.

```
<color name="red_with_alpha_45">#73FF0000</color>
```

шестнадцатеричное значение для красного цвета - # FF0000

Вы можете добавить 73 для непрозрачности 45% в префиксе - # 73FF0000

Работа с файлом strings.xml

Строковый ресурс предоставляет текстовые строки для вашего приложения с дополнительным стилем текста и форматированием. Существует три типа ресурсов, которые могут предоставить вашему приложению строки:

строка

```
XML resource that provides a single string.
```

Синтаксис:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="string_name">text_string</string>
</resources>
```

И использовать эту строку в макете:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/string_name" />
```

Строковый массив

```
XML resource that provides an array of strings.
```

Синтаксис:

```
    <resources>
<string-array name="planets_array">
    <item>Mercury</item>
    <item>Venus</item>
    <item>Earth</item>
    <item>Mars</item>
</string-array>
```

использование

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

Количество строк (плоских)

```
XML resource that carries different strings for pluralization.
```

Синтаксис:

```
<?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>
```

Использование:

```
int count = getNumberOfsongsAvailable();
Resources res = getResources();
String songsFound = res.getQuantityString(R.plurals.plural_name, count, count);
```

Прочитайте Ресурсы онлайн: https://riptutorial.com/ru/android/topic/108/ресурсы

глава 229: Сжатие изображения

Examples

Как сжимать изображение без изменения размера

Get **Compressed Bitmap** из класса 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;
    }
}
```

Размеры одинаковы после сжатия Bitmap .

Как я проверил?

```
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());
```

Выход:

```
Before Compress : Dimension: 1080-1452
After Compress : Dimension: 1080-1452
```

Прочитайте Сжатие изображения онлайн: https://riptutorial.com/ru/android/topic/5588/сжатие-изображения

глава 230: Синхронизация данных с адаптером синхронизации

Examples

Адаптер Dummy Sync с поддержкой Stub

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");
    }

}
```

Служба синхронизации

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

Authenticator

```
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();
    }
}
```

Служба аутентификации

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

Добавления 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" />
```

Рез / 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" />
```

Рез / 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;
    }
}
```

Вызовите эту функцию при успешном входе в систему, чтобы создать учетную запись с зарегистрированным идентификатором пользователя

```
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;
    }
```

Принуждение синхронизации

```
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);
```

Прочитайте Синхронизация данных с адаптером синхронизации онлайн: https://riptutorial.com/ru/android/topic/1944/синхронизация-данных-с-адаптером-синхронизации

глава 231: скольжение

Вступление

**** ПРЕДУПРЕЖДЕНИЕ Эта документация не поддерживается и часто неточна ****

Официальная документация Glide - гораздо лучший источник:

Для Glide v4 см. Http://bumptech.github.io/glide/ . Для Glide v3 см. Https://github.com/bumptech/glide/wiki .

замечания

Glide - это быстрый и эффективный механизм управления медиафайлами и изображений для Android с открытым исходным кодом, который включает в себя декодирование носителей, кеширование памяти и дисков, а также объединение ресурсов в простой и удобный интерфейс.

Glide поддерживает выборку, декодирование и отображение видеопотоков, изображений и анимированных GIF-файлов. Glide включает гибкий API, который позволяет разработчикам подключаться практически к любому сетевому стеклу.
По умолчанию Glide использует собственный стек, основанный на HttpUrlConnection , но также включает в себя библиотеки утилиты, подключаемые к проекту Volley Google или библиотеке OkHttp Square .

Основное внимание Glide уделяется тому, чтобы прокручивать любой вид списка изображений как можно более гладко и быстро, но Glide также эффективен практически для любого случая, когда вам нужно выбирать, изменять размер и отображать удаленное изображение.

Исходный код и дальнейшая документация доступны на GitHub:
https://github.com/bumptech/glide

Examples

Добавьте Glide в свой проект

Из официальной документации :

С 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'
}
```

С 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>
```

В зависимости от конфигурации и использования ProGuard (DexGuard) вам также может потребоваться включить следующие строки в ваш proguard.cfg (см. Дополнительную информацию о Glide's wiki):

```
-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
```

Загрузка изображения

ImageView

Чтобы загрузить изображение с указанного URL, Uri, идентификатор ресурса или любую другую модель в `ImageView` :

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
String yourUrl = "http://www.yoururl.com/image.png";

Glide.with(context)
    .load(yourUrl)
```

```
    .into(imageView);
```

Для Uris замените `yourUrl` своим Uri (`content://media/external/images/1`). Для Drawables замените `yourUrl` вашим идентификатором ресурса (`R.drawable.image`).

RecyclerView и ListView

В ListView или RecyclerView вы можете использовать точно такие же строки:

```
@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);
}
```

Если вы не хотите запускать загрузку в `onBindViewHolder` , убедитесь, что вы `onBindViewHolder` `clear()` любой `ImageView` Glide, который может управлять до изменения `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);
    }
}
```

Преобразование кругового круга (Загрузка изображения в круговом ImageView)

Создайте круглое изображение со скольжением.

```
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();
    }
}
```

Использование:

```
Glide.with(context)
    .load(yourimageurl)
    .transform(new CircleTransform(context))
    .into(userImageView);
```

Преобразования по умолчанию

Glide включает в себя два преобразования по умолчанию, по центру и центру.

Подходящий центр:

```
Glide.with(context)
    .load(yourUrl)
    .fitCenter()
    .into(yourView);
```

Fit center выполняет те же преобразования, что и ScaleType Android. FIT_CENTER .

Центровая культура:

```
Glide.with(context)
    .load(yourUrl)
```

```
    .centerCrop()
    .into(yourView);
```

Центральная обрезка выполняет ту же трансформацию, что и Android ScaleType.CENTER_CROP .

Для получения дополнительной информации см. Wiki в Glide .

Изображение с закругленными углами с закругленными углами с пользовательской целью Glide

Сначала создайте класс утилиты или используйте этот метод в классе

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

Загрузка изображения:

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Поскольку вы используете asBitmap (), анимация будет удалена, хотя. Вы можете использовать свою собственную анимацию в этом месте, используя метод animate ().

Пример с аналогичной затухающей анимацией Glide по умолчанию.

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .animate(R.anim.abc_fade_in)
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Обратите внимание, что эта анимация - это вспомогательный ресурс библиотеки поддержки - она не рекомендуется использовать, поскольку она может быть изменена или даже удалена.

Обратите внимание, что вам также необходимо иметь библиотеку поддержки для

использования RoundedBitmapDrawableFactory

Предварительная загрузка изображений

Чтобы предварительно загрузить удаленные изображения и убедитесь, что изображение загружается только один раз:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE)
    .preload();
```

Затем:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE) // ALL works here too
    .into(imageView);
```

Чтобы предварительно загрузить локальные изображения и убедитесь, что преобразованная копия находится в кеше диска (и, возможно, в кеше памяти):

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // Or whatever transformation you want
    .preload(200, 200); // Or whatever width and height you want
```

Затем:

```
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);
```

Обработка заполнителя и ошибок

Если вы хотите добавить Drawable, который будет отображаться во время загрузки, вы можете добавить местозаполнитель:

```
Glide.with(context)
    .load(yourUrl)
    .placeholder(R.drawable.placeholder)
    .into(imageView);
```

Если вы хотите, чтобы Drawable отображался, если сбой по какой-либо причине:

```
Glide.with(context)
    .load(yourUrl)
    .error(R.drawable.error)
```

```
    .into(imageView);
```

Если вы хотите, чтобы Drawable отображался, если вы предоставили нулевую модель (URL, Uri, путь к файлу и т. Д.):

```
Glide.with(context)
    .load(maybeNullUrl)
    .fallback(R.drawable.fallback)
    .into(imageView);
```

Загрузить изображение в круговом ImageView без пользовательских преобразований

Создайте собственный BitmapImageViewTarget, чтобы загрузить изображение:

```
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);
    }
}
```

Использование:

```
Glide
    .with(context)
    .load(yourimageidentifier)
    .asBitmap()
    .into(new CircularBitmapImageViewTarget(context, imageView));
```

Не удалось загрузить загрузку изображения Glide

```
Glide
        .with(context)
        .load(currentUrl)
        .into(new BitmapImageViewTarget(profilePicture) {
    @Override
    protected void setResource(Bitmap resource) {
```

```
        RoundedBitmapDrawable circularBitmapDrawable =
                RoundedBitmapDrawableFactory.create(context.getResources(), resource);
        circularBitmapDrawable.setCornerRadius(radius);
        imageView.setImageDrawable(circularBitmapDrawable);
    }

    @Override
    public void onLoadFailed(@NonNull Exception e, Drawable errorDrawable) {
        super.onLoadFailed(e, SET_YOUR_DEFAULT_IMAGE);
        Log.e(TAG, e.getMessage(), e);
    }
});
```

Здесь, в `SET_YOUR_DEFAULT_IMAGE` вы можете установить любой по умолчанию `Drawable` . Это изображение будет показано, если загрузка изображения не удалась.

Прочитайте скольжение онлайн: https://riptutorial.com/ru/android/topic/1091/скольжение

глава 232: Создание класса Singleton для сообщения Toast

Вступление

Тост-сообщения - это самый простой способ предоставления обратной связи пользователю. По умолчанию Android предоставляет сообщение с тонами grey color, где мы можем установить сообщение и продолжительность сообщения. Если нам нужно создать более настраиваемое и многоразовое тост-сообщение, мы можем реализовать его самостоятельно с использованием настраиваемого макета. Что еще более важно, когда мы его реализуем, использование шаблона проектирования Singelton упростит поддержание и развитие пользовательского класса сообщений toast.

Синтаксис

- Toast Toast (Контекст Contex)
- void setDuration (int duration)
- void setGravity (int gravity, int xOffset, int yOffset)
- void setView (Просмотреть представление)
- void show ()

параметры

параметр	подробности
контекст	Соответствующий контекст, который должен отображать ваше сообщение тоста. Если вы используете это в ключевом слове « this» или «Использовать в пропуске» как «getActivity ()».
Посмотреть	Создайте собственное представление и передайте этот объект просмотра.
сила тяжести	Пропустите гравитационное положение тостера. Вся позиция добавлена под классом Gravity как статические переменные. Наиболее распространенными позициями являются Gravity.TOP, Gravity.BOTTOM, Gravity.LEFT, Gravity.RIGHT.
xOffset	Горизонтальное смещение тоста.
YOffset	Вертикальное смещение тоста.

параметр	подробности
продолжительность	Продолжительность тоста. Мы можем установить либо Toast.LENGTH_SHORT, либо Toast.LENGTH_LONG

замечания

Тост-сообщение - это простой способ предоставления обратной связи пользователю о том, что происходит. Если вам нужен более продвинутый способ дать отзыв, вы можете использовать диалоги или закусочную.

Чтобы получить более подробную информацию об этом сообщении, пожалуйста, проверьте эту документацию. https://developer.android.com/reference/android/widget/Toast.html

Examples

Создайте собственный одноэлементный класс для тостов

Вот как создать свой собственный одноэлементный класс для тостов, если ваше приложение должно показывать успех, предупреждение и сообщения об опасности для разных случаев использования, вы можете использовать этот класс после того, как вы изменили его по своим собственным спецификациям.

```
    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();
        }
    }
```

Прочитайте Создание класса Singleton для сообщения Toast онлайн:
https://riptutorial.com/ru/android/topic/10843/создание-класса-singleton-для-сообщения-toast

глава 233: Создание обратных совместимых приложений

Examples

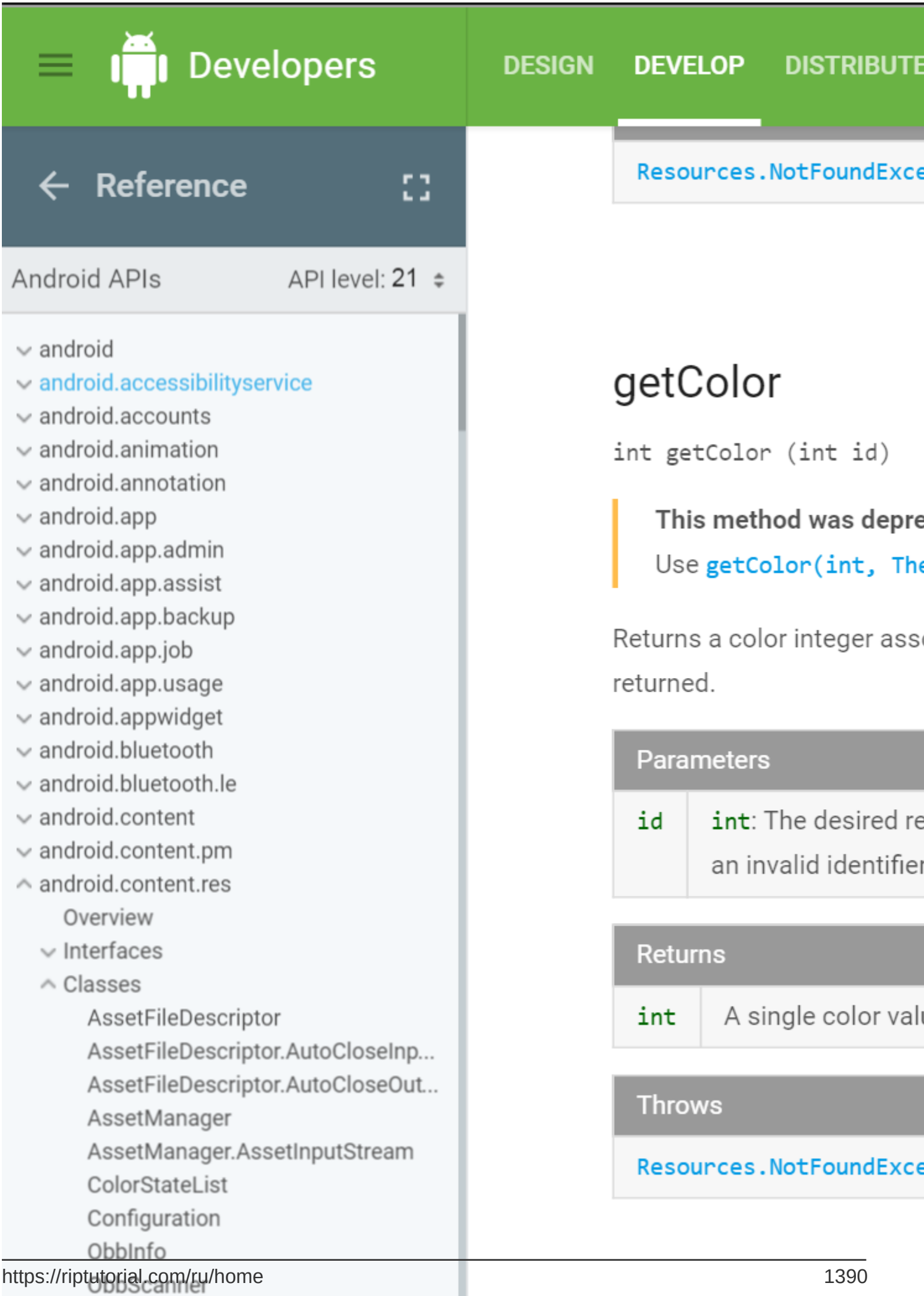
Как обращаться с устаревшим API

Маловероятно, что разработчик не столкнется с устаревшим API в процессе разработки. Устаревший программный элемент - это тот, который программистам не рекомендуется использовать, как правило, потому, что это опасно, или потому, что существует лучшая альтернатива. Компиляторы и анализаторы (например, LINT) предупреждают, когда устаревший программный элемент используется или переопределяется в устаревшем коде.

Устаревший API обычно идентифицируется в Android Studio с помощью аута. В приведенном ниже примере метод `.getColor(int id)` устарел:

```
getResources().~~getColor~~(R.color.colorAccent));
```

По возможности разработчикам рекомендуется использовать альтернативные API и элементы. Можно проверить обратную совместимость библиотеки, посетив документацию по Android для библиотеки и проверив раздел «Добавлен в уровень API x»:


Developers
DESIGN
DEVELOP
DISTRIBUTE
Reference
Android APIs
API level: 21
android
android.accessibilityservice
android.accounts
android.animation
android.annotation
android.app
android.app.admin
android.app.assist
android.app.backup
android.app.job
android.app.usage
android.appwidget
android.bluetooth
android.bluetooth.le
android.content
android.content.pm
android.content.res
Overview
Interfaces
Classes
AssetFileDescriptor
AssetFileDescriptor.AutoCloseInp...
AssetFileDescriptor.AutoCloseOut...
AssetManager
AssetManager.AssetInputStream
ColorStateList
Configuration
ObbInfo
ObbScanner
Resources.NotFoundExce
getColor
int getColor (int id)
This method was depre
Use getColor(int, The
Returns a color integer ass
returned.
Parameters
id
int: The desired re
an invalid identifier
Returns
int
A single color valu
Throws
Resources.NotFoundExce

которую используют ваши пользователи, перед использованием этой библиотеки вы должны проверить уровень API для пользователя. Например:

```
//Checks the API level of the running device
if (Build.VERSION.SDK_INT < 23) {
    //use for backwards compatibility with API levels below 23
    int color = getResources().getColor(R.color.colorPrimary);
} else {
    int color = getResources().getColor(R.color.colorPrimary, getActivity().getTheme());
}
```

Использование этого метода гарантирует, что ваше приложение будет оставаться совместимым с новыми версиями Android, а также с существующими версиями.

Простая альтернатива: используйте библиотеку поддержки

Если используются библиотеки поддержки, часто есть статические вспомогательные методы для выполнения одной и той же задачи с меньшим количеством клиентского кода. Вместо блока if / else выше просто используйте:

```
final int color = android.support.v4.content.ContextCompat
    .getColor(context, R.color.colorPrimary);
```

Большинство устаревших методов, которые имеют более новые методы с другой подписью и многие новые функции, которые, возможно, не могли быть использованы в старых версиях, имеют вспомогательные вспомогательные методы. Чтобы найти других, просмотрите библиотеку поддержки для таких классов, как `ContextCompat` , `ViewCompat` и т. Д.

Прочитайте Создание обратных совместимых приложений онлайн:
https://riptutorial.com/ru/android/topic/4291/создание-обратных-совместимых-приложений

глава 234: Создание окна Overlay (всегда на вершине) Windows

Examples

Всплывающее наложение

Чтобы разместить ваше представление поверх каждого приложения, вы должны назначить свое представление соответствующему оконному менеджеру. Для этого вам нужно разрешение системного предупреждения, которое можно запросить, добавив следующую строку в файл манифеста:

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

Примечание. Если ваше приложение будет уничтожено, ваше представление будет удалено из диспетчера окон. Поэтому лучше создать представление и назначить его оконному менеджеру с помощью функции переднего плана.

Назначение окна WindowManager

Вы можете получить экземпляр диспетчера окон следующим образом:

```
WindowManager mWindowManager = (WindowManager)
mContext.getSystemService(Context.WINDOW_SERVICE);
```

Чтобы определить позицию вашего представления, вам необходимо создать некоторые параметры макета следующим образом:

```
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;
```

Теперь вы можете назначить свое представление вместе с созданными параметрами макета для экземпляра диспетчера окон следующим образом:

```
mWindowManager.addView(yourView, mLayoutParams);
```

Вуаля! Ваше представление было успешно размещено поверх всех других приложений.

Примечание. Вы видите, что вы не будете помещены поверх клавиатуры.

Предоставление SYSTEM_ALERT_WINDOW Разрешения на Android 6.0 и выше

От android 6.0 это разрешение должно предоставлять динамически,

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

Бросок ниже разрешения отклонил ошибку на 6.0,

```
Caused by: android.view.WindowManager$BadTokenException: Unable to add window
android.view.ViewRootImpl$W@86fb55b -- permission denied for this window type
```

Решение :-

Запрос разрешения Overlay, как показано ниже,

```
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);
}
```

Проверьте результат,

```
@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...
        }
    }
}
```

Прочитайте Создание окна Overlay (всегда на вершине) Windows онлайн: https://riptutorial.com/ru/android/topic/6214/создание-окна-overlay--всегда-на-вершине--windows

глава 235: Создание пользовательских ПЗУ Android

Examples

Создание вашей машины для строительства!

Прежде чем вы сможете что-либо построить, вы должны подготовить машину к строительству. Для этого вам нужно установить множество библиотек и модулей. Наиболее рекомендуемым дистрибутивом Linux является Ubuntu, поэтому в этом примере основное внимание будет уделено установке всего, что необходимо на Ubuntu.

Установка Java

Сначала добавьте следующий Архив личных пакетов (PPA): `sudo apt-add-repository ppa:openjdk-r/ppa` .

Затем обновите источники, выполнив: `sudo apt-get update` .

Установка дополнительных зависимостей

Все необходимые дополнительные зависимости могут быть установлены с помощью следующей команды:

```
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
```

Подготовка системы для разработки

Теперь, когда все зависимости установлены, давайте подготовим систему для разработки, выполнив:

```
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
```

Наконец, давайте настроим кеш и репо следующими командами:

```
sudo install utils/repo /usr/bin/
sudo install utils/ccache /usr/bin/
```

Обратите внимание: мы также можем достичь этой настройки, запустив автоматические скрипты, сделанные Akhil Narang (*akhilnarang*), одним из поддерживающих OS Resurrection Remix . Эти скрипты можно найти на GitHub .

Прочитайте Создание пользовательских ПЗУ Android онлайн:
https://riptutorial.com/ru/android/topic/9212/создание-пользовательских-пзу-android

глава 236: Создание пользовательских представлений

Examples

Создание пользовательских представлений

Если вам требуется полностью настроенное представление, вам нужно подклассировать `View` (суперкласс всех Android-представлений) и предоставить свой собственный размер (`onMeasure(...)`) и `onDraw(...)` рисования (`onDraw(...)`):

1. **Создайте свой собственный скелет вида:** это в основном то же самое для каждого пользовательского представления. Здесь мы создаем скелет для пользовательского представления, которое может нарисовать смайлик под названием `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. **Инициализация ваших красок:** объекты `Paint` - это кисти вашего виртуального холста, определяющие, как визуализируются ваши геометрические объекты (например, цвет, стиль заливки и штрихов и т. Д.). Здесь мы создаем две `Paint` s, одну желтую заполненную краску для круга и одну черную краску для глаз и рта:

```
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. **Внедрите свой собственный `onMeasure(...)` :** это необходимо, чтобы родительские макеты (например, `FrameLayout`) могли правильно выровнять пользовательский вид. Она предоставляет набор `measureSpecs` , которые можно использовать , чтобы определить высоту и ширину вашего вида. Здесь мы создаем квадрат, убедившись, что высота и ширина одинаковы:

```
@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);
}
```

   Обратите внимание, что `onMeasure(...)` должен содержать хотя бы один вызов `setMeasuredDimension(..)` иначе ваше пользовательское представление будет сбой с помощью `IllegalStateException` .

4. **Внедрите собственный `onSizeChanged(...)` :** это позволяет вам `onSizeChanged(...)` текущую высоту и ширину вашего пользовательского представления для правильной настройки вашего кода рендеринга. Здесь мы просто вычислим наш центр и наш радиус:

```
@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. **Внедрите собственный `onDraw(...)` :** здесь вы реализуете фактический рендеринг своего представления. Он предоставляет объект `Canvas` который вы можете рисовать (см. Официальную документацию `Canvas` для всех доступных доступных методов рисования).

```
@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. **Добавьте свой собственный вид в макет:** теперь пользовательский вид может быть включен в любые файлы макетов, которые у вас есть. Здесь мы просто обертываем его внутри `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>
```

Обратите внимание: рекомендуется, чтобы проект был создан после завершения кода представления. Без его создания вы не сможете увидеть представление на экране предварительного просмотра в Android Studio.

После того, как вы собрали все вместе, вы должны приветствовать следующий экран после запуска операции, содержащей вышеуказанный макет:

Добавление атрибутов в представления

Пользовательские представления также могут принимать пользовательские атрибуты, которые могут использоваться в файлах ресурсов макета Android. Чтобы добавить атрибуты в пользовательский вид, вам необходимо сделать следующее:

1. **Определите имя и тип ваших атрибутов:** это делается внутри `res/values/attrs.xml` (при необходимости создайте его). Следующий файл определяет атрибут цвета для цвета лица смайлика и атрибут enum для выражения смайлика:

```
<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. **Используйте свои атрибуты внутри своего макета:** это можно сделать в любых файлах макета, которые используют ваш пользовательский вид. Следующий файл макета создает экран со счастливым желтым смайликом:

```
<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>
```

Совет. Пользовательские атрибуты не работают с `tools:` префикс в Android Studio 2.1 и старше (и, возможно, в будущих версиях). В этом примере замена `app:smileyColor` на `tools:smileyColor` приведет к `tools:smileyColor` что `smileyColor` не будет установлен во время выполнения или во время разработки.

3. **Прочтите свои атрибуты:** это делается внутри вашего пользовательского исходного кода. Следующий фрагмент `SmileyView` демонстрирует, как можно извлечь атрибуты:

```
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. **(Необязательно) Добавить стиль по умолчанию:** это делается путем добавления стиля со значениями по умолчанию и загрузки его в пользовательское представление. Следующий стиль смайлика по умолчанию представляет собой счастливый желтый цвет:

```
<!-- styles.xml -->
<style name="DefaultSmileyStyle">
    <item name="smileyColor">#ffff00</item>
    <item name="smileyExpression">happy</item>
</style>
```

Что применяется в нашем `SmileyView` , добавив его в качестве последнего параметра

вызова для `obtainStyledAttributes` (см. Код на шаге 3):

```
TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView,
defStyleAttr, R.style.DefaultSmileyViewStyle);
```

Обратите внимание, что любые значения атрибутов, установленные в файле раздутого макета (см. Код на шаге 2), переопределяют соответствующие значения стиля по умолчанию.

5. **(Необязательно). Предоставляйте стили внутри тем:** это делается путем добавления нового атрибута ссылки на стиль, который можно использовать внутри ваших тем и предоставления стиля для этого атрибута. Здесь мы просто назовем наш ссылочный атрибут `smileyStyle` :

```
<!-- attrs.xml -->
<attr name="smileyStyle" format="reference" />
```

Затем мы предоставляем стиль для нашей темы приложения (здесь мы просто используем стиль по умолчанию с шага 4):

```
<!-- themes.xml -->
<style name="AppTheme" parent="AppBaseTheme">
    <item name="smileyStyle">@style/DefaultSmileyStyle</item>
</style>
```

Создание сложного вида

`ViewGroup` **вид** представляет собой настраиваемую `ViewGroup` которая рассматривается как единое представление по окружающему программному коду. Такая ViewGroup может быть действительно полезна в DDD- подобном дизайне, поскольку она может соответствовать совокупности в этом примере Contact. Его можно повторно использовать везде, где отображается контакт.

Это означает, что окружающий код контроллера, Activity, Fragment или Adapter может просто передать объект данных в представление, не разделяя его на несколько различных виджетах пользовательского интерфейса.

Это облегчает повторное использование кода и обеспечивает лучшую конструкцию в соответствии с SOLID priciples .

Макет XML

Обычно вы начинаете. У вас есть существующий бит XML, который вы повторно используете, возможно, как `<include/>` . Извлеките его в отдельный XML-файл и оберните корневой тег в элемент `<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>
```

Этот XML-файл отлично работает в редакторе макетов в Android Studio. Вы можете рассматривать его как любой другой макет.

Составная группа ViewGroup

После того, как у вас есть файл XML, создайте настраиваемую группу представлений.

```
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);
        }
    }
}
```

Метод `init(Context, AttributeSet)` - это то, где вы читали бы любые пользовательские атрибуты XML, как описано в разделе «Добавление атрибутов в представления» .

Используя эти штуки, вы можете использовать его в своем приложении.

Использование в XML

Вот пример `fragment_contact_info.xml` который иллюстрирует, как вы поместили бы один ContactView поверх списка сообщений:

```
<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>
```

Использование в кодексе

Вот пример `RecyclerView.Adapter` который показывает список контактов. Этот пример иллюстрирует, насколько сильно становится код контроллера, когда он полностью свободен от манипуляций View.

```
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
        }
    }
}
```

Советы по настройке CustomView

Не **выделяйте** новые объекты в **onDraw**

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = new Paint(); //Do not allocate here
}
```

Вместо рисования чертежей в холсте ...

```
drawable.setBounds(boundsRect);

drawable.draw(canvas);
```

Используйте Bitmap для более быстрого рисования:

```
canvas.drawBitmap(bitmap, srcRect, boundsRect, paint);
```

Не перерисовывайте весь вид, чтобы обновить только небольшую часть. Вместо этого перерисуйте определенную часть представления.

```
invalidate(boundToBeRefreshed);
```

Если ваше представление делает некоторую непрерывную анимацию, например, часовую `onStop()` показывающую каждую секунду, по крайней мере останавливает анимацию на `onStop()` активности и запускает ее на `onStart()` активности.

Не делайте никаких вычислений внутри метода `onDraw` вида, вы должны закончить рисование перед вызовом `invalidate()` . Используя эту технику, вы можете избежать падения кадров в вашем представлении.

Повороты

Основные операции представления - это перевод, поворот и т. Д. Практически каждый разработчик столкнулся с этой проблемой, когда они используют растровые или градиенты в своем пользовательском представлении. Если в представлении будет показано повернутое представление, и растровое изображение должно быть повернуто в этом пользовательском представлении, многие из нас подумают, что это будет дорого. Многие считают, что поворот растрового изображения очень дорог, потому что для этого вам нужно перевести матрицу пикселей растрового изображения. Но правда в том, что это не так сложно! Вместо поворота растрового изображения просто поверните сам холст!

```
// 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.
```

Соединение для SVG / VectorDrawable как drawableRight

Главным мотивом для разработки этого сложного представления является то, что ниже 5.0 устройств не поддерживает svg в drawable внутри TextView / EditText. Еще одно преимущество - мы можем установить `height` и `width drawableRight` внутри `EditText` . Я отделил его от моего проекта и создал в отдельном модуле.

Имя модуля: custom_edit_drawable (сокращенное имя для prefix-c_d_e)

Префикс «c_d_e_» для использования, чтобы ресурсы модуля приложения не должны переопределять их по ошибке. Пример: префикс «abc» используется Google в библиотеке поддержки.

build.gradle

```
dependencies {
   compile 'com.android.support:appcompat-v7:25.3.1'
}
```

используйте AppCompat> = 23

Файл макета: 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>
```

Пользовательские атрибуты: 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>
```

Код: 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();
        }
    }
}
```

Пример: как использовать вид сверху

Макет: 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>
```

Деятельность: 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);
    }
}
```

Реагирование на события Touch

Многие пользовательские представления должны принимать пользовательское взаимодействие в виде сенсорных событий. Вы можете получить доступ к событиям касания, переопределив `onTouchEvent` . Существует ряд действий, которые вы можете отфильтровать. Основные из них:

- `ACTION_DOWN` : Это срабатывает один раз, когда ваш палец сначала касается вида.
- `ACTION_MOVE` : Это называется каждый раз, когда ваш палец немного перемещается по представлению. Его называют много раз.
- `ACTION_UP` : Это последнее действие, которое нужно вызывать, когда вы поднимаете палец с экрана.

Вы можете добавить следующий вид к вашему представлению, а затем наблюдать за выходом журнала, когда вы касаетесь и перемещаете свой палец вокруг своего вида.

```
@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;
}
```

Дальнейшее чтение:

- Официальная документация для Android: ответ на сенсорные события

Прочитайте Создание пользовательских представлений онлайн:
https://riptutorial.com/ru/android/topic/1446/создание-пользовательских-представлений

глава 237: Создание собственных библиотек для приложений Android

Examples

Создание проекта библиотеки

Чтобы создать libary, вы должны использовать `File -> New -> New Module -> Android Library`. Это создаст базовый проект библиотеки.

Когда это будет сделано, у вас должен быть проект, который настроен следующим образом:

```
[project root directory]
    [library root directory]
    [gradle]
    build.gradle //project level
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    settings.gradle //this is important!
```

Ваш файл `settings.gradle` должен содержать следующее:

```
include ':[library root directory]'
```

Ваш `[library root directory]` должен содержать следующее:

```
[libs]
[src]
   [main]
      [java]
         [library package]
   [test]
      [java]
         [library package]
build.gradle //"app"-level
proguard-rules.pro
```

Ваш файл `build.gradle` должен содержать следующее:

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

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.2"
```

```
    defaultConfig {
        minSdkVersion 14
        targetSdkVersion 23
    }
}
```

При этом ваш проект должен работать нормально!

Использование библиотеки в проекте в качестве модуля

Чтобы использовать библиотеку, вы должны включить ее как зависимость со следующей строкой:

```
compile project(':[library root directory]')
```

Создайте библиотеку, доступную на Jitpack.io

Выполните следующие шаги для создания библиотеки:

1. Создайте учетную запись GitHub.
2. Создайте репозиторий Git, содержащий проект библиотеки.
3. Измените файл `build.gradle` вашего библиотечного проекта, добавив следующий код:

```
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
}
```

Убедитесь, что вы совершили или нажали вышеуказанные изменения в GitHub.

4. Создайте выпуск из текущего кода в Github.
5. Запустите `gradlew install` на свой код.
6. Теперь ваша библиотека доступна по следующей зависимости:

```
compile 'com.github.[YourUser]:[github repository name]:[release tag]'
```

Прочитайте Создание собственных библиотек для приложений Android онлайн: https://riptutorial.com/ru/android/topic/4118/создание-собственных-библиотек-для-приложений-android

глава 238: Создание экрана заставки

замечания

Первый пример (базовый заставку) - не самый эффективный способ его обработки. Таким образом, это основной экран заставки.

Examples

Основной заставку

Экран заставки подобен любой другой деятельности, но он может обрабатывать все ваши потребности при запуске в фоновом режиме. Пример:

Manifest:

```
<?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>
```

Теперь наш всплеск будет называться первым.

Вот пример splashscreen, который также обрабатывает некоторые критические элементы приложения:

```
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(){
```

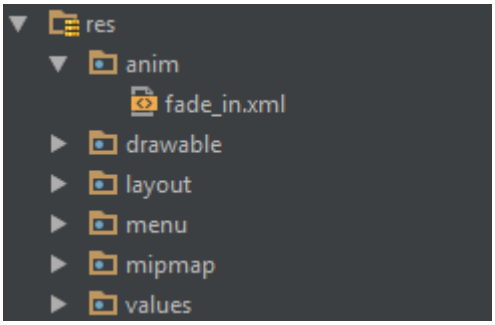
```
        super.onPause();
        finish();
    }

}
```

Заставка с анимацией

В этом примере показан простой, но эффективный заставку с анимацией, которую можно создать с помощью Android Studio.

Шаг 1. Создание анимации

Создайте новый каталог с именем *anim* в каталоге *res* . Щелкните его правой кнопкой мыши и создайте новый файл *ресурсов анимации с* именем *fade_in.xml* :



Затем *добавьте* следующий код в файл *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>
```

Шаг 2: Создайте действие

Создайте *пустую активность* с помощью Android Studio с именем `Splash` . Затем вставьте в него следующий код:

```
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);
    }
}
```

Затем добавьте следующий код в файл макета:

```
<?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>
```

Шаг 3: Замените пусковую установку по умолчанию

Превратите свою активность `Splash` в пусковую установку, добавив следующий код в файл *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>
```

Затем удалите действие запуска по умолчанию, удалив следующий код из файла *AndroidManifest* :

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />

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

Прочитайте Создание экрана заставки онлайн: https://riptutorial.com/ru/android/topic/9316/создание-экрана-заставки

глава 239: Сплит-экран / многоэкранная деятельность

Examples

Split Screen, реализованный в Android Nougat.

Установите этот атрибут в свой манифест или элемент, чтобы включить или отключить отображение нескольких окон:

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

Если для этого атрибута установлено значение true, активность можно запустить в режимах с разделенным экраном и свободной формой. Если для атрибута установлено значение false, действие не поддерживает многооконный режим. Если это значение является ложным, и пользователь пытается запустить эту операцию в многооконном режиме, действие переходит во весь экран.

Если ваше приложение нацелено на уровень API 24, но вы не указываете значение для этого атрибута, значение атрибута по умолчанию равно true.

В следующем коде показано, как указать размер и местоположение по умолчанию для своего действия и его минимальный размер, когда активность отображается в режиме свободной формы:

```
<--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>
```

Отключенные функции в многооконном режиме

Некоторые функции отключены или игнорируются, когда устройство находится в многооконном режиме, поскольку они не имеют смысла для активности, которая может совместно использовать экран устройства с другими действиями или приложениями. К таким функциям относятся:

1. Некоторые параметры настройки пользовательского интерфейса системы отключены; например, приложения не могут скрыть строку состояния, если они не работают в полноэкранном режиме.

2. Система игнорирует изменения атрибута **android: screenOrientation** .

Если ваше приложение нацелено на уровень API 23 или ниже

Если ваше приложение нацелено на уровень API 23 или ниже, и пользователь пытается использовать приложение в многооконном режиме, система принудительно изменяет размер приложения, если приложение не объявит фиксированную ориентацию.

Если ваше приложение не объявляет фиксированную ориентацию, вы должны запустить приложение на устройстве под управлением Android 7.0 или более поздней версии и попытаться разместить приложение в режиме разделения экрана. Убедитесь, что пользовательский интерфейс является приемлемым, когда приложение принудительно изменено.

Если приложение объявляет фиксированную ориентацию, вы должны попытаться поместить приложение в многооконный режим. Убедитесь, что при этом приложение остается в полноэкранном режиме.

Прочитайте Сплит-экран / многоэкранная деятельность онлайн:
https://riptutorial.com/ru/android/topic/7130/сплит-экран---многоэкранная-деятельность

глава 240: Строгий режим политики: инструмент, чтобы поймать ошибку во время компиляции.

Вступление

Strict Mode - специальный класс, введенный в Android 2.3 для отладки. Эти инструменты разработчика обнаруживают вещи, которые делаются случайно, и привлекают их к нашему вниманию, чтобы мы могли их исправить. Он чаще всего используется для обнаружения случайного доступа к диску или сети в основном потоке приложений, где выполняются операции пользовательского интерфейса и происходит анимация. StrictMode - это в основном инструмент для обнаружения ошибки в режиме компиляции.

замечания

StrictMode - это в основном инструмент для обнаружения ошибки в режиме компиляции. Используя это, мы можем избежать утечек памяти в наших приложениях.

Examples

Ниже приведен фрагмент кода для настройки политики StrictMode для потоков. Этот код должен быть установлен в точках входа в наше приложение.

```
StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
    .detectDiskWrites()
    .penaltyLog() //Logs a message to LogCat
    .build())
```

В приведенном ниже коде рассматриваются утечки памяти, такие как обнаружение, когда в SQLLite finalize вызывается или нет.

```
StrictMode.setVmPolicy(new StrictMode.VmPolicy.Builder()
    .detectActivityLeaks()
    .detectLeakedClosableObjects()
    .penaltyLog()
    .build());
```

Прочитайте Строгий режим политики: инструмент, чтобы поймать ошибку во время компиляции. онлайн: https://riptutorial.com/ru/android/topic/8756/строгий-режим-политики--

инструмент--чтобы-поймать-ошибку-во-время-компиляции-

глава 241: Таймер обратного отсчета

параметры

параметр	подробности
`long millisInFuture`	Общая продолжительность работы таймера, так как в будущем вы хотите, чтобы таймер заканчивался. В миллисекундах.
`long countDownInterval`	Интервал, на который вы хотите получать обновления таймера. В миллисекундах.
`long millisUntilFinished`	Параметр, указанный в `onTick()` , указывает, сколько времени осталось CountDownTimer. В миллисекундах

замечания

CountDownTimer - довольно скудный класс - он делает одну вещь очень хорошо. Поскольку вы можете только запустить / отменить CountDownTimer, вам нужно реализовать функции паузы / возобновления, как показано во втором примере. Для более сложных функций или для указания таймера, который должен выполняться неограниченное время, используйте объект Timer .

Examples

Создание простого таймера обратного отсчета

CountDownTimer полезен для многократного выполнения действия в устойчивом интервале для заданной продолжительности. В этом примере мы будем обновлять текстовое представление каждую секунду в течение 30 секунд, рассказывая, сколько времени осталось. Затем, когда таймер закончится, мы установим TextView, чтобы сказать «Готово».

```
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();
```

Более сложный пример

В этом примере мы остановимся / возобновим CountDownTimer на основе жизненного цикла Activity.

```
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;
}
```

Прочитайте Таймер обратного отсчета онлайн: https://riptutorial.com/ru/android/topic/6063/таймер-обратного-отсчета

глава 242: Текст для речи (TTS)

Examples

Текстовая база

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);
        }
    }
}
```

Язык, на котором должен быть произнесен, может быть установлен путем предоставления `Locale setLanguage()` :

```
tts.setLanguage(Locale.CHINESE); // Chinese language
```

Количество поддерживаемых языков зависит от уровня Android. Метод `isLanguageAvailable()` может использоваться для проверки поддержки определенного языка:

```
tts.isLanguageAvailable(Locale.CHINESE);
```

Уровень речевого тона можно установить с помощью `setPitch()` . По умолчанию значение основного тона равно 1.0. Используйте значения менее 1,0, чтобы уменьшить уровень основного тона или значения, превышающие 1,0, чтобы увеличить уровень основного тона:

```
tts.setPitch(0.6);
```

Скорость речи может быть установлена с помощью `setSpeechRate()` . Частота речи по

умолчанию - 1,0. Скорость речи можно удвоить, установив ее на 2,0 или половину, установив ее на 0,5:

```
tts.setSpeechRate(2.0);
```

Внедрение TextToSpeech в API

Холодная наблюдаемая реализация испускает истину, когда двигатель TTS заканчивает говорить, начинает говорить при подписке. Обратите внимание, что API уровня 21 вводит различный способ выполнения речи:

```
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;
}
```

}

Прочитайте Текст для речи (TTS) онлайн: https://riptutorial.com/ru/android/topic/3381/текст-для-речи--tts-

глава 243: Тема DayNight (AppCompat v23.2 / API 14+)

Examples

Добавление темы DayNight в приложение

Тема DayNight дает приложению прекрасную возможность переключения цветовых схем на основе времени суток и последнего известного местоположения устройства.

Добавьте в свой `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>
```

Темы, которые вы можете продлить, чтобы добавить функцию переключения тем дня ночью, следующие:

- `"Theme.AppCompat.DayNight"`
- `"Theme.AppCompat.DayNight.NoActionBar"`
- `"Theme.AppCompat.DayNight.DarkActionBar"`

Помимо `colorPrimary` , `colorPrimaryDark` и `colorAccent` вы также можете добавить любые другие цвета, которые вы хотели бы переключить, например `textColorPrimary` или `textColorSecondary` . Вы можете добавить пользовательские цвета вашего приложения в этот `style` .

Для переключения темы на работу вам необходимо определить значение по умолчанию `colors.xml` в `colors.xml res/values` и еще один `colors.xml` в `colors.xml res/values-night` и определить цвета день / ночь соответствующим образом.

Чтобы переключить тему, вызовите метод `AppCompatDelegate.setDefaultNightMode(int)` из вашего кода Java. (Это изменит цветовую схему для всего приложения, а не только одну активность или фрагмент.) Например:

```
AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
```

Вы можете передать любой из следующих трех по вашему выбору:

- `AppCompatDelegate.MODE_NIGHT_NO` : это задает тему по умолчанию для вашего приложения и принимает цвета, определенные в каталоге `res/values` . Для этой темы

рекомендуется использовать светлые цвета.

- `AppCompatDelegate.MODE_NIGHT_YES` : это устанавливает ночную тему для вашего приложения и принимает цвета, определенные в каталоге `res/values-night` . Для этой темы рекомендуется использовать темные цвета.
- `AppCompatDelegate.MODE_NIGHT_AUTO` : этот параметр автоматически переключает цвета приложения в зависимости от времени суток и цветов, которые вы определили в каталогах `values` и `values-night` .

Также можно получить текущий статус ночного режима с помощью метода `getDefaultNightMode()` . Например:

```
int modeType = AppCompatDelegate.getDefaultNightMode();
```

Обратите внимание, однако, что переключатель темы не будет сохраняться, если вы убьете приложение и снова его закроете. Если вы это сделаете, тема вернется к `AppCompatDelegate.MODE_NIGHT_AUTO` , который является значением по умолчанию. Если вы хотите, чтобы переключатель темы сохранялся, убедитесь, что вы сохраняете значение в общих настройках и загружаете сохраненное значение каждый раз, когда приложение открывается после его уничтожения.

Прочитайте Тема DayNight (AppCompat v23.2 / API 14+) онлайн:
https://riptutorial.com/ru/android/topic/7650/тема-daynight--appcompat-v23-2---api-14plus-

глава 244: Тема, Стиль, Атрибут

Examples

Использовать пользовательскую тему по всему миру

В themes.xml:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

В AndroidManifest.xml:

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">

    <!-- Activity declarations here -->

</application>
```

Определение первичных, первичных темных и акцентных цветов

Вы можете настроить цветовую палитру вашей темы .

Использование **каркасного** API ,

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>
```

Использование **библиотеки поддержки Appcompat** (и `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>
```

Использовать пользовательскую тему для каждой операции

В themes.xml:

```
<style name="MyActivityTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

В 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>
```

Цвет Overscroll (API 21+)

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorEdgeEffect">@color/my_color</item>
</style>
```

Цвет пульсации (API 21+)

5.0

Анимация пульсации отображается, когда пользователь нажимает интерактивные представления.

Вы можете использовать тот же цвет пульсации, который используется вашим приложением, назначая `?android:colorControlHighlight` в ваших представлениях. Вы можете настроить этот цвет, изменив атрибут `android:colorControlHighlight` в своей теме:

Этот цвет эффекта можно изменить:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorControlHighlight">@color/my_color</item>
</style>
```

Или, если вы используете тему материала:

```
<style name="AppTheme" parent="android:Theme.Material.Light">
    <item name="android:colorControlHighlight">@color/your_custom_color</item>
</style>
```

Световая строка состояния (API 23+)

Этот атрибут может изменить фон значков строки состояния (вверху экрана) на белый.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowLightStatusBar">true</item>
</style>
```

Прозрачная навигация и бары состояния (API 19+)

Панель навигации (внизу экрана) может быть прозрачной. Вот как это сделать.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentNavigation">true</item>
</style>
```

Строка состояния (верхняя часть экрана) может быть прозрачной, применяя этот атрибут к стилю:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentStatus">true</item>
</style>
```

Цвет панели навигации (API 21+)

5.0

Этот атрибут используется для изменения панели навигации (одна, содержащая кнопку Back, Home Recent). Обычно он черный, однако цвет может быть изменен.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:navigationBarColor">@color/my_color</item>
</style>
```

Наследование темы

При определении тем обычно используется тема, предоставляемая системой, а затем изменения изменяют внешний вид в соответствии с его собственным приложением. Например, так `Theme.AppCompat` тема `Theme.AppCompat` :

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

Эта тема теперь имеет все свойства стандартной темы `Theme.AppCompat` , кроме тех, которые мы явно изменили.

Существует также ярлык при наследовании, обычно используемый, когда он наследуется

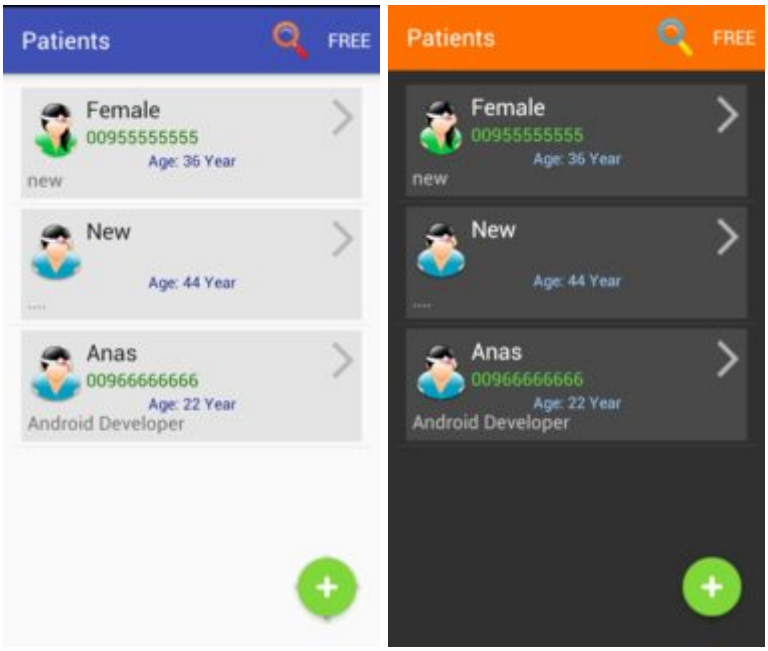
от его собственной темы:

```
<style name="AppTheme.Red">
    <item name="colorAccent">@color/red</item>
</style>
```

Поскольку у него уже есть `AppTheme.` в начале его имени он автоматически наследует его, не требуя определения `parent` темы. Это полезно, когда вам нужно создать определенные стили для части (например, одного действия) вашего приложения.

Несколько тем в одном приложении

Используя более чем одну тему в приложении для Android, вы можете добавлять пользовательские цвета для каждой темы, чтобы:



Во-первых, мы должны добавить наши темы в `style.xml` следующим образом:

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">

</style>

<!--  -->
<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >

</style>
......
```

Выше вы можете увидеть **OneTheme** и **TwoTheme** .

Теперь перейдите в свой `AndroidManifest.xml` и добавьте эту строку: `android:theme="@style/OneTheme"` в свой тег *приложения* , это сделает **тему OneTheme** по умолчанию:

```
<application
        android:theme="@style/OneTheme"
        ...>
```

Создайте новый xml-файл с именем `attrs.xml` и добавьте этот код:

```
<?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) -->
```

Вернитесь к `style.xml` и добавьте эти цвета со своими значениями для каждой темы:

```
<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>
```

Теперь у вас есть пользовательские цвета для каждой темы, давайте добавим этот цвет в наши представления.

Добавьте цвет **custom_blue** в TextView, используя «? Attr /»:

Перейдите к своему imageView и добавьте этот цвет:

```
<TextView>
    android:id="@+id/txte_view"
    android:textColor="?attr/custom_blue" />
```

Моу мы можем изменить тему только одной строкой `setTheme(R.style.TwoTheme);` эта строка должна быть до `setContentView()` методе `onCreate()` , как этот `Activity.java` :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setTheme(R.style.TwoTheme);
    setContentView(R.layout.main_activity);
    ....
}
```

изменить тему для всех действий сразу

Если мы хотим изменить тему для всех действий, нам нужно создать новый класс с именем MyActivity, который расширяет класс `AppCompatActivity` (или класс `Activity`) и добавляет строку `setTheme(R.style.TwoTheme);` to **onCreate ()** :

```
public class MyActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (new MySettings(this).isDarkTheme())
            setTheme(R.style.TwoTheme);
    }
}
```

Наконец, перейдите ко всем вашим действиям, добавьте, чтобы все они расширили базовый класс **MyActivity** :

```
public class MainActivity extends MyActivity {
    ....
}
```

Чтобы изменить тему, просто перейдите в **MyActivity** и измените `R.style.TwoTheme` на свою тему (`R.style.OneTheme` , `R.style.ThreeTheme` ....).

Прочитайте Тема, Стиль, Атрибут онлайн: https://riptutorial.com/ru/android/topic/1843/тема--стиль--атрибут

глава 245: Тестирование пользовательского интерфейса с помощью эспрессо

замечания

Эспрессо

Шрифт Espresso поможет вам написать ваши тесты и то, что вы хотите проверить:

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Также всегда хорошим местом для справки является официальная документация:

https://google.github.io/android-testing-support-library/docs/espresso/index.html

Расширенные предложения для эспрессо-видео от Google:
https://www.youtube.com/watch?v=isihPOY2vS4

Поиск проблемы

- При попытке прокрутки сначала закройте клавиатуру:

Watchout: не использование версии «Espresso» не будет делать ничего, если используется вне ViewAction. Это может быть не очевидно, если у вас есть импорт в версии ViewAction, поскольку у них есть точно такое же имя метода.

```
ViewActions.closeSoftKeyboard;
Espresso.closeSoftKeyboard();
```

- При выполнении тестов вместе в наборе, а не отдельно, имейте в виду, что активность из предыдущего теста все еще может выполняться. Не полагайтесь на предыдущий тест onDestroy (), который вызывается перед текущими тестами onResume (). **Оказывается, это на самом деле ошибка** : http://b.android.com/201513

Examples

Настройка эспрессо

В файле `build.gradle` вашего Android-приложения добавьте следующие зависимости:

```
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'
}
```

Укажите `AndroidJUnitRunner` для `testInstrumentationRunner` параметра в `build.gradle` файле.

```
android {

  defaultConfig {
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }

}
```

Кроме того, добавьте эту зависимость для обеспечения намеренной насмешливой поддержки

```
androidTestCompile 'com.android.support.test.espresso:espresso-intents:2.2.2'
```

И добавьте это для поддержки тестирования веб-просмотра

```
// Espresso-web for WebView support
androidTestCompile 'com.android.support.test.espresso:espresso-web:2.2.2'
```

Создать тестовый класс Espresso

Поместите следующий java-класс в **src / androidTest / java** и запустите его.

```
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
  }

}
```

Развернуть Свернуть 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);
      }
    };
  }

}
```

Эспрессо простой тест пользовательского интерфейса

Инструменты тестирования UI

Двумя основными инструментами, которые в настоящее время используются в основном для тестирования пользовательского интерфейса, являются Appium и Espresso.

Appium	Эспрессо
тест черного ящика	тестирование белого / серого ящика
то, что вы видите, это то, что вы можете проверить	может изменить внутреннюю работу приложения и подготовить его для тестирования, например, сохранить некоторые данные в базе данных или sharedpreferences перед запуском теста

Appium	Эспрессо
используемые в основном для интеграции тестов конца и конца и целых потоков пользователей	тестирование функциональности экрана и / или потока
может быть абстрагирован, так что тестовое исполнение может быть выполнено на iOS и Android	Только Android
хорошо поддерживается	хорошо поддерживается
поддерживает параллельное тестирование на нескольких устройствах с селеновой сеткой	Не из параллельного тестирования, плагины, такие как Spoon, существуют до тех пор, пока не появится истинная поддержка Google

Как добавить эспрессо в проект

```
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'
}
```

ПРИМЕЧАНИЕ. Если вы используете последние библиотеки поддержки, аннотации и т. Д., Вы должны исключить старые версии из эспрессо, чтобы избежать коллизий:

```
    // 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'
    }
```

Помимо этого импорта необходимо добавить контрольный бегун для андроидов для сборки.gradle android.defaultConfig:

```
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
```

Настройка устройства

Для не-flaky-теста рекомендуется установить следующие настройки на ваших устройствах:

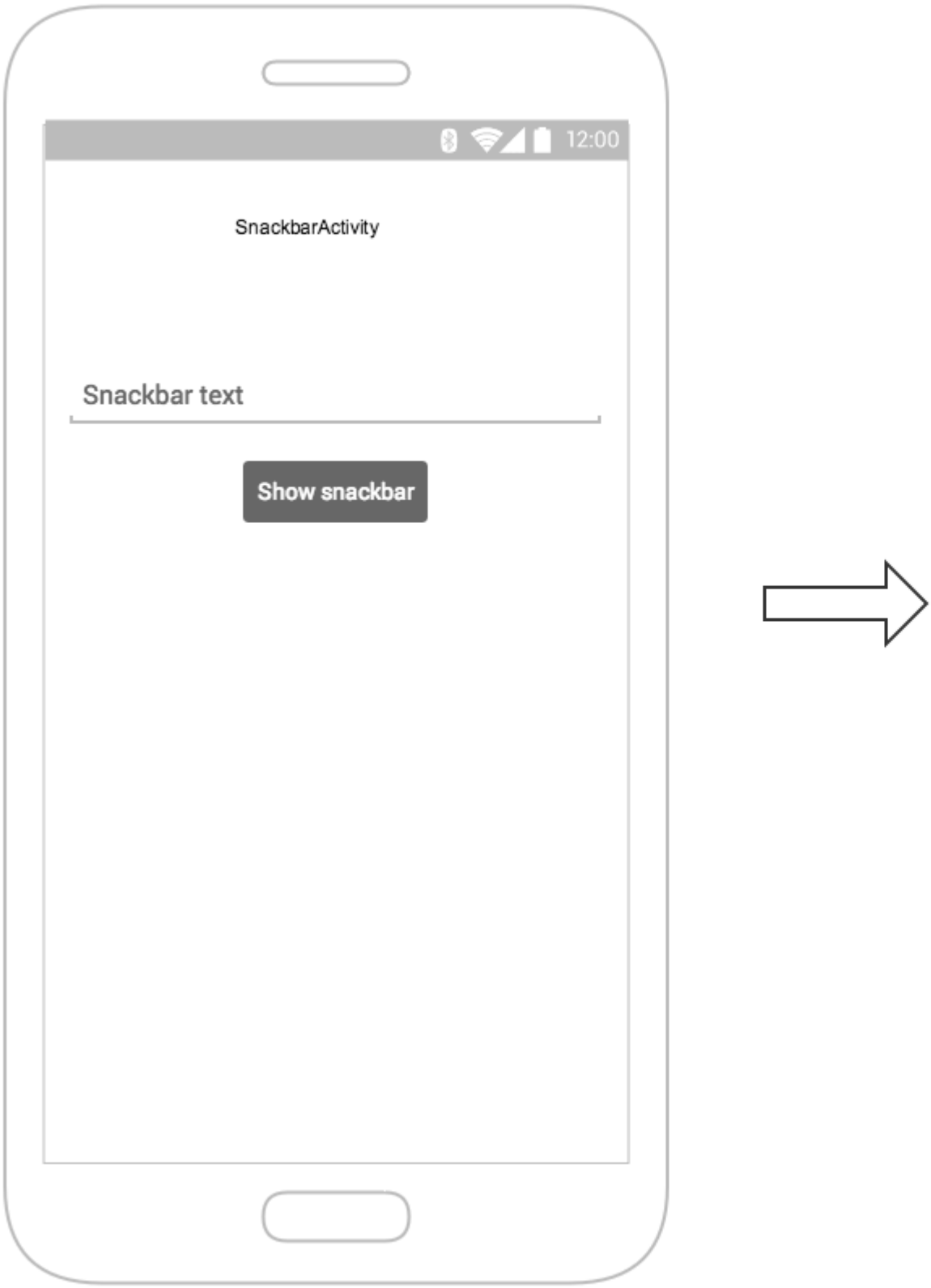
- Параметры разработчика / Отключить анимацию - уменьшает шелушение тестов
- Возможности разработчика / Будьте бодрым - если у вас есть специализированные устройства для тестов, это полезно
- Параметры разработчика / размеры буфера регистратора - установите более высокий номер, если на вашем телефоне запущены очень большие тестовые пакеты

- Доступность / Touch & Hold delay - долго, чтобы избежать проблем с нажатием на эспрессо

Достаточно установка из реального мира? Ну, теперь, когда это с пути, давайте посмотрим, как настроить небольшой тест

Написание теста

Предположим, что у нас есть следующий экран:



Экран содержит:

- текстовое поле ввода - **R.id.textEntry**
- кнопка, которая показывает закуску с напечатанным текстом при нажатии - **R.id.shownSnackbarBtn**
- snackbar, который должен содержать пользовательский текст - **android.support.design.R.id.snackbar_text**

Теперь давайте создадим класс, который будет проверять наш поток:

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

Как вы заметили, есть 3-4 вещи, которые вы могли заметить часто:

onView (withXYZ) <- viewMatchers с ними вы можете найти элементы на экране

выполнить (click ()) <- viewActions, вы можете выполнять действия над элементами, которые вы ранее обнаружили

check (matches (isDisplayed ())) <- viewAssertions, проверки, которые вы хотите сделать на экранах, которые вы ранее обнаружили

Все эти и многие другие можно найти здесь: https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/index.html

То есть, теперь вы можете запустить тест либо щелкнув правой кнопкой мыши по имени / тесту класса, либо выбрав команду «Запустить тест» или с помощью команды:

```
./gradlew connectedFLAVORNAMEAndroidTest
```

Вверх Навигация

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

Обратите внимание, что это обходное решение и столкнется с другими представлениями, которые имеют одно и то же описание содержимого.

Выполнение действия в представлении

С помощью метода выполнения `ViewActions` просматривать `ViewActions` в представлении. Класс `ViewActions` предоставляет вспомогательные методы для наиболее распространенных действий, таких как:

```
ViewActions.click()
ViewActions.typeText()
ViewActions.clearText()
```

Например, чтобы щелкнуть по виду:

```
onView(...).perform(click());
onView(withId(R.id.button_simple)).perform(click());
```

Вы можете выполнить более одного действия с помощью одного вызова:

```
onView(...).perform(typeText("Hello"), click());
```

Если представление, с которым вы работаете, находится внутри `ScrollView` (вертикальное или горизонтальное), рассмотрите предыдущие действия, которые требуют отображения вида (например, `click()` и `typeText()`) с помощью `scrollTo()` . Это гарантирует, что

представление отображается до перехода к другому действию:

```
onView(...).perform(scrollTo(), click());
```

Поиск представления с помощью onView

С помощью `ViewMatchers` вы можете найти представление в текущей иерархии представлений.

Чтобы найти представление, используйте метод `onView()` с `onView()` вида, который выбирает правильный вид. `onView()` возвращают объект типа `ViewInteraction` .

Например, поиск вида по его `R.id` так же просто, как:

```
onView(withId(R.id.my_view))
```

Поиск представления с текстом:

```
onView(withText("Hello World"))
```

Эспрессо

У эспрессо по умолчанию есть много совпадений, которые помогут вам найти представления, которые вам понадобятся для проверки или взаимодействия с ними.

Наиболее важные из них можно найти в следующем чит-листе:

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Вот некоторые примеры совпадений:

- withId (R.id.ID_of_object_you_are_looking_for);
- withText («Некоторый текст, который вы ожидаете от объекта»);
- isDisplayed () <- проверка - вид видимый
- doesNotExist () <- проверить, что вид не существует

Все они очень полезны для повседневного использования, но если у вас более сложные взгляды, написанные вами, пользовательские матчи могут сделать тесты более читабельными, и вы можете их повторно использовать в разных местах.

Существует 2 наиболее распространенных типа матчи, которые вы можете расширить: **TypeSafeMatcher BoundedMatcher**

Для реализации TypeSafeMatcher требуется проверить экземпляр instanceOf, против которого вы утверждаете, если его правильный тип соответствует некоторым его свойствам против значения, которое вы предоставили совпадению.

Например, введите безопасный соединитель, который проверяет изображение, имеет правильную возможность:

```
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("]");
        }
    }
}
```

Использование соединителя можно обернуть следующим образом:

```
  public static Matcher<View> withDrawable(final int resourceId) {
    return new DrawableMatcher(resourceId);
}

 onView(withDrawable(R.drawable.someDrawable)).check(matches(isDisplayed()));
```

Ограниченные шаблоны похожи, вам просто не нужно делать проверку типа, но, поскольку это делается автоматически для вас:

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

Подробнее о помощниках можно прочитать:

http://hamcrest.org/

https://developer.android.com/reference/android/support/test/espresso/matcher/ViewMatchers.html

Общий эспрессо

Настройка эспрессо:

```
androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
androidTestCompile 'com.android.support.test:runner:0.5'
```

ViewMatchers - коллекция объектов, реализующих `Matcher<? super View>` . Вы можете передать один или несколько из них методу `onView` чтобы найти представление в текущей

иерархии представлений.

ViewActions - коллекция `ViewActions` которая может быть передана `ViewInteraction.perform()` (например, `click()`).

ViewAssertions - коллекция `ViewAssertions` которая может быть передана `ViewInteraction.check()` . В большинстве случаев вы будете использовать утверждение совпадений, которое использует совпадение View для утверждения состояния выбранного в данный момент вида.

Кошелек для эспрессо от 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()
```

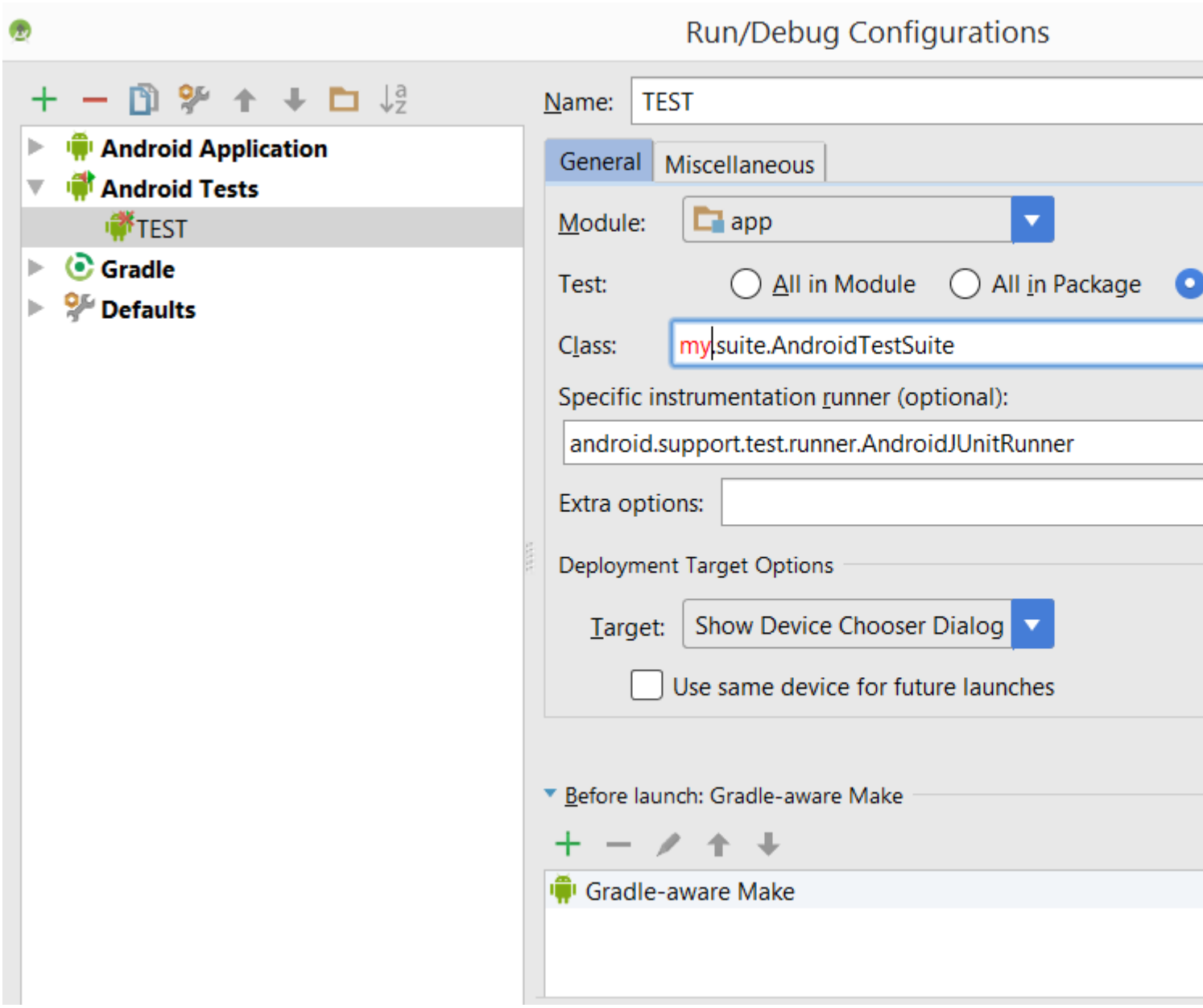
SEE ALSO

Preference matchers

Suite .

```
/**
 * Runs all unit tests.
 */
@RunWith(Suite.class)
@Suite.SuiteClasses({MyTest1.class ,
         MyTest2.class,
         MyTest3.class})
public class AndroidTestSuite {}
```

Затем в AndroidStudio вы можете работать с градиентом или настраивать новую конфигурацию, например:



Набор тестов может быть вложен.

Прочитайте Тестирование пользовательского интерфейса с помощью эспрессо онлайн: https://riptutorial.com/ru/android/topic/3485/тестирование-пользовательского-интерфейса-с-

помощью-эспрессо

глава 246: Тост

Вступление

Тост обеспечивает простую обратную связь о операции в небольшом всплывающем окне и автоматически исчезает после таймаута. Он заполняет объем пространства, необходимый для сообщения, и текущая активность остается видимой и интерактивной.

Синтаксис

- Toast makeText (контекст контекста, текст CharSequence, int duration)
- Toast makeText (контекст контекста, int resId, int duration)
- void setGravity (int gravity, int xOffset, int yOffset)
- void show ()

параметры

параметр	подробности
контекст	Контекст для отображения вашего Toast in. `this` обычно используется в Activity, а `getActivity()` обычно используется в фрагменте
текст	CharSequence, который указывает, какой текст будет показан в Toast. Любой объект, реализующий CharSequence, может использоваться, включая String
RESID	Идентификатор ресурса, который может использоваться для предоставления ресурса Строка для отображения в Toast
продолжительность	Целочисленный флаг, показывающий, как долго будет отображаться тост. Параметры: `Toast.LENGTH_SHORT` и `Toast.LENGTH_LONG`
сила тяжести	Целое число, определяющее позицию или «гравитацию» Тоста. См. Параметры здесь
xOffset	Задает горизонтальное смещение для позиции Toast
YOffset	Определяет вертикальное смещение для позиции Toast

замечания

Тост обеспечивает простую обратную связь об операции в небольшом всплывающем окне. Он заполняет объем пространства, необходимый для сообщения, и текущая активность остается видимой и интерактивной.

Более поздняя альтернатива Toast - SnackBar. SnackBar предлагает обновленный визуальный стиль и позволяет пользователю отклонить сообщение или предпринять дальнейшие действия. Подробности см. В документации SnackBar .

Официальная документация:

https://developer.android.com/reference/android/widget/Toast.html

Examples

Установить положение тоста

Стандартное уведомление тоста появляется в нижней части экрана, выровненного в горизонтальном центре. Вы можете изменить эту позицию с помощью `setGravity(int, int, int)` . Он принимает три параметра: константу силы тяжести, смещение по оси x и смещение позиции y.

Например, если вы решите, что тост должен появиться в верхнем левом углу, вы можете установить гравитацию следующим образом:

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

Отображение сообщения о Toast

В Android, Toast - это простой элемент пользовательского интерфейса, который может использоваться для предоставления контекстной обратной связи пользователю.

Чтобы отобразить простое сообщение Toast, мы можем сделать следующее.

```
// 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();
```

Или, чтобы показать Toast inline, не удерживая объект Toast, вы можете:

```
Toast.makeText(context, "Ding! Your Toast is ready.", Toast.LENGTH_SHORT).show();
```

ВАЖНО: Убедитесь, что метод `show()` вызывается из потока пользовательского интерфейса. Если вы пытаетесь показать `Toast` из другого потока, вы можете, например, использовать метод `runOnUiThread` для `Activity` .

В противном случае, имея в виду попытку изменить пользовательский интерфейс, создав Toast, вы получите `RuntimeException` которое будет выглядеть следующим образом:

```
java.lang.RuntimeException: Can't create handler inside thread that has not called
Looper.prepare()
```

Самый простой способ обработки этого исключения - просто использовать runOnUiThread: синтаксис показан ниже.

```
   runOnUiThread(new Runnable() {
        @Override
        public void run() {
            // Your code here
        }
    });
```

Создание пользовательского тоста

Если вы не хотите использовать представление Toast по умолчанию, вы можете предоставить свой собственный, используя метод `setView(View)` на объекте `Toast` .

Во-первых, создайте XML-макет, который вы хотели бы использовать в своем 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>
```

Затем, создавая свой тост, надуйте свой собственный вид из XML и вызовите `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();
```

Реальный безопасный способ отображения 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();
                }
            }
        });
    }
```

Не забудьте добавить `MainApplication` в `manifest` .

Теперь вызовите его из любого потока, чтобы отобразить тост-сообщение.

```
MainApplication.getApp().showToast("Some message", Toast.LENGTH_LONG);
```

Показать сообщение с тостами над мягкой клавиатурой

По умолчанию Android отображает сообщения Toast в нижней части экрана, даже если клавиатура отображается. Это покажет сообщение Toast чуть выше клавиатуры.

```
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 безопасный способ отображения сообщения Toast (для AsyncTask)

Если вы не хотите распространять приложение и поддерживать потоки сообщений в потоковом режиме, убедитесь, что вы показываете их в разделе после выполнения своих задач 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();
    }

}
```

Прочитайте Тост онлайн: https://riptutorial.com/ru/android/topic/1741/тост

глава 247: Уведомления

Examples

Создание простого уведомления

В этом примере показано, как создать простое уведомление, которое запускает приложение, когда пользователь нажимает на него.

Укажите содержание уведомления:

```
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
```

Создайте намерение стрелять по клику:

```
Intent intent = new Intent(this, MainActivity.class);
PendingIntent pi = PendingIntent.getActivity(this, 0, intent, Intent.FLAG_ACTIVITY_NEW_TASK);
mBuilder.setContentIntent(pi);
```

Наконец, создайте уведомление и покажите его

```
NotificationManager mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(0, mBuilder.build());
```

Уведомление заголовков с тикером для более старых устройств

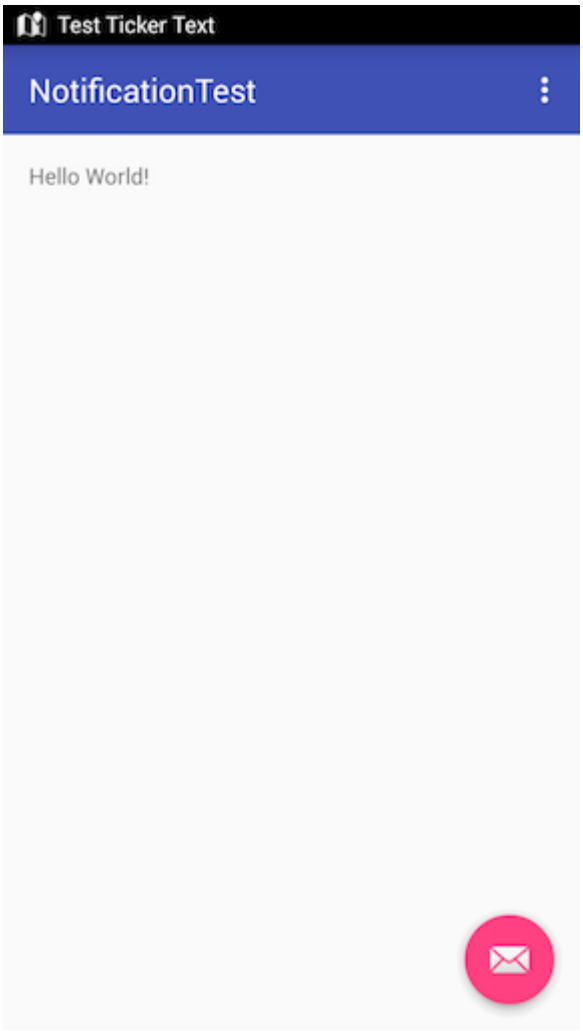
Вот как сделать уведомление Heads Up для совместимых устройств и использовать тикер для более старых устройств.

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

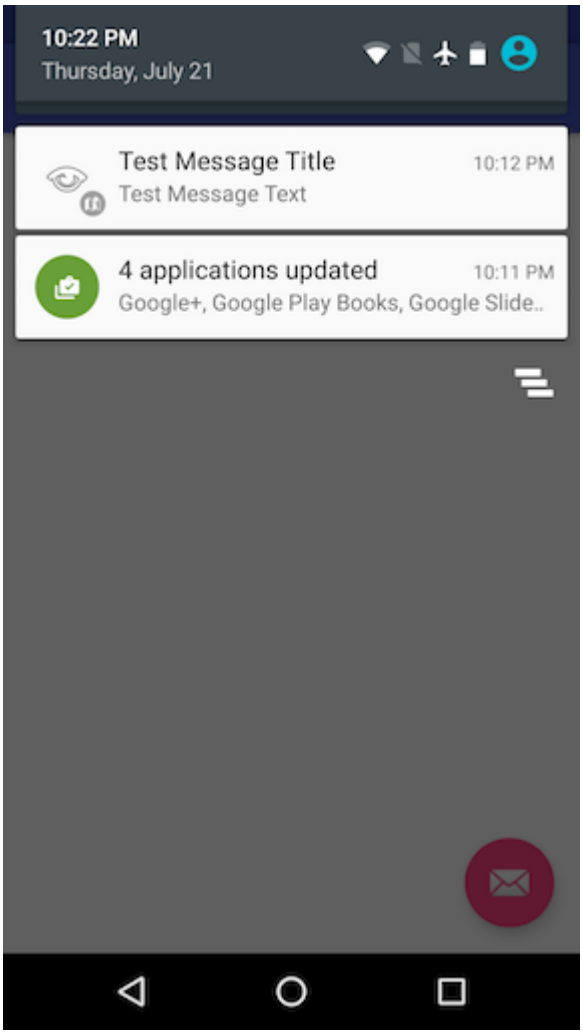
Вот как это выглядит на Android Marshmallow с уведомлением Heads Up:

Вот как это выглядит на Android KitKat с тикером:

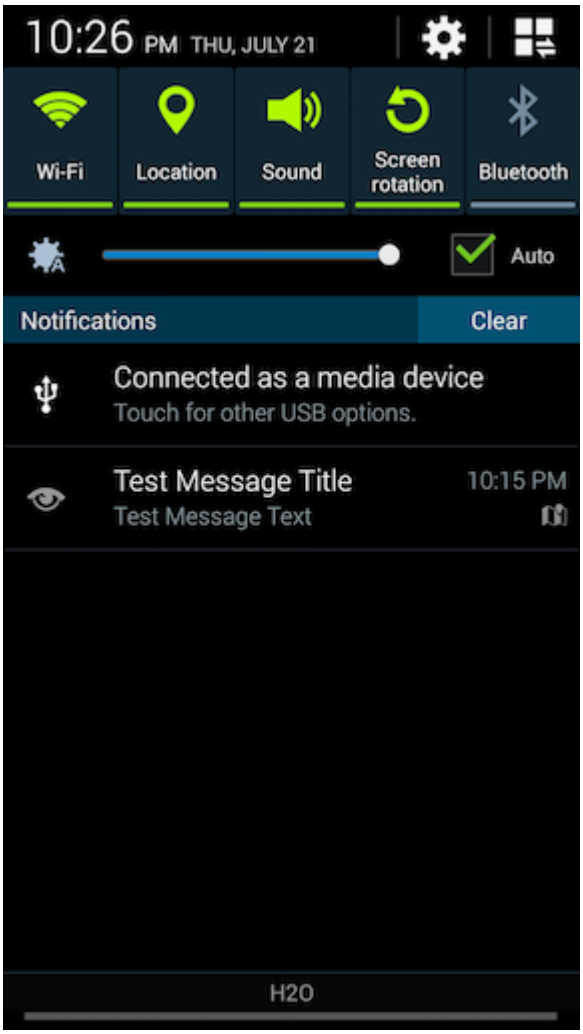


Во всех версиях Android `Notification` отображается в ящике уведомлений.

Android 6.0 Marshmallow:



Android 4.4.x KitKat:



Настройка различных приоритетов в уведомлении

```
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);
```

> Когда уведомление содержит изображение, и вы хотите автоматически развернуть изображение, когда полученное уведомление использует « PRIORITY_MAX», вы можете использовать другие уровни приоритета согласно требованиям

Различные приоритетные уровни Информация:

PRIORITY_MAX - использовать для критических и срочных уведомлений, которые предупреждают пользователя о том, что это критически важно или необходимо решить, прежде чем они смогут продолжить выполнение определенной задачи.

PRIORITY_HIGH - Используйте прежде всего важные сообщения, такие как сообщения или

чаты с содержанием, которое особенно интересно для пользователя. Высокоприоритетные уведомления вызывают отображение уведомлений о хедз-ап.

PRIORITY_DEFAULT - используется для всех уведомлений, которые не попадают ни в один из других приоритетов, описанных здесь.

PRIORITY_LOW - используется для уведомлений, о которых вы хотите, чтобы информация о них была информирована, но это менее актуально. Низкоприоритетные уведомления, как правило, отображаются в нижней части списка, что делает их хорошим выбором для таких вещей, как публичные или неориентированные социальные обновления. Пользователь попросил сообщить о них, но эти уведомления никогда не должны иметь приоритет над срочными или прямой связи.

PRIORITY_MIN - использовать для контекстной или справочной информации, такой как информация о погоде или информация о контекстном местоположении. Уведомления о минимальном приоритете не отображаются в строке состояния. Пользователь обнаруживает их при расширении оттенка уведомления.

Ссылки: Руководство по дизайну материалов - уведомления

Планирование уведомлений

Иногда требуется отображать уведомление в определенное время, задачу, которая, к сожалению, не является тривиальной для системы Android, так как нет метода `setTime()` или аналогичного для уведомлений. В этом примере описываются шаги, необходимые для планирования уведомлений с помощью `AlarmManager` :

1. **Добавьте** **`BroadcastReceiver`** который прослушивает `Intent` s, транслируемую Android `AlarmManager` .

Это место, где вы создаете свое уведомление на основе дополнений, предусмотренных в `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. **Зарегистрируйте** `BroadcastReceiver` в вашем файле `AndroidManifest.xml` (в противном случае получатель не получит никаких `Intent` s из `AlarmManager`):

```
<receiver
    android:name=".NotificationReceiver"
    android:enabled="true" />
```

3. **График уведомления** пропускания `PendingIntent` для `BroadcastReceiver` с необходимым `Intent` статистов в систему `AlarmManager` . Ваш `BroadcastReceiver` получит `Intent` по `Intent` заданного времени и отобразит уведомление. Следующий метод рассылает уведомление:

```
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);
}
```

Обратите внимание, что вышеперечисленное `42` должно быть уникальным для каждого запланированного уведомления, иначе `PendingIntent` s заменит друг друга, вызывая нежелательные эффекты!

4. **Отмените уведомление** , восстановив связанный `PendingIntent` и отменив его в системе `AlarmManager` . Следующий метод отменяет уведомление:

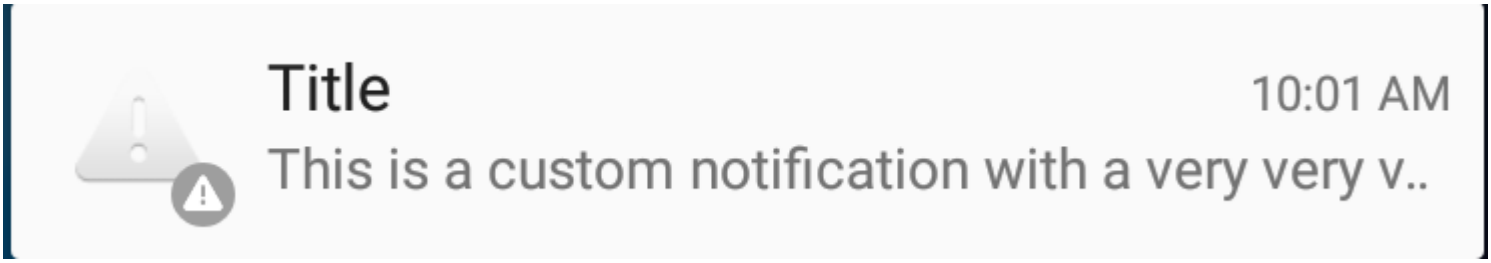
```
public static void cancelNotification(Context context, String title, String text) {
    Intent intent = new Intent(context, NotificationReceiver.class);
    intent.putExtra("title", title);
    intent.putExtra("text", text);
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
    // Cancel notification
    AlarmManager manager = (AlarmManager)
context.getSystemService(Context.ALARM_SERVICE);
    manager.cancel(pending);
}
```

Обратите внимание, что приведенное выше число `42` должно соответствовать числу с шага 3!

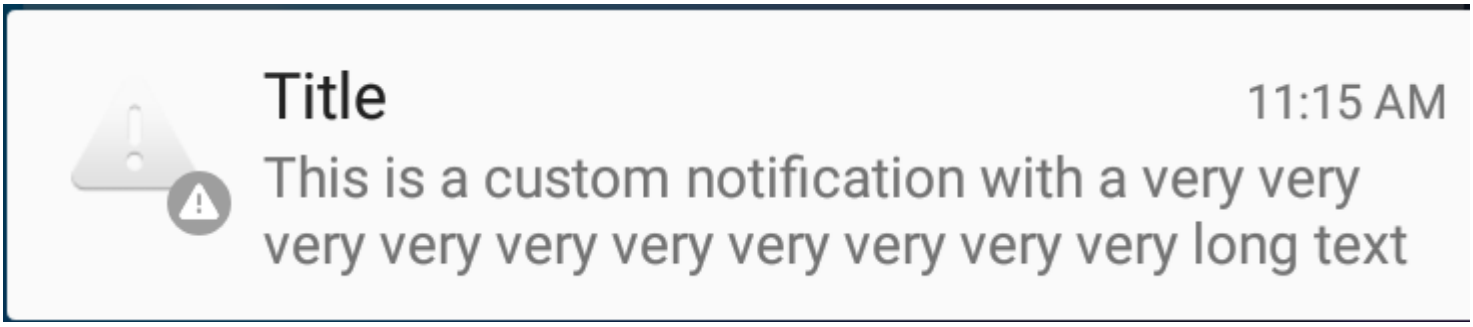
Установить настраиваемое уведомление - показать полный текст контента

Если вы хотите иметь длинный текст для отображения в контексте, вам нужно установить пользовательский контент.

Например, у вас есть следующее:



Но вы хотите, чтобы ваш текст полностью отображался:



Все, что вам нужно сделать, это **добавить стиль** к вашему контенту, как показано ниже:

```
  private void generateNotification(Context context) {
        String message = "This is a custom notification with a very very very very very very
very very very very very long text";
        Bitmap largeIcon = BitmapFactory.decodeResource(getResources(),
android.R.drawable.ic_dialog_alert);

        NotificationCompat.Builder builder = new NotificationCompat.Builder(context);

        builder.setContentTitle("Title").setContentText(message)
                .setSmallIcon(android.R.drawable.ic_dialog_alert)
                .setLargeIcon(largeIcon)
                .setAutoCancel(true)
                .setWhen(System.currentTimeMillis())
                .setStyle(new NotificationCompat.BigTextStyle().bigText(message));

        Notification notification = builder.build();
        NotificationManagerCompat notificationManager =
NotificationManagerCompat.from(context);
        notificationManager.notify(101, notification);
    }
```

Установите значок пользовательского уведомления, используя библиотеку «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);
```

А затем определите макет внутри папки макетов:

```
<?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>
```

Динамическое получение правильного размера пикселя для большого значка

Если вы создаете изображение, декодируете изображение или изменяете размер изображения в соответствии с большой областью изображения уведомлений, вы можете получить правильные размеры пикселей так:

```
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);
```

Текущее уведомление с кнопкой «Действие»

```
// 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);
```

Зарегистрируйте широковещательный приемник для того же действия, чтобы обработать событие нажатия кнопки действия.

Прочитайте Уведомления онлайн: https://riptutorial.com/ru/android/topic/1347/уведомления

глава 248: укротитель

замечания

Обработчик может быть легко использован для выполнения кода после отложенного времени. Это также полезно для выполнения кода повторно через определенное количество времени, вызывая метод Handler.postDelayed () снова из метода Runnable run ().

Examples

Использование обработчика для выполнения кода после задержки времени

Выполнение кода через 1,5 секунды:

```
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
```

Выполнение кода раз в 1 секунду:

```
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 и связь между Threads

Поскольку `Handler` используются для отправки `Message` s и `Runnable` s в очередь сообщений Thread, легко реализовать связь на основе событий между несколькими Threads. Каждый поток, имеющий `Looper` , способен принимать и обрабатывать сообщения. `HandlerThread` - это `HandlerThread` , который реализует такой `Looper` , например, главный Thread (`HandlerThread` пользовательского интерфейса) реализует функции `HandlerThread` .

Создание обработчика для текущей темы

```
Handler handler = new Handler();
```

Создание обработчика для основного потока (поток пользовательского интерфейса)

```
Handler handler = new Handler(Looper.getMainLooper());
```

Отправить Runnable из другого потока в главную тему

```
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();
```

Создание обработчика для другого HandlerThread и отправка ему событий

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

Остановить обработчик от выполнения

Чтобы остановить обработчик от выполнения, удалите обратный вызов, прикрепленный к нему, используя runnable, запущенный внутри него:

```
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);
}
```

Используйте Handler для создания таймера (аналогично javax.swing.Timer)

Это может быть полезно, если вы пишете игру или что-то, что нужно выполнить часть кода каждые несколько секунд.

```
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 ();
    }
}
```

Пример использования:

```
Timer timer = new Timer(new Runnable() {
    public void run() {
        System.out.println("Hello");
    }
}, 1000, true)
```

Этот код будет печатать «Hello» каждую секунду.

Прочитайте укротитель онлайн: https://riptutorial.com/ru/android/topic/1425/укротитель

глава 249: Улучшение диалогов оповещений

Вступление

Эта тема посвящена улучшению `AlertDialog` с дополнительными функциями.

Examples

Диалоговое окно оповещения, содержащее ссылку с возможностью клика

Чтобы отобразить диалоговое окно предупреждения, содержащее ссылку, которую можно открыть, щелкнув по ней, вы можете использовать следующий код:

```
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
```

Прочитайте Улучшение диалогов оповещений онлайн:
https://riptutorial.com/ru/android/topic/10163/улучшение-диалогов-оповещений

глава 250: Улучшение производительности Android с помощью знаковых шрифтов

замечания

Значок Шрифты похожи на обычные типы шрифтов, которые имеют символы вместо букв. Он может быть использован в вашем приложении максимально легко.

Они есть:

- гибкий
- Масштабируемость
- векторы
- Быстрая обработка
- Легкий вес
- доступной

Влияние на размер

Экспортирование изображения в различных размерах для устройств Android будет стоить вашему приложению, размер дополнительных активов - около 30 КБ на изображение. При добавлении файла шрифта (.ttf) с примерно 36 значками будет стоить всего 9 КБ. Представьте себе случай, если вы добавите 36 отдельных файлов различных конфигураций, это будет около 1000 КБ. Это разумное пространство, которое вы сэкономите с помощью значков.

Ограничения иконок.

- Значки шрифтов могут использоваться в навигационном ящике. Использование их в навигационных представлениях как значок пунктов меню невозможно, так как файл меню не может быть создан без указания заголовка. Поэтому рекомендуется использовать svg-файлы в качестве ресурсов для этих значков.

- Символьные шрифты нельзя использовать в кнопке с плавающим действием. поскольку они не имеют атрибута `setText()` .

- Внешние шрифты нельзя применять из xml. Они должны быть указаны с использованием java-файла. Или вам нужно расширить основной вид и создать представление, указанное в этом сообщении

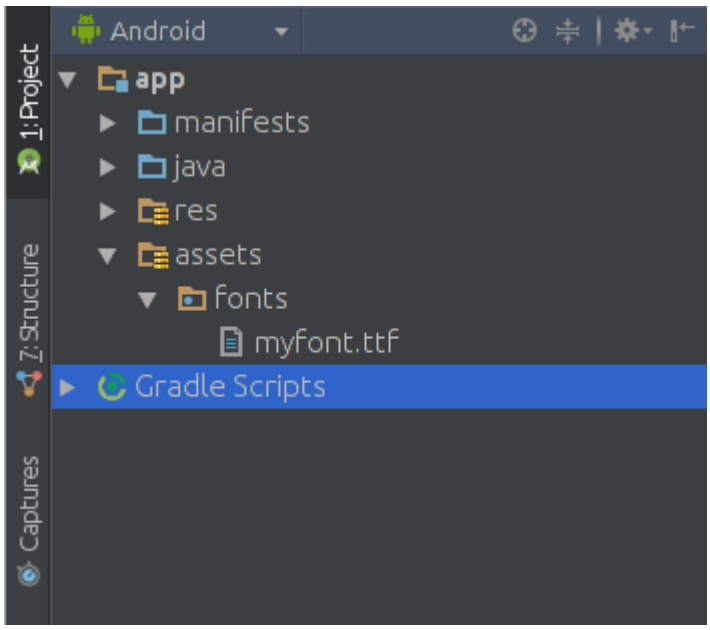
Examples

Как интегрировать знаковые шрифты

Чтобы использовать значки, выполните следующие действия:

- **Добавить файл шрифта в ваш проект**

  Вы можете создать файл значков шрифтов с интернет-сайтов, таких как icomoon , где вы можете загружать файлы SVG с необходимыми значками, а затем загружать созданный шрифт значка. Затем поместите *файл* шрифта *.ttf* в папку с именем *fonts* (укажите его как хотите) в папке с ресурсами:



- **Создание класса помощника**

  Теперь создайте следующий вспомогательный класс, чтобы избежать повторения кода инициализации шрифта:

```
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);
  }
}
```

  Вы можете использовать класс `Typeface` для выбора шрифта из активов. Таким образом, вы можете установить шрифт на различные виды, например, на кнопку:

```
Button button=(Button) findViewById(R.id.button);
Typeface iconFont=FontManager.getTypeface(getApplicationContext());
button.setTypeface(iconFont);
```

  Теперь шрифт кнопки был изменен на вновь созданный шрифт значка.

- **Подберите значки, которые вы хотите**

  Откройте файл *styles.css,* прикрепленный к шрифту значка. Там вы найдете стили с символами Unicode ваших значков:

```
.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";
}
```

  Этот файл ресурсов будет служить в качестве словаря, который сопоставляет символ Юникода, связанный с определенным значком, с человекочитаемым именем. Теперь создайте строковые ресурсы следующим образом:

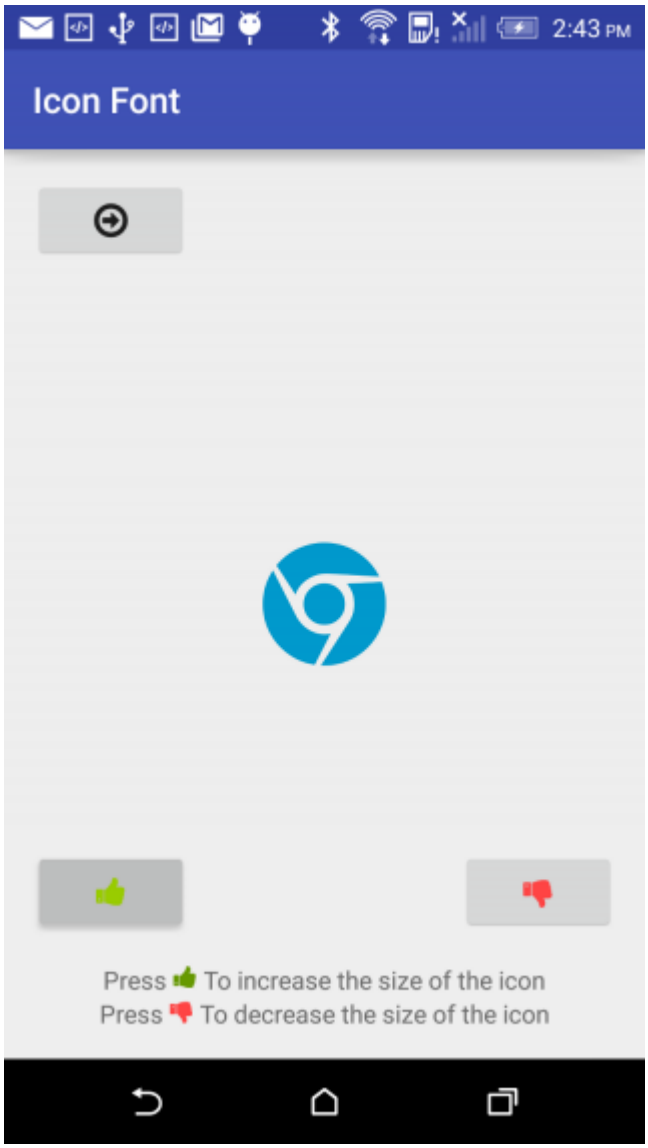
```
<resources>
  <! — Icon Fonts -->
  <string name="icon_arrow_circle_down">&#xe001; </string>
  <string name="icon_arrow_circle_left">&#xe002; </string>
  <string name="icon_arrow_circle-o_down">&#xe003; </string>
  <string name="icon_arrow_circle_o_left">&#xe004; </string>
</resources>
```

- **Используйте значки в коде.**

  Теперь вы можете использовать свой шрифт в разных представлениях, например, следующим образом:

```
button.setText(getString(R.string.icon_arrow_circle_left))
```

  Вы также можете создавать текстовые представления кнопок с помощью значков:



TabLayout с иконками шрифтов

```
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;
}

}
```

- В этом примере myfont.ttf находится в папке «Активы». Создание папки с активами
- В вашем классе активности

```
//..
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);
    //..
```

Прочитайте Улучшение производительности Android с помощью знаковых шрифтов онлайн: https://riptutorial.com/ru/android/topic/3642/улучшение-производительности-android-с-помощью-знаковых-шрифтов

глава 251: умысел

Вступление

Intent - это небольшое сообщение, переданное по системе Android. Это сообщение может содержать информацию о нашем намерении выполнить задачу.

Это в основном пассивная структура данных, содержащая абстрактное описание действия, которое необходимо выполнить.

Синтаксис

- Intent Intent ()
- Намерение намерения (намерение намерения)
- Intent Intent (String action)
- Intent Intent (String action, Uri uri)
- Intent Intent (Контекстный пакетContext, Class <?> Cls)
- Intent Intent (String action, Uri uri, Context packageContext, Class <?> Cls)
- void startActivity (намерение намерения)
- void startActivity (намерение намерения, опции пакета)
- void startActivityForResult (намерение намерения, int requestCode)
- void startActivityForResult (Intent intent, int requestCode, Bundle options)
- Intent putExtra (имя строки, double [])
- Intent putExtra (имя строки, значение int)
- Intent putExtra (имя строки, значение CharSequence)
- Intent putExtra (имя строки, значение char)
- Intent putExtra (имя строки, значение привязки)
- Intent putExtra (имя строки, значение Parcelable [])
- Intent putExtra (имя строки, значение Serializable)
- Intent putExtra (имя строки, значение int [])
- Intent putExtra (имя строки, значение поплавка)
- Intent putExtra (имя строки, байт [])
- Intent putExtra (имя строки, длинное значение [])
- Intent putExtra (Имя строки, Исходное значение)
- Intent putExtra (имя строки, значение float [])
- Intent putExtra (имя строки, длинное значение)
- Intent putExtra (имя строки, значение String [])
- Intent putExtra (имя строки, логическое значение)
- Intent putExtra (имя строки, значение boolean [])
- Intent putExtra (имя строки, короткое значение)
- Intent putExtra (имя строки, двойное значение)

- Intent putExtra (имя строки, короткое значение [])
- Intent putExtra (String name, String value)
- Intent putExtra (имя строки, значение байта)
- Intent putExtra (имя строки, значение char [])
- Intent putExtra (имя строки, значение CharSequence [])

параметры

параметр	подробности
намерение	Цель начать
код заявки	Уникальный номер для идентификации запроса
опции	Дополнительные параметры того, как следует запускать Activity
название	Название дополнительных данных
значение	Значение дополнительных данных
CHOOSE_CONTACT_REQUEST_CODE	код запроса, чтобы идентифицировать его по методу `onActivityResult`
действие	Любые действия для выполнения с помощью этого намерения, например: Intent.ACTION_VIEW
URI	данные uri, которые будут использоваться намерением для выполнения указанного действия
packageContext	Контекст, используемый для инициализации намерения
ЦБС	Класс, который будет использоваться этим намерением

замечания

Оговорки об использовании неявного

намерения

При вызове неявного намерения всегда полезно проверить, может ли система обрабатывать его.

Это можно сделать, проверив использование `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
}
```

Начальная активность, которая представляет собой `singleTask` или `singleTop`

Когда режим запуска активности - `singleTask` или `singleTop` , `onActivityResult` будет вызываться, как только начнется действие с нулевым значением. Чтобы предотвратить это, используйте `Intent.setFlags(0)` чтобы сбросить флаги по умолчанию.

Examples

Начало деятельности

В этом примере начнется `DestinationActivity` из `OriginActivity` .

Здесь конструктор `Intent` принимает два параметра:

1. Контекст как его первый параметр (это используется, потому что класс Activity является подклассом Context)
2. Класс компонента приложения, которому система должна доставлять намерение (в этом случае, деятельность, которая должна быть запущена)

```
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.
    }
}
```

Другой способ создания `Intent` для открытия `DestinationActivity` - использовать конструктор по умолчанию для `Intent` и использовать метод `setClass()` чтобы сообщить ему, какую активность открыть:

```
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
```

Передача данных между действиями

Этот пример иллюстрирует отправку `String` со значением как `"Some data!"` от `OriginActivity` до `DestinationActivity` .

ПРИМЕЧАНИЕ. Это самый простой способ отправки данных между двумя действиями. См. Пример использования шаблона стартера для более надежной реализации.

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);
    }
}
```

Также возможно передавать другие `primitive` типы данных, а также `arrays` , данные `Bundle` и `Parcelable` . Передача `Serializable` также возможна, но ее следует избегать, поскольку она более чем в три раза медленнее, чем `Parcelable` .

Serializable - это стандартный `interface` Java. Вы просто отмечаете класс как `Serializable` , реализуя `interface Serializable` и Java автоматически сериализует его во время необходимых ситуаций.

Parcelable - это специфический для Android `interface` который может быть реализован на пользовательских типах данных (то есть на ваших собственных объектах / объектах POJO), что позволяет сглаживать и восстанавливать ваш объект без назначения, которое нужно что-либо сделать. Существует пример документации, позволяющий сделать объект более простым .

Как только у вас есть `parcelable` объект, вы можете отправить его как примитивный тип с целевым объектом:

```
intent.putExtra(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

Или в связке / в качестве аргумента для фрагмента:

```
bundle.putParcelable(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

а затем также прочитать его с намерением в пункте назначения, используя getParcelableExtra:

```
final MyParcelableType data = intent.getParcelableExtra(EXTRA_DATA);
```

Или при чтении в фрагменте из пакета:

```
final MyParcelableType data = bundle.getParcelable(EXTRA_DATA);
```

Когда у вас есть объект `Serializable` вы можете поместить его в объект намерения:

```
bundle.putSerializable(DestinationActivity.EXTRA_DATA, mySerializableObject);
```

а затем также прочитать его из объекта намерения в пункте назначения, как показано ниже:

```
final SerializableType data = (SerializableType)bundle.getSerializable(EXTRA_DATA);
```

Отправка писем

```
// 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
}
```

Это предварительно заполнит письмо в почтовом приложении по выбору пользователя.

Если вам нужно добавить вложение, вы можете использовать `Intent.ACTION_SEND` вместо `Intent.ACTION_SENDTO` . Для нескольких вложений вы можете использовать `ACTION_SEND_MULTIPLE`

`ACTION_SENDTO` : не у каждого устройства есть провайдер `ACTION_SENDTO` и вызов функции `startActivity()` без проверки с помощью `resolveActivity()` может сначала вызвать `resolveActivity()` ActivityNotFoundException.

Получение результата от другой деятельности

Используя `startActivityForResult(Intent intent, int requestCode)` вы можете начать другую `Activity` , а затем получить результат от этой `Activity` в `onActivityResult(int requestCode, int resultCode, Intent data)` метод. Результат будет возвращен как `Intent` . Умысел может содержать данные через Bundle

---

В этом примере `MainActivity` запустит `DetailActivity` а затем ожидает от него результата. Каждый тип запроса должен иметь свой собственный `int` код запроса, так что в *перегруженной* `onActivityResult(int requestCode, int resultCode, Intent data)` метод `MainActivity` , можно определить , какой запрос обрабатывать путем сравнения значений

`requestCode` и `REQUEST_CODE_EXAMPLE` (хотя в этом Например, есть только один).

Основная деятельность:

```
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();
    }
}
```

Несколько вещей, о которых вы должны знать:

- Данные возвращаются только после вызова `finish()` . Вам нужно вызвать `setResult()` перед вызовом `finish()` , в противном случае результат не будет возвращен.
- Убедитесь, что ваша `Activity` не использует `android:launchMode="singleTask"` , или это приведет к тому, что `Activity` будет выполняться в отдельной задаче, и поэтому вы не получите результат от нее. Если ваша `Activity` использует `singleTask` качестве режима запуска, она немедленно вызовет `onActivityResult()` с результирующим кодом `Activity.RESULT_CANCELED` .
- Будьте осторожны при использовании `android:launchMode="singleInstance"` . На устройствах до Lollipop (Android 5.0, API Level 21) действия не возвращают результат.
- Вы можете использовать явные или неявные намерения, когда вы вызываете `startActivityForResult()` . При запуске одного из ваших собственных действий для

получения результата вы должны использовать явное намерение обеспечить получение ожидаемого результата. Явное `intent` всегда передается его цели, независимо от того, что она содержит; `filter` не проконсультируется. Но неявное намерение доставляется компоненту только в том случае, если он может проходить через один из фильтров компонента.

Открыть URL-адрес в браузере

Открытие браузера по умолчанию

В этом примере показано, как вы можете открывать URL-адрес программно во встроенном веб-браузере, а не в своем приложении. Это позволяет вашему приложению открывать веб-страницу без необходимости включения разрешения `INTERNET` в файл манифеста.

```
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);
    }
}
```

Вызов пользователя для выбора браузера

Обратите внимание, что в этом примере используется метод `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"));

}
```

В некоторых случаях URL-адрес может начинаться с **«www»** . Если это так, вы получите это исключение:

> `android.content.ActivityNotFoundException` : активности не найдено для обработки Intent

URL-адрес должен всегда начинаться с **«http: //»** или **«https: //»** . Поэтому ваш код должен проверять его, как показано в следующем фрагменте кода:

```
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);
}
```

Лучшие практики

Проверьте, нет ли на устройстве приложений, которые могут получить неявное намерение. В противном случае ваше приложение выйдет из `startActivity()` при `startActivity()` . Чтобы в первую очередь проверить, существует ли приложение для получения намерения, вызовите `resolveActivity()` для вашего объекта Intent. Если результат не равен нулю, существует хотя бы одно приложение, которое может обрабатывать намерение, и можно безопасно вызвать `startActivity()` . Если результат равен нулю, вы не должны использовать намерение и, если возможно, вы должны отключить функцию, которая вызывает намерение.

Очистка стека действий

Иногда вы можете начать новое действие, удаляя предыдущие действия из заднего стека, поэтому кнопка «Назад» не возвращает вас к ним. Одним из примеров этого может быть запуск приложения в активности входа, переход к основному действию вашего приложения, но при выходе из системы вы хотите вернуться в Login без возможности вернуться. В таком случае вы можете установить флаг `FLAG_ACTIVITY_CLEAR_TOP` для намерения, то есть, если запущенная деятельность уже запущена в текущей задаче (LoginActivity), то вместо запуска нового экземпляра этого действия все остальные действия сверху он будет закрыт, и это намерение будет передано (теперь сверху) старой деятельности в качестве нового намерения.

```
Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
```

Также можно использовать флаги `FLAG_ACTIVITY_NEW_TASK` вместе с `FLAG_ACTIVITY_CLEAR_TASK` если вы хотите очистить все действия в `FLAG_ACTIVITY_CLEAR_TASK` стеке:

```
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 URI

В этом примере показано, как начать умываться из браузера:

```
<a href="intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end">Start
intent</a>
```

Это намерение запустит приложение с пакетом `com.sample.test` или откроет игру Google с этим пакетом.

Также это намерение можно запустить с помощью javascript:

```
var intent = "intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end";
window.location.replace(intent)
```

В действии этот хост и путь могут быть получены из данных о намерениях:

```
@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    Uri data = getIntent().getData(); // returns host.com/path
}
```

Синтаксис URI Intent:

```
HOST/URI-path // Optional host
#Intent;
    package=[string];
    action=[string];
    category=[string];
    component=[string];
    scheme=[string];
end;
```

Передача сообщений другим компонентам

Intents могут использоваться для передачи сообщений другим компонентам вашего приложения (например, работающей фоновой службе) или ко всей системе Android.

Чтобы отправить широковещательную рассылку **в своем приложении** , используйте класс `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);
```

Чтобы отправить трансляцию компонентам за пределами вашего приложения, используйте метод `sendBroadcast()` для объекта `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);
```

Информацию о *получении* трансляций можно найти здесь: Broadcast Receiver

CustomTabsIntent для пользовательских вкладок Chrome

4.0.3

Используя `CustomTabsIntent` , теперь можно настроить пользовательские вкладки Chrome , чтобы настроить ключевые компоненты пользовательского интерфейса в браузере, который открывается из вашего приложения.

Это хорошая альтернатива использованию WebView для некоторых случаев. Это позволяет загружать веб-страницу с помощью Intent с добавленной способностью вставлять некоторую степень внешнего вида вашего приложения в браузер.

Вот пример того, как открыть URL-адрес с помощью `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));
```

Замечания:

Чтобы использовать пользовательские вкладки, вам нужно добавить эту зависимость в свой build.gradle

```
compile 'com.android.support:customtabs:24.1.1'
```

Совместное использование нескольких файлов с помощью намерения

Список строк, переданный как параметр для метода `share()` содержит пути всех файлов, которые вы хотите разделить.

Он в основном проходит через пути, добавляет их в Uri и запускает Activity, который может принимать файлы этого типа.

```
  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);
    }
```

Стартовый шаблон

Этот шаблон является более строгим подходом к началу `Activity` . Его цель - улучшить читаемость кода, в то же время уменьшить сложность кода, затраты на обслуживание и соединение ваших компонентов.

В следующем примере реализуется шаблон стартера, который обычно реализуется как статический метод для самой `Activity` . Этот статический метод принимает все необходимые параметры, строит действительный `Intent` из этих данных и затем запускает `Activity` .

`Intent` - это объект, который обеспечивает привязку времени выполнения между отдельными компонентами, такими как два действия. Intent представляет собой «намерение сделать что-то». Вы можете использовать намерения для самых разных задач, но здесь ваше намерение начинает другую деятельность.

```
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);
    }
}
```

Этот шаблон также позволяет принудительно передавать дополнительные данные с

намерением.

Затем `ExampleActivity` может быть запущен следующим образом: `context` активности:

```
ExampleActivity.start(context, "Some data!");
```

Запуск несвязанной службы с использованием намерения

Служба - это компонент, который работает в фоновом режиме (в потоке пользовательского интерфейса) без прямого взаимодействия с пользователем. Несвязанная служба только начинается и не привязана к жизненному циклу любой Деятельности.

Чтобы запустить службу, вы можете сделать это, как показано в следующем примере:

```
// 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);
```

Вы можете использовать любые дополнения из намерения с помощью переопределения `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;
    }

}
```

Совместное намерение

Поделитесь простой информацией с различными приложениями.

```
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)));
```

Поделитесь изображением с различными приложениями.

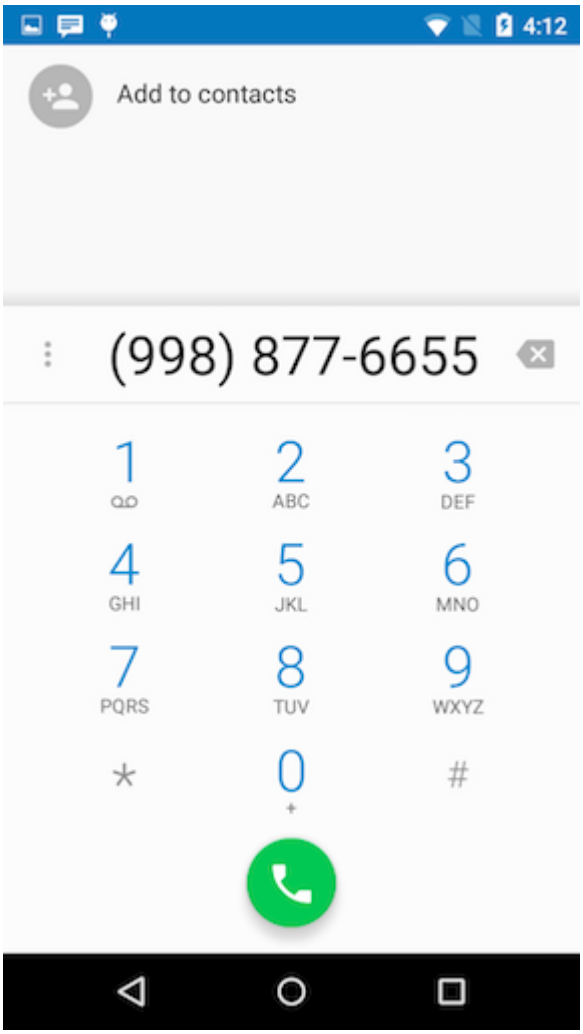
```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uriToImage);
shareIntent.setType("image/jpeg");
startActivity(Intent.createChooser(shareIntent, getResources().getText(R.string.send_to)));
```

Запуск дозвона

В этом примере показано, как открыть набор номера по умолчанию (приложение, которое выполняет регулярные вызовы), с предоставленным номером телефона уже на месте:

```
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);
```

Результат выполнения приведенного выше кода:



Откройте карту Google с указанной широтой, долготой

Вы можете передавать широту, долготу из вашего приложения на карту Google, используя 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);
```

Передача разных данных через Intent in Activity

1. Передача целочисленных данных:

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. Передача двойных данных:

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. Передача данных строки:

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");
```

или же

```
Intent mIntent = getIntent();
String stringValue = mIntent.getStringExtra("stringVariableName");
```

4. Передача данных 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. Передача данных объекта:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("ObjectVariableName", yourObject);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
yourObj = mIntent.getSerializableExtra("ObjectVariableName");
```

> **Примечание.** Имейте в виду, что ваш пользовательский класс должен реализовывать интерфейс `Serializable` .

6. Передача данных 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. Передача данных растрового изображения:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("image",bitmap);
startActivity(mIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
Bitmap bitmap = mIntent.getParcelableExtra("image");
```

Отображение выбора файла и чтение результата

Запуск операции выбора файла

```
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);
}
```

Чтение результата

```
@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;
}
```

Передача пользовательского объекта между действиями

Также можно передать свой пользовательский объект другим действиям, используя класс `Bundle` .

Существует два способа:

- `Serializable` интерфейс - для Java и Android
- `Parcelable` интерфейс-память эффективна, только для Android (рекомендуется)

Parcelable

Поэтапная обработка намного быстрее, чем сериализуемая. Одной из причин этого является то, что мы говорим о процессе сериализации вместо использования рефлексии для его вывода. Также понятно, что для этой цели код был сильно оптимизирован.

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

Код операции отправки

```
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);
```

Получение объекта в операции назначения.

```
//Getting MyObjects
Intent mIntent = getIntent();
MyObjects workorder = (MyObjects) mIntent.getParcelable("UniqueKey");
```

Вы можете передать объект Arraylist of Parceble, как показано ниже

```
//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");
```

> **Примечание.** Плагины Android Studio, такие как этот, доступны для создания кода Parcelable

Сериализуемый

Код операции отправки

```
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);
```

Получение объекта в операции назначения.

```
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` объекта Serializable: тот же, что и для передачи одного объекта

Пользовательский объект должен реализовывать интерфейс `Serializable` .

Получение результата из Activity to Fragment

Как и получение результата от другой `startActivityForResult(Intent intent, int requestCode)` вам нужно вызвать метод `Fragment` `startActivityForResult(Intent intent, int requestCode)` . обратите внимание, что вы не должны вызывать `getActivity().startActivityForResult()` как это возвращает результат к родительской `Activity` `Fragment` .

Получение результата может быть сделано с использованием `Fragment` метода «ы `onActivityResult()` . Вы должны убедиться, что родительская активность фрагмента также переопределяет `onActivityResult()` и называет ее `super` реализацией.

В следующем примере `ActivityOne` содержит `FragmentOne` , который запустит `ActivityTwo` и ожидает от него результата.

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();
    }
}
```

Прочитайте умысел онлайн: https://riptutorial.com/ru/android/topic/103/умысел

глава 252: Универсальный загрузчик изображений

замечания

Допустимые примеры URI:

```
"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

Инициализировать универсальный загрузчик изображений

1. Добавьте следующую зависимость в файл *build.gradle* :

   ```
   compile 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
   ```

2. Добавьте в файл *AndroidManifest.xml* следующие разрешения:

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

3. Инициализируйте универсальный загрузчик изображений. Это необходимо сделать до первого использования:

   ```
   ImageLoaderConfiguration config = new ImageLoaderConfiguration.Builder(this)
       // ...
       .build();
   ImageLoader.getInstance().init(config);
   ```

   Полные параметры конфигурации можно найти здесь .

Основное использование

1. Загрузите изображение, декодируйте его в растровое изображение и покажите растровое изображение в `ImageView` (или любом другом представлении, которое реализует интерфейс `ImageAware`):

```
ImageLoader.getInstance().displayImage(imageUri, imageView);
```

2. Загрузите изображение, декодируйте его в растровое изображение и верните растровое изображение в обратный вызов:

```
ImageLoader.getInstance().loadImage(imageUri, new SimpleImageLoadingListener() {
    @Override
    public void onLoadingComplete(String imageUri, View view, Bitmap loadedImage) {
        // Do whatever you want with the bitmap.
    }
});
```

3. Загрузите изображение, декодируйте его в растровое изображение и сразу же верните растровое изображение:

```
Bitmap bmp = ImageLoader.getInstance().loadImageSync(imageUri);
```

Прочитайте Универсальный загрузчик изображений онлайн:
https://riptutorial.com/ru/android/topic/2760/универсальный-загрузчик-изображений

глава 253: Установка приложений с помощью ADB

Examples

Установка приложения

Напишите в терминале следующую команду:

```
adb install [-rtsdg] <file>
```

Обратите внимание, что вам необходимо передать файл, который находится на вашем компьютере, а не на вашем устройстве.

Если вы добавите `-r` в конце, то любые существующие конфликтующие апки будут перезаписаны. В противном случае команда выйдет с ошибкой.

`-g` немедленно предоставит все разрешения во время выполнения.

`-d` позволяет понизить версию кода (только применительно к отлаживаемым пакетам).

Используйте `-s` для установки приложения на внешнюю SD-карту.

`-t` позволит использовать тестовые приложения.

Удаление приложения

Напишите на своем компьютере следующую команду для удаления приложения с предоставленным именем пакета:

```
adb uninstall <packagename>
```

Установить все файлы apk в каталог

Windows:

```
for %f in (C:\your_app_path\*.apk) do adb install "%f"
```

Linux:

```
for f in *.apk ; do adb install "$f" ; done
```

Прочитайте Установка приложений с помощью ADB онлайн:

https://riptutorial.com/ru/android/topic/5301/установка-приложений-с-помощью-adb

глава 254: Утечки памяти

Examples

Общие утечки памяти и способы их устранения

1. Исправьте свои контексты:

Попробуйте использовать соответствующий контекст: например, поскольку тост можно увидеть во многих действиях, а не только в одном, используйте `getApplicationContext()` для тостов, и поскольку службы могут продолжать работать, даже если действие закончилось, запустите службу с помощью:

```
Intent myService = new Intent(getApplicationContext(), MyService.class);
```

Используйте эту таблицу в качестве краткого руководства для контекста:

	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	

Оригинальная статья о контексте здесь .

2. Статическая ссылка на контекст

Серьезной ошибкой памяти является сохранение статической ссылки на `View` . Каждый `View` имеет внутреннюю ссылку на `Context` . Это означает, что старая активность со всей иерархией представления не будет собираться мусором, пока приложение не будет прекращено. При повороте экрана у вас будет ваше приложение дважды в памяти.

Убедитесь, что нет никаких статических ссылок на View, Context или их потомков.

3. Убедитесь, что вы фактически завершаете свои услуги.

Например, у меня есть намеренная служба, которая использует API службы местоположения Google. И я забыл называть `googleApiClient.disconnect();` :

```
//Disconnect from API onDestroy()
if (googleApiClient.isConnected()) {
    LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient,
GoogleLocationService.this);
    googleApiClient.disconnect();
}
```

4. Проверьте использование изображений и растровых изображений:

Если вы используете **библиотеку** Square **Picasso,** я обнаружил, что я `.fit()` память, не используя `.fit()` , что резко сократило объем памяти с 50 МБ в среднем до менее 19 МБ:

```
Picasso.with(ActivityExample.this)                    //Activity context
                .load(object.getImageUrl())
                .fit()                                 //This avoided the OutOfMemoryError
                .centerCrop()                          //makes image to not stretch
                .into(imageView);
```

5. Если вы используете широковещательные приемники, отмените их регистрацию.

6. Если вы используете `java.util.Observer` (шаблон наблюдателя):

Обязательно используйте `deleteObserver(observer);`

Избегайте утечки активности с помощью AsyncTask

> **Слово предостережения:** AsyncTask имеет много Гочи обособленно от утечки памяти , описанной здесь. Поэтому будьте осторожны с этим API или избегайте его вообще, если вы не полностью понимаете последствия. Существует много альтернатив (Thread, EventBus, RxAndroid и т. Д.).

Одна из распространенных ошибок с `AsyncTask` заключается в том, чтобы зафиксировать сильную ссылку на `Activity` хоста (или `Fragment`):

```
class MyActivity extends Activity {
  private AsyncTask<Void, Void, Void> myTask = new AsyncTask<Void, Void, Void>() {
    // Don't do this! Inner classes implicitly keep a pointer to their
    // parent, which in this case is the Activity!
  }
}
```

Это проблема, потому что `AsyncTask` может легко пережить родительскую `Activity` , например, если во время выполнения задачи происходит изменение конфигурации.

Правильный способ сделать это - сделать вашу задачу `static` классом, который не захватывает родителя и содержит слабую ссылку на хост. `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
    }
  }
}
```

Анонимный обратный вызов в действиях

Каждый раз, когда вы создаете анонимный класс, он сохраняет неявную ссылку на свой родительский класс. Поэтому, когда вы пишете:

```
public class LeakyActivity extends Activity
{

...

    foo.registerCallback(new BarCallback()
    {
```

```
        @Override
        public void onBar()
        {
            // do something
        }
    });
}
```

Фактически вы отправляете ссылку на ваш экземпляр LeakyActivity для foo. Когда пользователь переходит от вашей LeakyActivity, эта ссылка может помешать экземпляру LeakyActivity быть собранным мусором. Это серьезная утечка, поскольку действия содержат ссылку на всю их иерархию представлений и, следовательно, являются довольно большими объектами в памяти.

Как избежать этой утечки:

Конечно, вы можете избежать анонимных обратных вызовов в действиях. Вы также можете отменить регистрацию всех ваших обратных вызовов в отношении жизненного цикла деятельности. вот так:

```
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);
    }
}
```

Контекст активности в статических классах

Часто вы захотите перенести некоторые классы Android в более простые в использовании служебные классы. Эти классы полезности часто требуют контекста для доступа к ресурсам Android и вашим приложениям. Общим примером этого является оболочка для класса SharedPreferences. Чтобы получить доступ к общим настройкам Androids, необходимо написать:

`context.getSharedPreferences(prefsName, mode);`

И поэтому может возникнуть соблазн создать следующий класс:

```
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);
    }
}
```

теперь, если вы вызываете `init()` с вашим контекстом активности, LeakySharedPrefsWrapper сохранит ссылку на вашу деятельность, не позволяя ей собирать мусор.

Как избежать:

При вызове статических вспомогательных функций вы можете отправить контекст приложения, используя `context.getApplicationContext();`

При создании статических вспомогательных функций вы можете извлечь контекст приложения из предоставленного вами контекста (вызов метода getApplicationContext () в контексте приложения возвращает контекст приложения). Итак, исправление нашей обертки просто:

```
public static void init(Context context)
{
    sContext = context.getApplicationContext();
}
```

Если контекст приложения не подходит для вашего варианта использования, вы можете включить параметр Context в каждую функцию утилиты, вы должны избегать ссылок на эти параметры контекста. В этом случае решение будет выглядеть так:

```
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);
}
```

Обнаружение утечек памяти с помощью библиотеки LeakCanary

LeakCanary - это библиотека Java с открытым исходным кодом для обнаружения утечек памяти в ваших отладочных сборках.

Просто добавьте зависимости в `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'
 }
```

Затем в вашем классе `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);
  }
}
```

Теперь LeakCanary автоматически отобразит уведомление, когда в вашей **отладочной** сборке обнаружена утечка памяти активности.

ПРИМЕЧАНИЕ. Код деблокирования не будет содержать ссылки на LeakCanary, кроме двух пустых классов, которые существуют в `leakcanary-android-no-op` .

Избегайте утечки активности с помощью прослушивателей

Если вы реализуете или создаете слушателя в Activity, всегда обращайте внимание на жизненный цикл объекта, на котором зарегистрирован слушатель.

Рассмотрим приложение, в котором у нас есть несколько различных действий / фрагментов, заинтересованных в том, что пользователь вошел в систему или вышел из системы. Одним из способов сделать это было бы иметь экземпляр singleton `UserController` который можно подписаться, чтобы получить уведомление при изменении состояния пользователя:

```
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();
    }
}
```

Затем есть два действия: `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();
    }
```

```
}
```

И `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();
    }
}
```

Что происходит с этим примером, так это то, что каждый раз, когда пользователь входит в систему, а затем выходит из системы снова, экземпляр `MainActivity` просачивается. Утечка происходит из-за ссылки на активность в `UserController#listeners` .

Обратите внимание: даже если мы используем анонимный внутренний класс в качестве слушателя, активность все равно будет протекать:

```
...
this.userController.registerUserStateChangeListener(new UserController.StateListener() {
    @Override
    public void userLoggedIn() {
        showUserAccount();
    }

    @Override
    public void userLoggedOut() {
        finish();
    }
});
...
```

Активность все равно будет протекать, потому что анонимный внутренний класс имеет неявную ссылку на внешний класс (в данном случае активность). Вот почему во внешнем

классе можно вызвать методы экземпляра из внутреннего класса. Фактически, единственным типом внутренних классов, которые не имеют ссылки на внешний класс, являются **статические** внутренние классы.

Короче говоря, все экземпляры нестатических внутренних классов содержат неявную ссылку на экземпляр внешнего класса, который их создал.

Существует два основных подхода к решению этого вопроса, добавив метод удаления слушателя из прослушивателей `UserController#listeners` или используя `WeakReference` для хранения ссылки слушателей.

Альтернатива 1: Удаление слушателей

Начнем с создания нового метода `removeUserStateChangeListener(StateListener listener)` :

```
public class UserController {

    ...

    public void registerUserStateChangeListener(StateListener listener) {
        listeners.add(listener);
    }

    public void removeUserStateChangeListener(StateListener listener) {
        listeners.remove(listener);
    }

    ...
}
```

Затем назовем этот метод в методе `onDestroy` активности:

```
public class MainActivity extends Activity implements UserController.StateListener{
    ...

    @Override
    protected void onDestroy() {
        super.onDestroy();
        userController.removeUserStateChangeListener(this);
    }
}
```

С этой модификацией экземпляры `MainActivity` больше не просачиваются, когда пользователь входит в систему и выходит из `MainActivity` . Однако, если документация не ясна, возможно, что следующий разработчик, который начинает использовать `UserController` может пропустить, что требуется отменить регистрацию слушателя, когда действие будет уничтожено, что приведет нас к второму методу предотвращения этих типов утечек.

Альтернатива 2: Использование слабых ссылок

Прежде всего, давайте начнем с объяснения того, что такое слабая ссылка. Слабая ссылка, как следует из названия, содержит слабую ссылку на объект. По сравнению с обычным полем экземпляра, которое является сильной ссылкой, слабые ссылки не останавливают сборщик мусора, GC, от удаления объектов. В приведенном выше примере это позволит `MainActivity` быть собранным с мусором после его уничтожения, если `UserController` использовал `WeakReference` для ссылки на слушателей.

Короче говоря, слабая ссылка говорит GC, что, если никто другой не имеет сильной ссылки на этот объект, продолжайте и удалите его.

Давайте `WeakReference` `UserController` чтобы использовать список `WeakReference` чтобы отслеживать его слушателей:

```
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);
            }
        }
```

```
    }
    ...
}
```

С этой модификацией не имеет значения, удалены ли слушатели или нет, поскольку `UserController` содержит сильных ссылок на любого из слушателей. Однако писать этот шаблонный код каждый раз является громоздким. Поэтому создадим общий класс под названием `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);
        }
    }
}
```

Теперь давайте `WeakCollection<T> UserController` вместо `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();
        }
    }

    ...
}
```

Как показано в приведенном выше примере кода, `WeakCollection<T>` удаляет весь шаблонный код, необходимый для использования `WeakReference` вместо обычного списка. В довершение всего: если вызов `UserController#removeUserStateChangeListener(StateListener)` пропущен, слушатель и все объекты, на которые он ссылается, не будут течь.

Избегайте утечек памяти с помощью анонимного класса, обработчика, задачи таймера, потока

В android каждый разработчик использует `Anonymous Class` (Runnable) хотя бы один раз в проекте. Любой `Anonymous Class` имеет ссылку на родителя (активность). Если мы выполним долговременную задачу, родительская активность не будет уничтожена до завершения задачи.
В примере используется класс обработчика и Анонимный `Runnable` . Память будет протекать, когда мы прекратим действие до завершения `Runnable` .

```
  new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            // do abc long 5s or so
        }
    }, 10000); // run "do abc" after 10s. It same as timer, thread...
```

Как мы его решаем?

1. Не делайте никаких `WeakReference` с `Anonymous Class` или нам нужен `Static class` и `WeakReference` в него `WeakReference` (например, активность, просмотр ...). `Thread` же с `Anonymous Class` .
2. Отмените `Handler` , `Timer` когда действие уничтожено.

Прочитайте Утечки памяти онлайн: https://riptutorial.com/ru/android/topic/2687/утечки-памяти

глава 255: Учетные записи и учетные записи

Examples

Понимание пользовательских учетных записей / аутентификация

Следующий пример - это высокий уровень охвата ключевых концепций и базовой скелетной установки: -

1. Собирает учетные данные от пользователя (обычно с созданного вами экрана входа в систему)
2. Аутентификация учетных данных с сервером (сохранение пользовательской аутентификации)
3. Сохраняет учетные данные на устройстве

Расширьте AbstractAccountAuthenticator *(прежде всего, для получения аутентификации и повторной проверки подлинности)*

```
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.
  }
}
```

Создайте службу *(Framework Account Manager подключается к расширенному AbstractAccountAuthenticator через интерфейс службы)*

```
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();
    }
}
```

Конфигурация XML аутентификатора *(требуется инфраструктура менеджера учетных записей. Это то, что вы увидите внутри настроек -> Учетные записи в 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" />
```

Изменения в AndroidManifest.xml *(объедините все вышеуказанные концепции, чтобы сделать его полезным для использования через 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>
```

В следующем примере будет показано, как использовать эту настройку.

Прочитайте Учетные записи и учетные записи онлайн:
https://riptutorial.com/ru/android/topic/7003/учетные-записи-и-учетные-записи

глава 256: Файл манифеста

Вступление

Манифест является обязательным файлом с именем точно «AndroidManifest.xml» и находится в корневом каталоге приложения. Он определяет имя приложения, значок, имя пакета Java, версию, объявление о действиях, службах, разрешениях приложений и другую информацию.

Examples

Объявление компонентов

Основная задача манифеста - информировать систему о компонентах приложения. Например, файл манифеста может объявлять действие следующим образом:

```
<?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>
```

В элементе `<application>` атрибут `android:icon` указывает на ресурсы для значка, который идентифицирует приложение.

В элементе атрибут `android:name` указывает полное имя класса подкласса Activity, а атрибут android: label указывает строку, используемую в качестве видимой пользователем метки для этой активности.

Вы должны объявить все компоненты приложения таким образом:

- `<activity>` элементов для деятельности

- `<service>` элементы для служб

- элементы `<receiver>` для широковещательных приемников

- `<provider>` элементы для поставщиков контента

Службы, службы и поставщики контента, которые вы включаете в свой источник, но не объявляете в манифесте, не видны системе и, следовательно, никогда не могут работать. Однако широковещательные приемники могут быть либо объявлены в манифесте, либо

созданы динамически в коде (как объекты `BroadcastReceiver`) и зарегистрированы в системе, вызвав `registerReceiver()` .

Подробнее о том, как структурировать файл манифеста для вашего приложения, см. В документации на AndroidManifest.xml.

Объявление разрешений в файле манифеста

Любые разрешения, требуемые вашим приложением для доступа к защищенной части API или для взаимодействия с другими приложениями, должны быть объявлены в вашем файле `AndroidManifest.xml` . Это делается с помощью `<uses-permission />` .

Синтаксис

```
<uses-permission android:name="string"
   android:maxSdkVersion="integer"/>
```

android: name: Это имя требуемого разрешения

android: maxSdkVersion: самый высокий уровень API, на котором это разрешение должно быть предоставлено вашему приложению. Установка этого разрешения является необязательной и должна устанавливаться только в том случае, если требуется разрешение, требуемое вашему приложению на определенном уровне API.

Пример 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>
```

* Также см. Раздел « Разрешения» .

Прочитайте Файл манифеста онлайн: https://riptutorial.com/ru/android/topic/1848/файл-манифеста

глава 257: Форматирование строк

Examples

Форматирование строкового ресурса

Вы можете добавить подстановочные знаки в строковых ресурсах и заполнить их во время выполнения:

1. Изменить strings.xml

```
<string name="my_string">This is %1$s</string>
```

2. При необходимости форматируйте строку

```
String fun = "fun";
context.getString(R.string.my_string, fun);
```

Отформатируйте метку времени в строку

Полное описание шаблонов см. В разделе справки SimpleDateFormat

```
Date now = new Date();
long timestamp = now.getTime();
SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy", Locale.US);
String dateStr = sdf.format(timestamp);
```

Форматирование типов данных в String и наоборот

Типы данных для форматирования строк

Типы данных, такие как int, float, double, long, boolean, могут быть отформатированы в строку с использованием 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"
```

И наоборот, форматирование строки для другого типа данных

```
Integer.parseInt("1"); //Output -> 1
Float.parseFloat("1.2"); //Output -> 1.2
Boolean.parseBoolean("true"); //Output -> true
```

Прочитайте Форматирование строк онлайн: https://riptutorial.com/ru/android/topic/1346/

форматирование-строк

глава 258: Форматирование телефонных номеров с рисунком.

Вступление

В этом примере показано, как форматировать номера телефонов с помощью patter

Вам понадобится следующая библиотека в градиенте.

compile 'com.googlecode.libphonenumber: libphonenumber: 7.2.2'

Examples

Шаблоны + 1 (786) 1234 5678

Учитывая нормализованный номер телефона, например +178612345678, мы получим отформатированное число с предоставленным шаблоном.

```
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;
}
```

Прочитайте Форматирование телефонных номеров с рисунком. онлайн: https://riptutorial.com/ru/android/topic/9083/форматирование-телефонных-номеров-с-рисунком-

глава 259: Фрагменты

Вступление

Введение об фрагментах и их механизм взаимодействия

Синтаксис

- void onActivityCreated (Bundle savedInstanceState) // Вызывается, когда была создана активность фрагмента и создана иерархия представлений этого фрагмента.
- void onActivityResult (int requestCode, int resultCode, Intent data) // Получать результат из предыдущего вызова startActivityForResult (Intent, int).
- void onAttach (активность активности) // Этот метод устарел на уровне API 23. Вместо этого используйте onAttach (Context).
- void onAttach (контекст контекста) // Вызывается, когда фрагмент сначала привязан к его контексту.
- void onAttachFragment (фрагмент childFragment) // Вызывается, когда фрагмент прикреплен как дочерний элемент этого фрагмента.
- void onConfigurationChanged (Конфигурация newConfig) // Вызывается системой при изменении конфигурации устройства во время работы вашего компонента.
- void onCreate (Bundle savedInstanceState) // Вызывается для первоначального создания фрагмента.
- Просмотр onCreateView (LayoutInflater inflater, контейнер ViewGroup, Bundle savedInstanceState) // Вызывается, чтобы фрагмент создавал экземпляр своего пользовательского интерфейса.
- void onDestroy () // Вызывается, когда фрагмент больше не используется.
- void onDestroyView () // Вызывается, когда представление, ранее созданное onCreateView (LayoutInflater, ViewGroup, Bundle), было отделено от фрагмента.
- void onDetach () // Вызывается, когда фрагмент больше не привязан к его активности.
- void onInflate (активность активности, AttributeSet attrs, Bundle savedInstanceState) // Этот метод был устаревшим на уровне API 23. Вместо этого используйте onInflate (Context, AttributeSet, Bundle).
- void onInflate (контекст контекста, AttributeSet attrs, Bundle savedInstanceState) //

Вызывается, когда фрагмент создается как часть инфляции макета представления, как правило, из настройки представления контента для действия.

- void onPause () // Вызывается, когда фрагмент больше не возобновляется.
- void onResume () // Вызывается, когда фрагмент виден пользователю и активно работает.
- void onSaveInstanceState (Bundle outState) // Вызывается, чтобы попросить фрагмент сохранить его текущее динамическое состояние, чтобы впоследствии его можно было восстановить в новом экземпляре его процесса.
- void onStart () // Вызывается, когда фрагмент отображается пользователю.
- void onStop () // Вызывается, когда фрагмент больше не запускается.
- void onViewStateRestored (Bundle savedInstanceState) // Вызывается, когда все сохраненное состояние было восстановлено в иерархию представления фрагмента.

замечания

Фрагмент представляет собой поведение или часть пользовательского интерфейса в Activity. Вы можете объединить несколько фрагментов в одном действии для создания многоуровневого пользовательского интерфейса и повторного использования фрагмента в нескольких действиях. Вы можете представить фрагмент как модульный раздел активности, который имеет свой собственный жизненный цикл, получает свои собственные входные события и которые вы можете добавлять или удалять во время работы (вроде как «вспомогательная активность», которую вы можете повторное использование в разных видах деятельности).

Конструктор

Каждый фрагмент должен иметь **пустой конструктор** , поэтому его можно создать при восстановлении состояния своей активности. Настоятельно рекомендуется, чтобы в подклассах не было других конструкторов с параметрами, поскольку эти конструкторы не будут вызываться, когда фрагмент будет повторно создан; вместо этого аргументы могут предоставляться вызывающим абонентом с помощью setArguments (Bundle), а затем извлекаются фрагментом с помощью getArguments ().

Examples

Шаблон newInstance ()

Хотя можно создать конструктор фрагментов с параметрами, Android внутренне вызывает конструктор нулевого аргумента при воссоздании фрагментов (например, если они восстанавливаются после того, как их убили по собственным причинам Android). По этой причине нецелесообразно полагаться на конструктор с параметрами.

Чтобы гарантировать, что ожидаемые аргументы фрагмента всегда присутствуют, вы можете использовать статический метод `newInstance()` для создания фрагмента и поместить все параметры, которые вы хотите, в пакет, который будет доступен при создании нового экземпляра.

```
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);
    }
  }
}
```

Теперь в Управлении:

```
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();
```

Этот шаблон является наилучшей практикой для обеспечения того, чтобы все необходимые аргументы были переданы фрагментам при создании. Обратите внимание, что когда система уничтожает фрагмент и повторно создает его позже, он автоматически восстанавливает свое состояние - но вы должны предоставить ему реализацию `onSaveInstanceState(Bundle)` .

Навигация между фрагментами с использованием backstack и статической структуры ткани

Прежде всего, нам нужно добавить наш первый `Fragment` в начале, мы должны сделать это в `onCreate()` нашей деятельности:

```
if (null == savedInstanceState) {
    getSupportFragmentManager().beginTransaction()
      .addToBackStack("fragmentA")
      .replace(R.id.container, FragmentA.newInstance(), "fragmentA")
      .commit();
}
```

Затем нам нужно управлять нашей задней стопой. Самый простой способ - использовать функцию, добавленную в нашу деятельность, которая используется для всех 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();
}
```

Наконец, мы должны переопределить `onBackPressed()` чтобы выйти из приложения,

возвращаясь из последнего фрагмента, доступного в 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();
    }
}
```

Исполнение в деятельности:

```
replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Выполнение внешней деятельности (при условии, что `MainActivity` - это наша деятельность):

```
((MainActivity) getActivity()).replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Передача данных из Activity в Fragment с помощью Bundle

Все фрагменты должны иметь пустой конструктор (т. Е. Метод конструктора без входных аргументов). Поэтому, чтобы передать ваши данные в созданный фрагмент, вы должны использовать метод `setArguments()` . Эти методы получают пакет, в который хранятся ваши данные, и хранит Bundle в аргументах. Впоследствии этот Bundle может быть восстановлен в `onCreate()` и `onCreateView()` вызов фрагмента.

Деятельность:

```
 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();
```

Фрагмент:

```
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    String myValue = this.getArguments().getString("message");
    ...
 }
```

Отправка событий обратно в действие с интерфейсом обратного вызова

Если вам нужно отправить события от фрагмента к активности, одним из возможных решений является определение интерфейса обратного вызова и требование его реализации.

пример

Отправлять обратный вызов в действие, когда кнопка фрагмента нажата

Прежде всего, определите интерфейс обратного вызова:

```
public interface SampleCallback {
    void onButtonClicked();
}
```

Следующий шаг - назначить этот обратный вызов в фрагменте:

```
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;
    }
```

```
}
```

И, наконец, реализовать обратный вызов в действии:

```
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
    }
}
```

Анимировать переход между фрагментами

Чтобы оживить переход между фрагментами или оживить процесс показа или скрытия фрагмента, вы используете `FragmentManager` для создания `FragmentTransaction` .

Для одного `FragmentTransaction` существует два разных способа выполнения анимаций: вы можете использовать стандартную анимацию или можете предоставить свои собственные анимации.

Стандартные анимации определяются вызовом `FragmentTransaction.setTransition(int transit)` и использованием одной из предварительно определенных констант, доступных в классе `FragmentTransaction` . На момент написания эти константы:

```
FragmentTransaction.TRANSIT_NONE
FragmentTransaction.TRANSIT_FRAGMENT_OPEN
FragmentTransaction.TRANSIT_FRAGMENT_CLOSE
FragmentTransaction.TRANSIT_FRAGMENT_FADE
```

Полная транзакция может выглядеть примерно так:

```
getSupportFragmentManager()
    .beginTransaction()
    .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

Пользовательские анимации задаются путем вызова либо `FragmentTransaction.setCustomAnimations(int enter, int exit)` либо `FragmentTransaction.setCustomAnimations(int enter, int exit, int popEnter, int popExit)` .

Анимации `enter` и `exit` будут воспроизводиться для `FragmentTransaction` s, которые не включают выпадение фрагментов из заднего стека. `popEnter` и `popExit` будут воспроизводиться при появлении фрагмента из заднего стека.

Следующий код показывает, как вы замените фрагмент, скользя один фрагмент и

скользящий другой в своем месте.

```
getSupportFragmentManager()
    .beginTransaction()
    .setCustomAnimations(R.anim.slide_in_left, R.anim.slide_out_right)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

Определения анимации XML будут использовать тег `objectAnimator` . Пример **файла slide_in_left.xml** может выглядеть примерно так:

```
<?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>
```

Связь между фрагментами

Все сообщения между фрагментами должны проходить через Activity. Фрагменты **не могут** связываться друг с другом без участия.

Дополнительные ресурсы

- Как реализовать OnFragmentInteractionListener
- Android | Общение с другими фрагментами

В этом примере у нас есть `MainActivity` котором размещены два фрагмента, `SenderFragment` и `ReceiverFragment` , для отправки и получения `message` (простая строка в этом случае) соответственно.

Кнопка в `SenderFragment` инициирует процесс отправки сообщения. TextView в `ReceiverFragment` обновляется, когда сообщение получено им.

Ниже приведен фрагмент для MainActivity с комментариями, объясняющими важные строки кода:

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

Файл макета для `MainActivity` содержит два фрагмента внутри 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` предоставляет интерфейс `SendMessageListener` который помогает `MainActivity` знать, когда была нажата кнопка в `SenderFragment` .

Ниже приведен фрагмент кода для `SenderFragment` объясняющий важные строки кода:

```
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;
}
}
```

Файл макета для `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>
```

Функция `ReceiverFragment` проста и предоставляет простой публичный метод для обновления TextView. Когда `MainActivity` получает сообщение от `SenderFragment` он вызывает этот общедоступный метод `ReceiverFragment`

Ниже приведен фрагмент кода для `ReceiverFragment` с комментариями, объясняющими важные строки кода:

```
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);
}
}
```

Файл макета для `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>
```

Прочитайте Фрагменты онлайн: https://riptutorial.com/ru/android/topic/1396/фрагменты

глава 260: фреска

Вступление

Fresco - мощная система для отображения изображений в приложениях Android.

В Android 4.x и ниже Fresco помещает изображения в особый регион **памяти Android** (называемый ashmem). Это позволяет вашему приложению работать быстрее - и страдать от ужасного OutOfMemoryError гораздо реже.

Fresco также поддерживает потоковое воспроизведение JPEG.

замечания

Как настроить зависимости в файле build.gradle на уровне приложения:

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

Более подробную информацию можно найти здесь .

Examples

Начало работы с Fresco

Сначала добавьте Fresco в свой `build.gradle` как показано в разделе «Примечания»:

Если вам нужны дополнительные функции, такие как анимированная поддержка GIF или WebP, вам также необходимо добавить соответствующие артефакты Fresco .

Фреска должна быть инициализирована. Вы должны делать это только 1 раз, поэтому размещение инициализации в вашем `Application` - хорошая идея. Примером этого может быть:

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

Если вы хотите загружать удаленные изображения с сервера, вашему приложению

требуется разрешение интерната. Просто добавьте его в свой `AndroidManifest.xml` :

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

Затем добавьте `SimpleDraweeView` в свой XML-макет. Fresco не поддерживает `wrap_content` для размеров изображения, поскольку у вас может быть несколько изображений с различными размерами (изображение с образцом, изображение ошибки, фактическое изображение, ...).

Таким образом, вы можете добавить `SimpleDraweeView` с фиксированными параметрами (или `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" />
```

Или укажите *пропорции* для вашего изображения:

```
<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" />
```

Наконец, вы можете установить свой URI изображения в Java:

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

Это оно! Вы должны увидеть, что ваш placeholder доступен до тех пор, пока изображение сети не будет извлечено.

Использование OkHttp 3 с Fresco

Во-первых, в дополнение к нормальной зависимости Fresco Gradle вам нужно добавить зависимость OkHttp 3 к вашему `build.gradle` :

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

Когда вы инициализируете Fresco (обычно в вашей пользовательской реализации `Application`), вы можете указать свой клиент 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);
```

Потоковое воспроизведение JPEG с помощью Fresco с помощью DraweeController

В этом примере предполагается, что вы уже добавили Fresco в свое приложение (см. Этот пример):

```
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.
```

Прочитайте фреска онлайн: https://riptutorial.com/ru/android/topic/5217/фреска

глава 261: Хранение файлов во внутреннем и внешнем хранилищах

Синтаксис

- FileOutputStream openFileInput (имя строки)
- FileOutputStream openFileOutput (имя строки, режим int)
- Файл (File dir, String name)
- Файл (строка)
- Файл getExternalStoragePublicDirectory (тип строки)
- Файл getExternalFilesDir (тип строки)

параметры

параметр	подробности
название	Имя файла для открытия. ПРИМЕЧАНИЕ. Нельзя содержать разделители путей
Режим	Рабочий режим. Используйте `MODE_PRIVATE` для работы по умолчанию и `MODE_APPEND` чтобы добавить существующий файл. Другие режимы включают `MODE_WORLD_READABLE` и `MODE_WORLD_WRITEABLE` , которые оба устарели в API 17.
реж	Каталог файла для создания нового файла в
дорожка	Путь для указания местоположения нового файла
тип	Тип каталога файлов для извлечения. Может быть `null` или любым из следующих: `DIRECTORY_MUSIC` , `DIRECTORY_PODCASTS` , `DIRECTORY_RINGTONES` , `DIRECTORY_ALARMS` , `DIRECTORY_NOTIFICATIONS` , `DIRECTORY_PICTURES` или `DIRECTORY_MOVIES`

Examples

Использование внутреннего хранилища

По умолчанию любые файлы, которые вы сохраняете во внутреннем хранилище, являются приватными для вашего приложения. К другим приложениям и пользователям при обычных условиях они не могут быть доступны. **Эти файлы удаляются, когда пользователь удаляет приложение** .

Запись текста в файл

```
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();
}
```

Добавление текста в существующий файл

Используйте `Context.MODE_APPEND` для параметра режима `openFileOutput`

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

Использование внешнего хранилища

«Внешнее» хранилище - это еще один тип хранилища, который мы можем использовать для сохранения файлов на устройстве пользователя. Он имеет некоторые ключевые отличия от «внутреннего» хранилища, а именно:

- Это не всегда доступно. В случае съемного носителя (SD-карта) пользователь может просто удалить хранилище.
- Это не личное. У пользователя (и других приложений) есть доступ к этим файлам.
- Если пользователь удаляет приложение, файлы, которые вы сохраняете в каталоге, полученном с помощью `getExternalFilesDir()` будут удалены.

Чтобы использовать Внешнее хранилище, мы должны сначала получить соответствующие разрешения. Вам нужно будет использовать:

- `android.permission.WRITE_EXTERNAL_STORAGE` для чтения и записи
- `android.permission.READ_EXTERNAL_STORAGE` только для чтения

Чтобы предоставить эти разрешения, вам необходимо будет идентифицировать их в вашем `AndroidManifest.xml` как таковые

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

> ПРИМЕЧАНИЕ. Поскольку они являются опасными разрешениями, если вы используете **API-уровень 23** или выше, вам нужно будет запросить **разрешения во время выполнения** .

Перед попыткой записи или чтения из внешнего хранилища вы всегда должны проверить,

доступен ли носитель данных.

```
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
}
```

При записи файлов во внешнее хранилище вы должны решить, должен ли файл быть общедоступным или приватным. Хотя оба этих типа файлов по-прежнему доступны для пользователя и других приложений на устройстве, существует различие между ними.

Публичные файлы должны оставаться на устройстве, когда пользователь удаляет приложение. Примером файла, который должен быть сохранен как «Публичный», являются фотографии, которые были сделаны через ваше приложение.

Частные файлы должны быть удалены, когда пользователь удалит приложение. Эти типы файлов будут специфичными для приложения и не будут использоваться для пользователей или других приложений. Ex. временные файлы, загруженные / используемые вашим приложением.

Вот как получить доступ к папке « `Documents` как для общедоступных, так и для частных файлов.

общественного

```
// 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();
```

Частный

```
// 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: внутреннее и внешнее хранилище - уточнение терминологии

Разработчики Android (в основном новички) путались относительно терминологии внутреннего и внешнего хранилищ. Есть много вопросов по Stackoverflow относительно того же. Это в основном из-за того, что *терминология в* соответствии с официальной документацией Google / официального Android совсем отличается от *терминологии*

обычного пользователя ОС Android. Поэтому я думал, что документальное подтверждение поможет.

Что мы думаем - терминология пользователя (UT)

Внутреннее хранилище (UT)	Внешнее хранилище (UT)
встроенная внутренняя память телефона	съемная карта Secure Digital (SD) или micro SD
Пример: 32 ГБ внутренней памяти Nexus 6P.	**Пример: место для** хранения на съемных SD-картах, поставляемых такими поставщиками, как samsung, sandisk, strontium, transcend и другие

Но, **согласно Android Documentation / Guide - терминология Google (GT)**

Внутреннее хранилище (GT):

> По умолчанию файлы, сохраненные во внутреннем хранилище, являются приватными для вашего приложения, а другие приложения не могут получить к ним доступ (а также не могут).

Внешнее хранилище (GT):

> Это может быть съемный носитель (например, SD-карта) или внутреннее (несъемное) хранилище.

Внешнее хранилище (GT) можно разделить на два типа:

Первичное внешнее хранилище	Вторичное внешнее хранилище или съемное хранилище (GT)
Это то же самое, что встроенная внутренняя память телефона (или) Внутреннее хранилище (UT)	Это то же самое, что и съемное хранилище micro SD-карт (или) Внешнее хранилище (UT)
Пример: 32 ГБ внутренней памяти Nexus 6P.	**Пример: место для** хранения на съемных SD-картах, поставляемых такими поставщиками, как samsung, sandisk, strontium, transcend и другие
Этот тип хранилища можно получить на ПК с Windows, подключив телефон к ПК через USB-кабель и выбрав « *Камера» (PTP)* в уведомлении о настройках USB.	Этот тип хранилища можно получить на ПК с Windows, подключив телефон к ПК через USB-кабель и выбрав « *Передача файлов»* в уведомлении о параметрах USB.

В двух словах,

Внешнее хранилище (GT) = Внутреннее хранилище (UT) и внешнее хранилище (UT)

Съемное хранилище (GT) = Внешнее хранилище (UT)

Внутреннее хранилище (GT) не имеет термина в UT.

Позвольте мне объяснить,

Внутреннее хранилище (GT): по умолчанию файлы, сохраненные во внутреннем хранилище, являются приватными для вашего приложения, а другие приложения не могут получить к ним доступ. Пользователь вашего приложения также не может получить к ним доступ с помощью диспетчера файлов; даже после включения опции «показать скрытые файлы» в диспетчере файлов. Чтобы получить доступ к файлам во внутреннем хранилище (GT), вам необходимо укрепить свой телефон Android. Более того, когда пользователь удаляет ваше приложение, эти файлы удаляются / удаляются.

Так Internal Storage (GT) является **не** то , что мы думаем , как внутренняя память 32/64 Гб Nexus 6P в

Как правило, **местоположение внутреннего хранилища (GT)** будет:
`/data/data/your.application.package.appname/someDirectory/`

Внешнее хранилище (GT):

> Каждое Android-совместимое устройство поддерживает общую «внешнюю память», которую вы можете использовать для сохранения файлов. Файлы, сохраненные во внешнем хранилище, читаются в мире и могут быть изменены пользователем при включении массовой памяти USB для передачи файлов на компьютер.

Расположение внешнего хранилища (GT): оно может быть в *любом месте* вашего внутреннего хранилища (UT) или на съемном носителе (GT), то есть на карте micro SD. Это зависит от OEM вашего телефона, а также от версии ОС Android.

Чтобы читать или записывать файлы на внешнем хранилище (GT), ваше приложение должно получить системные разрешения `READ_EXTERNAL_STORAGE` или `WRITE_EXTERNAL_STORAGE` .

Например:

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

> Если вам нужно как читать, так и записывать файлы, вам необходимо запросить только разрешение `WRITE_EXTERNAL_STORAGE` , так как оно также требует доступа к

чтению.

В **External Storage (GT)** вы также можете сохранять файлы, **закрытые для приложения**

Но,

> Когда пользователь удаляет ваше приложение, этот каталог и все его содержимое удаляются.

Когда вам нужно сохранять файлы, **закрытые для приложения,** в **External Storage (GT)** ?

> Если вы обрабатываете файлы, которые не предназначены для использования другими приложениями (например, графические текстуры или звуковые эффекты, используемые только вашим приложением), вы должны использовать личный каталог хранилища на внешнем хранилище
>
> Начиная с Android 4.4, чтение или запись файлов в личных каталогах вашего приложения не требует `READ_EXTERNAL_STORAGE` или `WRITE_EXTERNAL_STORAGE` . Таким образом, вы можете заявить, что разрешение должно запрашиваться только в более низких версиях Android, добавив атрибут `maxSdkVersion` :

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

Способы хранения во внутреннем хранилище (GT):

Оба эти метода присутствуют в классе Context

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

File getFilesDir ()
```

Способы хранения в основном внешнем хранилище, т.е. внутреннем хранилище (UT):

```
File getExternalStorageDirectory ()

File getExternalFilesDir (String type)

File getExternalStoragePublicDirectory (String type)
```

В начале все использовали Environment.getExternalStorageDirectory () , который указывал на **корень Первичного внешнего хранилища** . В результате первичное внешнее хранилище было заполнено случайным контентом.

Позже эти два метода были добавлены:

1. В классе `Context` они добавили getExternalFilesDir () , указав на **каталог приложения для** первичного внешнего хранилища. Этот каталог и его содержимое **будут удалены,** когда приложение **будет удалено** .

2. Environment.getExternalStoragePublicDirectory () для централизованных мест для хранения известных типов файлов, таких как фотографии и фильмы. Этот каталог и его содержимое **НЕ удаляются** при **удалении** приложения.

Способы хранения в съемном хранилище (GT), то есть карта micro SD

До **уровня API 19** не было **официального способа** хранения на SD-карте. Но многие могли сделать это с помощью неофициальных библиотек или API.

Официально один метод был введен в классе `Context` в API 19 (Android версии 4.4 - Kitkat).

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

> Он возвращает абсолютные пути к каталогам конкретных приложений на всех устройствах общего доступа / внешних устройств хранения, где приложение может размещать постоянные файлы, которыми он владеет. Эти файлы являются внутренними для приложения и обычно не видны пользователю как носители.

Это означает, что он вернет пути к **обоим** типам внешнего хранилища (GT) - Внутренняя память и карта Micro SD. Обычно **второй путь -** это **путь** хранения микро SD-карты (но не всегда). Поэтому вам нужно проверить это, выполнив код с помощью этого метода.

Пример с фрагментом кода:

Я создал новый проект андроида с пустой активностью, написал следующий код внутри

`protected void onCreate(Bundle savedInstanceState)` метод `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);
```

После выполнения выше кода,

Выход:

```
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
```

Примечание. Я подключил свой телефон к ПК с ОС Windows; включил обе опции разработчика, отладку USB и затем запустил этот код. Если вы **не подключите свой телефон** ; но вместо этого запускайте это на **эмуляторе Android** , ваш выход может отличаться. Моя модель телефона - Coolpad Note 3 - работает на Android 5.1

Место хранения на моем телефоне:

Место хранения микро SD : `/storage/sdcard1`

Внутреннее хранилище (UT) : `/storage/sdcard0` .

Обратите внимание, что `/sdcard` & `/storage/emulated/0` `/sdcard` `/storage/emulated/0` также указывают на внутреннее хранилище (UT). Но это символические `/storage/sdcard0` на `/storage/sdcard0` .

Чтобы четко понимать разные пути хранения в Android, пройдите этот ответ

Отказ от ответственности: все упомянутые выше пути хранения - это пути на **моем** телефоне. Ваши файлы могут **не** храниться на одинаковых путях хранения. Поскольку места хранения / пути хранения могут отличаться на других мобильных телефонах в зависимости от вашего поставщика, производителя и различных версий ОС Android.

Сохранить базу данных на SD-карте (резервная БД на 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;
    }
}
```

вызовите этот метод следующим образом:

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

Выбор каталога устройств:

Сначала добавьте разрешение на хранение для чтения / извлечения каталога устройства.

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

Создать класс модели

```
//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;
    }
}
```

Создайте список, используя модель каталога, чтобы добавить данные каталога.

```
//define list to show directory

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

Fetch, используя следующий метод.

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

Печатать список каталогов в журнале.

```
//print directory list in logs

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

использование

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

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

Для извлечения внутренних файлов / папки определенного каталога используйте тот же самый метод, просто измените аргумент, передайте текущий выбранный путь в аргументе и ответьте дескриптор для этого.

Чтобы получить расширение файла:

```
private String getExtension(String filename) {

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

    return extension;
}
```

Прочитайте Хранение файлов во внутреннем и внешнем хранилищах онлайн: https://riptutorial.com/ru/android/topic/150/хранение-файлов-во-внутреннем-и-внешнем-хранилищах

глава 262: Цвета

Examples

Манипуляция цветом

Чтобы манипулировать цветами, мы будем изменять значения цвета argb (Alpha, Red, Green и Blue).

Сначала извлеките значения RGB из вашего цвета.

```
int yourColor = Color.parse("#ae1f67");

int red = Color.red(yourColor);
int green = Color.green(yourColor);
int blue = Color.blue(yourColor);
```

Теперь вы можете уменьшить или увеличить красные, зеленые и синие значения и снова объединить их в цвет:

```
 int newColor = Color.rgb(red, green, blue);
```

Или, если вы хотите добавить к нему некоторую альфу, вы можете добавить ее при создании цвета:

```
 int newColor = Color.argb(alpha, red, green, blue);
```

Значения альфа и RGB должны находиться в диапазоне [0-225].

Прочитайте Цвета онлайн: https://riptutorial.com/ru/android/topic/4986/цвета

глава 263: Чтение штрих-кода и QR-кода

замечания

QRCodeReaderView

ZXing

Examples

Использование QRCodeReaderView (на основе Zxing)

QRCodeReaderView реализует представление Android, которое показывает камеру и уведомляет, когда в предварительном просмотре есть код QR.

В нем используется zxing c открытым исходным кодом, многоформатная библиотека обработки изображений штрих-кода 1D / 2D.

Добавление библиотеки в ваш проект

Добавьте зависимость QRCodeReaderView к вашему build.gradle

```
dependencies{
    compile 'com.dlazaro66.qrcodereaderview:qrcodereaderview:2.0.0'
}
```

Первое использование

- Добавьте в свой макет `QRCodeReaderView`

```
<com.dlazaro66.qrcodereaderview.QRCodeReaderView
       android:id="@+id/qrdecoderview"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />
```

- Создайте действие, которое реализует `onQRCodeReadListener` , и используйте его как слушатель `QrCodeReaderView` .
- Убедитесь, что у вас есть разрешения камеры, чтобы использовать библиотеку. (https://developer.android.com/training/permissions/requesting.html)

Затем в своей деятельности вы можете использовать его следующим образом:

```
    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();
    }
}
```

Прочитайте Чтение штрих-кода и QR-кода онлайн: https://riptutorial.com/ru/android/topic/6067/чтение-штрих-кода-и-qr-кода

глава 264: Что такое ProGuard? Что используется в Android?

Вступление

Proguard - это бесплатный инструмент для сжатия файлов Java, оптимизатор, обфускатор и преверизатор. Он обнаруживает и удаляет неиспользуемые классы, поля, методы и атрибуты. Он оптимизирует байт-код и удаляет неиспользуемые инструкции. Он переименовывает остальные классы, поля и методы, используя короткие бессмысленные имена.

Examples

Сократите свой код и ресурсы с помощью proguard

Чтобы файл APK был как можно меньше, вы должны включить сокращение, чтобы удалить неиспользуемый код и ресурсы в вашей версии сборки. На этой странице описано, как это сделать и как указывать, какой код и ресурсы сохранить или удалить во время сборки.

Сокращение кода доступно с помощью ProGuard, который обнаруживает и удаляет неиспользуемые классы, поля, методы и атрибуты из вашего упакованного приложения, в том числе из включенных библиотек кода (что делает его ценным инструментом для работы с лимитом ссылки 64k). ProGuard также оптимизирует байт-код, удаляет неиспользуемые инструкции кода и обфускает оставшиеся классы, поля и методы с короткими именами. Обфусканный код делает ваш АПК сложным для обратного проектирования, что особенно ценно, когда ваше приложение использует чувствительные к безопасности функции, такие как проверка лицензии.

Сокращение ресурсов доступно с плагином Android для Gradle, который удаляет неиспользуемые ресурсы из вашего упакованного приложения, включая неиспользуемые ресурсы в библиотеках кода. Он работает в сочетании с сокращением кода, так что после удаления неиспользуемого кода любые удаленные ссылки могут быть также безопасно удалены.

Сжатие кода

Чтобы включить сокращение кода с помощью `ProGuard` , добавьте `minifyEnabled` true в соответствующий тип сборки в файле `build.gradle` .

Имейте в виду, что сокращение кода замедляет время сборки, поэтому вам следует избегать использования его при сборке отладки, если это возможно. Тем не менее, важно,

чтобы вы включили сокращение кода в своем конечном APK, используемом для тестирования, потому что это может привести к ошибкам, если вы недостаточно настроили какой код сохранить.

Например, следующий фрагмент файла `build.gradle` позволяет `build.gradle` для сборки релиза:

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

В дополнение к `minifyEnabled` собственности, `proguardFiles` свойство определяет `ProGuard rules` :

Метод getDefaultProguardFile ('proguard-android.txt') получает настройки ProGuard по умолчанию из `tools/proguard/ folder` Android SDK `tools/proguard/ folder` . Совет. Для еще большего сокращения кода попробуйте `proguard-android-optimize.txt` , расположенный в том же месте. Он включает в себя те же правила ProGuard, но с другими оптимизациями, которые выполняют анализ на уровне байт-кода внутри и между методами - чтобы уменьшить размер вашего APK и помочь ему работать быстрее. Файл `proguard-rules.pro` - это то, где вы можете добавить пользовательские правила ProGuard. По умолчанию этот файл находится в корне модуля (рядом с файлом build.gradle). Чтобы добавить дополнительные правила ProGuard, специфичные для каждого варианта сборки, добавьте другое свойство proguardFiles в соответствующий блок `productFlavor` . Например, следующий файл Gradle добавляет flavor2-rules.pro к вкусу аромата 2. Теперь flavor2 использует все три правила ProGuard, поскольку также применяются те из блока release.

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

Прочитайте Что такое ProGuard? Что используется в Android? онлайн:
https://riptutorial.com/ru/android/topic/9205/что-такое-proguard--что-используется-в-android-

глава 265: Шаблоны проектирования

Вступление

Шаблоны проектирования - это формализованные передовые методы, которые программист может использовать для решения общих проблем при разработке приложения или системы.

Шаблоны проектирования могут ускорить процесс разработки, предоставив проверенные, проверенные парадигмы развития.

Повторное использование шаблонов проектирования помогает предотвратить тонкие проблемы, которые могут вызвать серьезные проблемы, а также улучшает читаемость кода для кодеров и архитекторов, знакомых с шаблонами.

Examples

Пример Singleton Class

Шаблон Java Singleton

Для реализации шаблона Singleton у нас есть разные подходы, но все они имеют общие понятия.

- Частный конструктор для ограничения экземпляра класса из других классов.
- Частная статическая переменная того же класса, которая является единственным экземпляром класса.
- Открытый статический метод, возвращающий экземпляр класса, это глобальный доступ
- point для внешнего мира, чтобы получить экземпляр класса 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;
  }
}
```

Схема наблюдателя

Шаблон наблюдателя является общей схемой, которая широко используется во многих контекстах. Реальный пример можно взять с YouTube: когда вам нравится канал и вы хотите получать все новости и смотреть новые видео с этого канала, вам нужно подписаться на этот канал. Затем, всякий раз, когда этот канал публикует новости, вы (и все остальные подписчики) получите уведомление.

Наблюдатель будет иметь два компонента. Один из них - вещатель (канал), а другой - приемник (вы или любой другой абонент). Передатчик будет обрабатывать все экземпляры получателя, которые подписались на него. Когда вещатель запускает новое событие, он объявит об этом всем экземплярам приемника. Когда получатель получает событие, он должен будет реагировать на это событие, например, включив YouTube и воспроизведя новое видео.

Внедрение шаблона наблюдателя

1. Передатчик должен предоставить методы, позволяющие получателям подписываться и отписаться от него. Когда вещатель запускает событие, абоненты должны быть уведомлены о том, что произошло событие:

```
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. Получателю необходимо реализовать метод, который обрабатывает событие от вещателя:

```
interface Subscriber {
    void doSubscribe(Channel channel);
```

```
    void doUnsubscribe(Channel channel);
    void handleEvent();  // Process the new event.
}
```

Прочитайте Шаблоны проектирования онлайн: https://riptutorial.com/ru/android/topic/9949/шаблоны-проектирования

глава 266: Шифрование / дешифрование данных

Вступление

В этом разделе обсуждается, как работает шифрование и дешифрование в Android.

Examples

AES шифрование данных с использованием пароля безопасным способом

В следующем примере шифруется данный блок данных с использованием AES . Ключ шифрования выведен безопасным способом (случайная соль, 1000 раундов SHA-256). Шифрование использует AES в режиме CBC со случайным IV .

Обратите внимание, что данные, хранящиеся в классе `EncryptedData` (`salt` , `iv` и `encryptedData`), могут быть объединены в один массив байтов. Затем вы можете сохранить данные или передать их получателю.

```
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;
}
```

Следующий пример кода показывает, как тестировать шифрование и дешифрование:

```
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();
}
```

Прочитайте Шифрование / дешифрование данных онлайн:
https://riptutorial.com/ru/android/topic/3471/шифрование---дешифрование-данных

глава 267: Эмулятор

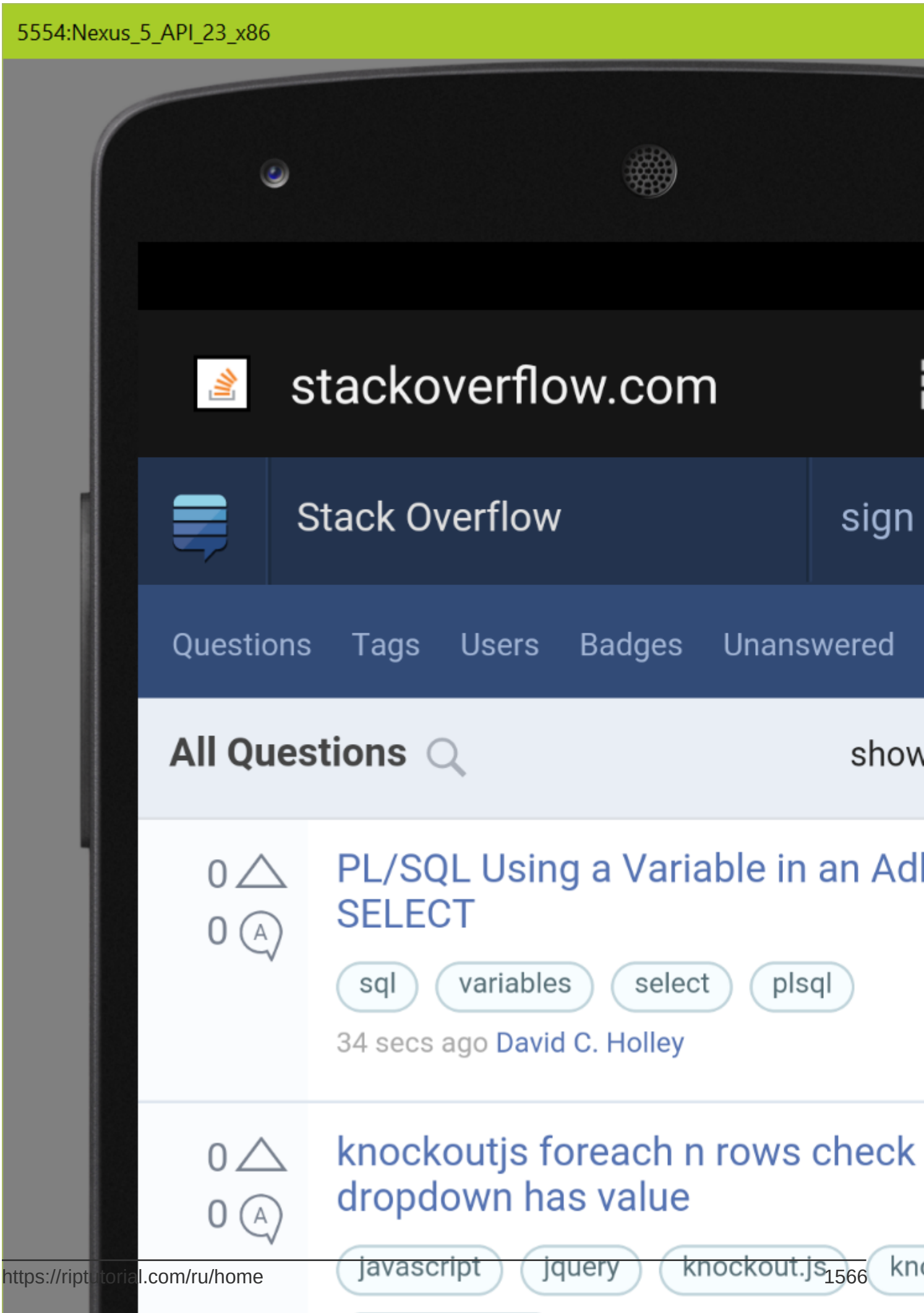
замечания

AVD означает *Android Virtual Device*

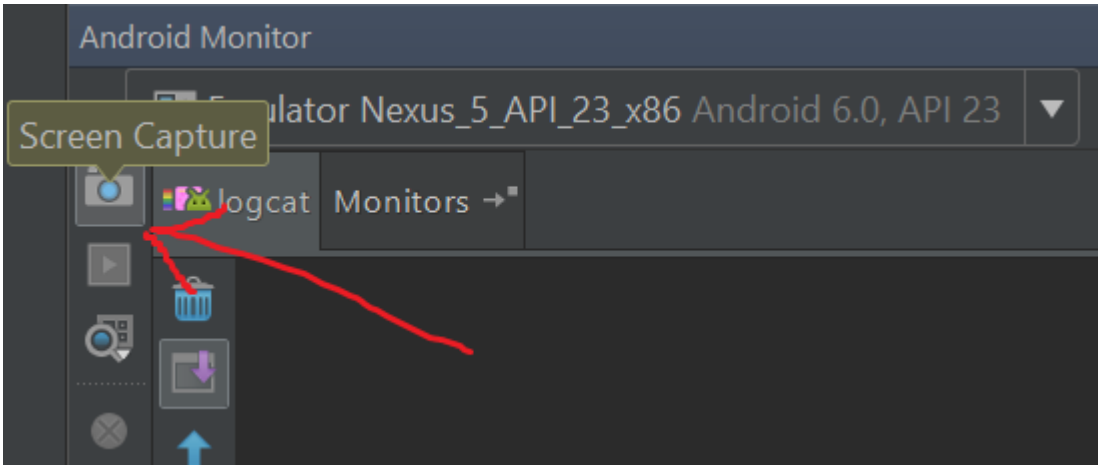
Examples

Создание скриншотов

Если вы хотите снять скриншот с Android Emulator (2.0), вам просто нужно нажать `Ctrl` + `S` или щелкнуть значок камеры на боковой панели:


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

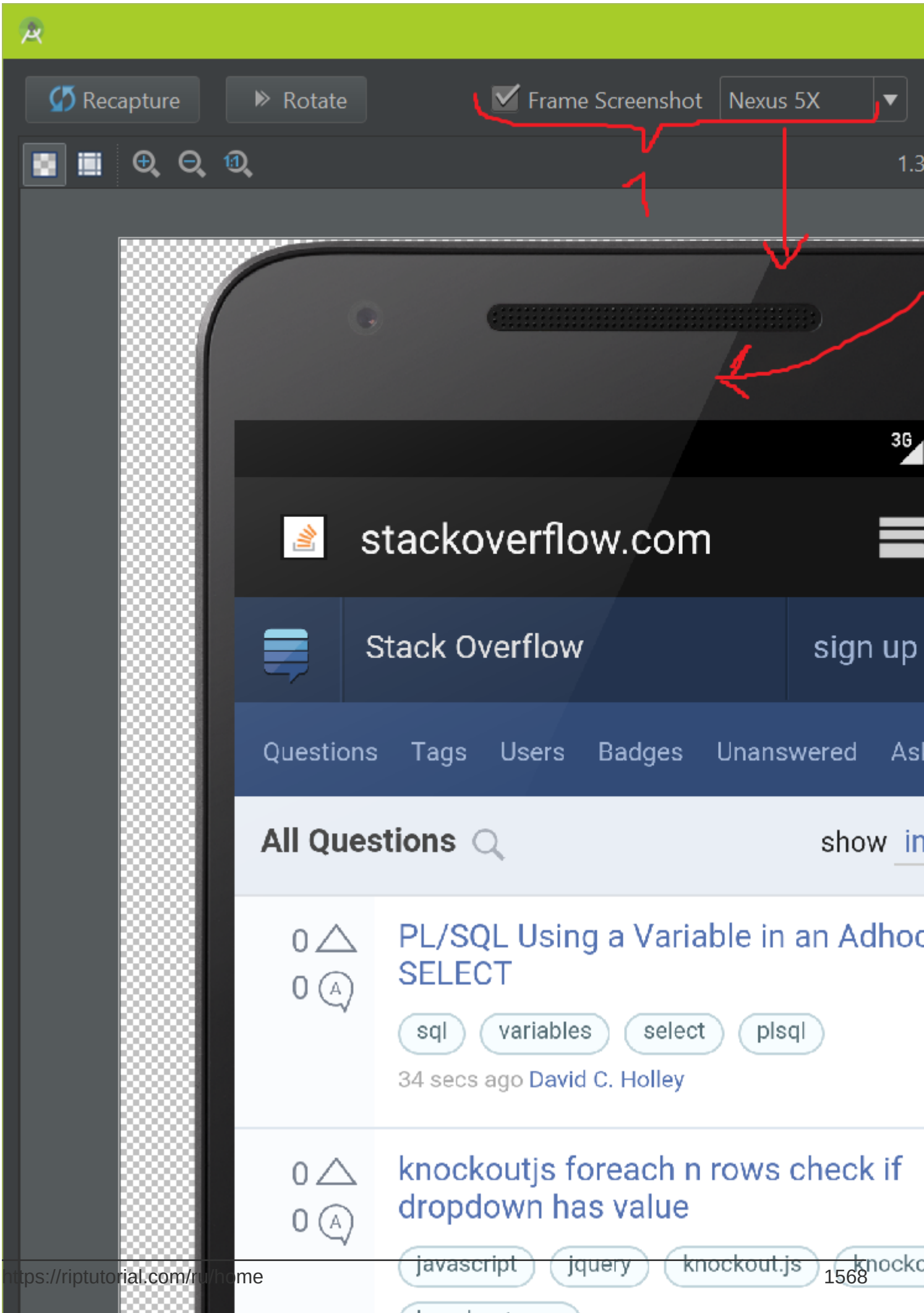
или хотите сделать снимок экрана с реального устройства, вам нужно щелкнуть по значку камеры в Android Monitor:



Дважды проверьте, что вы выбрали правильное устройство, потому что это обычная ошибка.

После снятия скриншота вы можете дополнительно добавить к нему следующие украшения (также см. Изображение ниже):

1. Рамка устройства вокруг снимка экрана.
2. Тень ниже рамки устройства.
3. Экранная блики на раме устройства и снимок экрана.

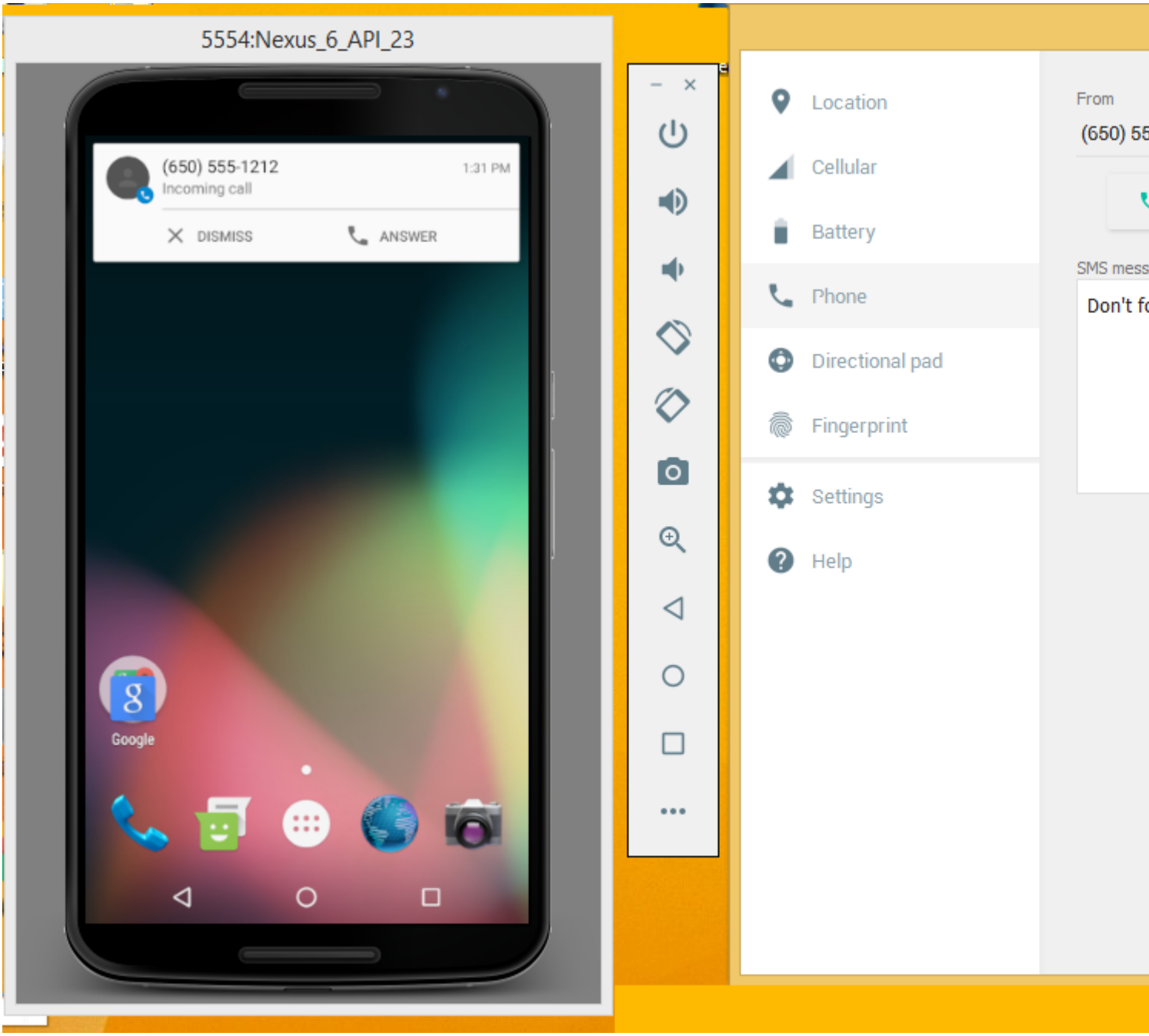

Recapture
Rotate
Frame Screenshot
Nexus 5X
3G
stackoverflow.com
Stack Overflow
sign up
Questions
Tags
Users
Badges
Unanswered
All Questions
show
0
0
PL/SQL Using a Variable in an Adhoc SELECT
sql
variables
select
plsql
34 secs ago David C. Holley
0
0
knockoutjs foreach n rows check if dropdown has value
javascript
jquery
knockout.js

из командной строки, используя `android avd` .

Вы также можете получить доступ к AVD Manager из студии Android, используя `Tools` > `Android` > `AVD Manager` или щелкнув значок AVD Manager на панели инструментов, который является вторым на скриншоте ниже.

Имитировать вызов

Чтобы смоделировать телефонный звонок, нажмите кнопку «Расширенные элементы управления», обозначенную тремя точками, выберите «Телефон» и выберите «Вызов». Вы также можете изменить номер телефона.



Разрешение ошибок при запуске эмулятора

Прежде всего, убедитесь, что вы включили « **виртуализацию**» в настройках BIOS.

Запустите **Android SDK Manager** , выберите « **Дополнительно»,** а затем выберите « **Аппаратное ускоренное выполнение диспетчера приложений Intel»** и дождитесь завершения загрузки. Если он все еще не работает, откройте папку SDK и запустите `/extras/intel/Hardware_Accelerated_Execution_Manager/IntelHAXM.exe` .

Для завершения установки следуйте инструкциям на экране.

Или для OS X вы можете сделать это без экранных подсказок, например: `/extras/intel/Hardware_Accelerated_Execution_Manager/HAXM\ installation`

> Если ваш процессор не поддерживает VT-x или SVM, вы не можете использовать образы Android на основе x86. Вместо этого используйте изображения на основе ARM.

После завершения установки убедитесь, что драйвер виртуализации работает правильно, открыв окно командной строки и выполнив следующую команду: `sc query intelhaxm`

Для запуска эмулятора x86 с ускорением VM: если вы запускаете эмулятор из командной строки, просто укажите x86-based AVD: `emulator -avd <avd_name>`

Если вы выполните все описанные выше шаги правильно, вы наверняка сможете увидеть, что ваш AVD с HAXM подходит нормально.

Прочитайте Эмулятор онлайн: https://riptutorial.com/ru/android/topic/122/эмулятор

глава 268: Эффективная загрузка растровых изображений

Вступление

Эта тема в основном сосредоточена на эффективной загрузке растровых изображений на устройствах Android.

Когда дело доходит до загрузки растрового изображения, вопрос возникает там, откуда он загружается. Здесь мы поговорим о том, как загрузить Bitmap из Resource с помощью устройства Android. например, из галереи.

Мы рассмотрим это на примере, который обсуждается ниже.

Синтаксис

- `<uses-permission>` **->** Тег, используемый для разрешения.
- `android:name` **->** Атрибут, используемый для указания имени для разрешения, которое мы будем запрашивать.
- `android.permission.READ_EXTERNAL_STORAGE` **->** Это системные разрешения
- пример "android.permission.CAMERA" или "android.permission.READ_CONTACTS"

Examples

Загрузите изображение из ресурса с устройства Android. Использование намерений.

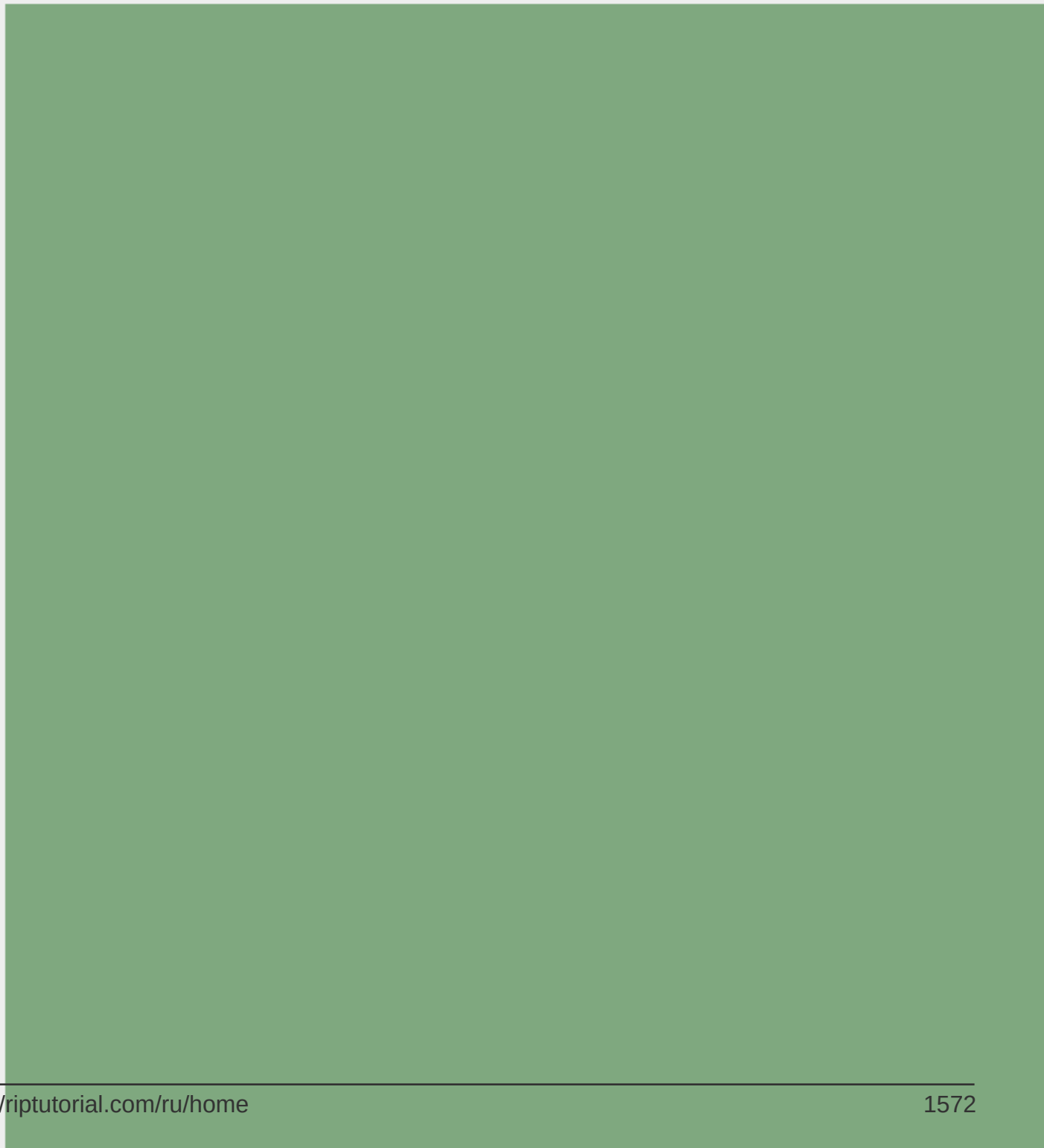
Использование намерений для загрузки изображения из галереи.

1. Первоначально вам нужно иметь разрешение

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

2. Используйте следующий код, чтобы следующий макет был разработан.

LoadImageFrmGallery



https://riptutorial.com/ru/android/topic/10902/эффективная-загрузка-растровых-изображений

кредиты

S. No	Главы	Contributors
1	Начало работы с 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, Charuක, 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, רתוא בהוא ושי
2	9-патч-изображения	Knossos, Nissim R, Tomik
3	ACRA	Zarul Izham, Zoe
4	ADB (Android Debug Bridge)	3VYZkz7t, adao7000, Ahmad Aghazadeh, Amit Thakkar, AndroidMechanic, Anirudh Sharma, Anup Kulkarni, auval, Barend, Blackbelt, Burak Day, Charuක, Chris Stratton, Da-Jin C, Dale, Daniel Nugent, David Cheung, Erik, Fabio, fyfyone Google, g4s8, Gabriele Mariotti, grebulon, Hannoun Yassir, Hi I'm Frogatto, hichris123, honk, jim, Kashyap Jha, Laurel, MCeley, Menasheh, Natali, Nemus, Pavel Durov, Piyush, R. Zagórski, RishbhSharma, stkent, Sudip Bhandari, sukumar, theFunkyEngineer, thiagolr, Tien , Xaver Kapeller, Yassie, younes zeboudj, Yury Fedorov
5	adb shell	3VYZkz7t, Ahmad Aghazadeh, auval, Burak Day, Fabio, fyfyone Google, Hannoun Yassir, Natali, Pavel Durov, R. Zagórski, sukumar, Yury Fedorov

6	AdMob	Carlos Borau, honk, RamenChef, Sukrit Kumar, Zarul Izham, Zoe
7	AIDL	Krishnakanth
8	AlarmManager	Daniel Nugent, devnull69, Greg T, honk, TR4Android
9	Android Authenticator	4444, kRiZ
10	Android NDK	Alex, astuter, Doron Yakovlev-Golani, Flayn, Onik, samgak , still_learning, Täg, thiagolr
11	Android Studio	AndroidMechanic, auval, Blackbelt, Charuක, Daniel Nugent, Gabriele Mariotti, Hiren Patel, Inzimam Tariq IT, Jon Adams, N J, Phan Van Linh, R. Zagórski, Squidward, Sujith Niraikulathan, ThomasThiebaud
12	Android Vk Sdk	alexey polusov
13	Android Вещи	Fabio, honk
14	Android-x86 в VirtualBox	Daniel Nugent, Enrique de Miguel
15	API Android Places	busradeniz, honk, Karan Razdan, Murali
16	API Google Диска	Christlin Joseph, honk
17	API Twitter	Mahmoud Ibrahim
18	API камеры 2	ChemicalFlash, devnull69, RamenChef, webo80
19	API отпечатков пальцев в android	Doron Yakovlev-Golani, RamenChef, user01232
20	API поддержки Google	Dus, honk, Willie Chalmers III
21	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

22	AudioManager	honk, Nicolai Weitkemper
23	AutoCompleteTextView	Harish Gyanani, Jon Adams, Ricardo Vieira, Vivek Mishra
24	Bluetooth и Bluetooth LE API	antonio, Jon Adams, Lukas, Myon, Pavel Durov, R. Zagórski, Reaz Murshed, V-PTR, WMios
25	BottomNavigationView	Abdul Wasae, Daniel Nugent, Gabriele Mariotti, guik, Pankaj Kumar, Pratik Butani, Priyank Patel, RamenChef, rciovati, Stephen Leppik, sud007
26	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
27	CardView	Carlos, Dan, Er. Kaushik Kajavadara, Gabriele Mariotti, Kaushik, Nougat Lover, RamenChef, S.R, Sneh Pandya, Somesh Kumar, Stephen Leppik, sud007, WarrenFaith, Yury Fedorov
28	CleverTap	Jordan, judepereira
29	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
30	ConstraintSet	Pratik Butani
31	ExoPlayer	Hamed Gh
32	Facebook SDK для Android	Akeshwar Jha, AndiGeeky, Community, Daniel Nugent, honk, Zarul Izham
33	Fastjson	KeLiuyue
34	Fastlane	Gokhan Arik
35	FileIO с Android	h22, sun-solar-arrow
36	FileProvider	Joost Verbraeken, pedros
37	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
38	FloatingActionButton	Ahmad Aghazadeh, Charuක, Daniel Nugent, Gabriele

		Mariotti, mattfred, RamenChef, Shinil M S, Stephen Leppik
39	Genymotion для android	Atef Hares, Harish Gyanani
40	Google Maps API v2 для 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
41	Google Play магазин	dakshbhatt21, Daniel Nugent, reVerse
42	Gradle для 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, Charuක, 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
43	GreenBot EventBus	CaseyB, Daniel Nugent, Hamed Momeni, RamenChef
44	GreenDAO	Allan Pereira, Carl Poole, Grundy, MiguelHincapieC, R. Zagórski, RamenChef, Stephen Leppik
45	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
46	HttpURLConnection	Aleks G, Daniel Nugent, Duan Bressan, honk, KDeogharkar, marshmallow, Shantanu Paul, Simone Carletti
47	ImageView	Ahmad Aghazadeh, Ali Sherafat, Chip, Daniel Nugent, DanielDiSu, Dinesh, Gabriele Mariotti, Harish Gyanani, kit, lax1089, Pratik Butani, Squidward, Sup
48	IntentService	Anax, Daniel Nugent, honk, JonasCz, TRINADH KOYA, Yashaswi Maharshi
49	Java на Android	Eugen Martynov

50	JCodec	Adhikari Bishwash
51	JSON в Android с 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
52	Leakcanary	Rakshit Nawani, tynn
53	Lint Warnings	ben75, Daniel Nugent, Gabriele Mariotti, GensaGames, R. Zagórski, rekire, SuperBiasedMan
54	Looper	tynn
55	LruCache	Daniel Nugent, honk, LordSidious, RamenChef, Stephen Leppik
56	MediaSession	Disk Crasher, honk, KuroObi, RamenChef
57	MediaStore	Daniel Nugent, honk, RamenChef, Uttam Panchasara
58	Moshi	Blundell
59	MVVM (Архитектура)	Daniel W., RamenChef, Stephen Leppik
60	NavigationView	Adam Lear, akshay, Charuᔓ, Daniel Nugent, Gabriele Mariotti, Kedar Tendolkar, petrumo, RamenChef, rekire, SANAT, Sevle, Stephen Leppik, sud007
61	Notification Channel Android O	Lokesh Desai
62	OkHttp	A-Droid Tech, Daniel Nugent, Gabriele Mariotti, Gubbel, noob, Rohit Arya, Vucko, Zarul Izham
63	ORMLite в android	Manos
64	PackageManager	FredMaggiowski, Hi I'm Frogatto, Muthukrishnan Rajendran, Piyush, Squidward
65	Parcelable	Alex Sullivan, Andrei T, HoseinIT, Nick Cardoso
66	Ping ICMP	Carl Poole

67	ProGuard - Обфускация и сокращение вашего кода	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
68	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
69	RecyclerView onClickListeners	abhishesh, Braj Bhushan Singh, Bryan, FromTheSeventhSky, fuwaneko, Gabriele Mariotti, honk, RamenChef, Smit.Satodia, Stephen Leppik
70	RecyclerView и LayoutManagers	4444, BalaramNayak, Felix Edelmann, Gabriele Mariotti, Kayvan N, MidasLefko, RamenChef, Stephen Leppik
71	Renderscript	Ankit Popli, Dalija Prasnikar, Froyo, honk, Rucha Bhatt, Xaver Kapeller
72	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
73	Retrofit2 с RxJava	Anand Singh, gaara87, GurpreetSK95, Lukas, mrtuovinen, R. Zagórski, Zarul Izham
74	RoboGuice	AndroidRuntimeException, Lewis McGeary, Rajesh
75	Robolectric	Blundell, g4s8
76	SearchView	Daniel Nugent, Dmide, Hiren Patel, RamenChef, Stephen Leppik, sud007

77	SensorManager	honk, Simon, TDG
78	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
79	ShortcutManager	g4s8, Sukrit Kumar
80	SpannableString	S.R
81	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
82	SyncAdapter с периодической синхронизацией данных	Bhargavi Yamanuri
83	TabLayout	Daniel Nugent, Willie Chalmers III
84	TensorFlow	Pratik Butani
85	TextInputLayout	Adarsh Ashok, BrickTop, Gabriele Mariotti, Hi I'm Frogatto, RamenChef, Shashanth, Sneh Pandya, Stephen Leppik
86	TextView	Beena, Daniel Nugent, Eyad Mhanna, gaara87, Gabriele Mariotti, Hiren Patel, honk, keno, Michele, Sohail Zahid, Sujith Niraikulathan, sun-solar-arrow
87	TransitionDrawable	S.R, Yogesh Umesh Vaity
88	Typedef Аннотации: @IntDef, @StringDef	Gabriele Mariotti, hardik m, mmBs, Pongpat
89	VectorDrawable и	Ahmad Aghazadeh, Aleksandar Stefanović, gaara87, honk,

	AnimatedVectorDrawable	Lewis McGeary, RamenChef, Stephen Leppik
90	VideoView	iravul, Sashabrava
91	ViewFlipper	Anita Kunjir, Daniel Nugent, honk
92	ViewPager	Adarsh Ashok, Adrián Pérez, Daniel Nugent, Gabriele Mariotti, Moustachauve, RamenChef, RediOne1, Rucha Bhatt, Sneh Pandya, Stephen Leppik, Usman, ZeroOne
93	WebView	Amod Gokhale, Daniel Nugent, g4s8, j2ko, jasonlam604, JonasCz, Mohammad Yahia, ppeterka, Prakash Bala, shtolik, Squidward, Sukrit Kumar, sukumar
94	XMPP зарегистрировать логин и чат простой пример	4444, RamenChef, Saveen
95	Xposed	Medusalix
96	Youtube-API	abhishesh, Giannis, honk, MashukKhan, Zarul Izham, Zeeshan Shabbir
97	Zip-файл в android	Adnan
98	Автобус Отто	gus27, tynn
99	Автообновление TextViews	honk, Priyank Patel
100	Аниматоры	Aryan, Bartek Lipinski, Blundering Philosopher, Brenden Kromhout, Caruඞ, Daniel Nugent, Eixx, Hiren Patel, Lewis McGeary, Piyush, TR4Android, Uriel Carrillo, Yury Fedorov
101	Анимированная панель предупреждения AlertDialog	krunal patel, Thomas Easo
102	Аппаратная кнопка События / намерения (PTT, LWP и т. Д.)	JensV
103	Архитектура MVP	Atif Farrukh, Harish Gyanani, honk, Jon Adams, Magesh Pandian, N J, zmingchun
104	Атрибуты инструментов	Dalija Prasnikar, Gabriele Mariotti, Harsh Sharma, Kayvan N, TR4Android

105	База данных Firebase Realtime	Aawaz Gyawali, drulabs, Gabriele Mariotti, honk, Md. Ali Hossain, RamenChef, Sneh Pandya, Stephen Leppik, yennsarah, Zarul Izham
106	Безопасность	xDragonZ
107	Безопасные общие ресурсы	Christlin Joseph
108	Библиотека привязки данных	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
109	Биллинг в приложении	Hussein El Feky, Pro Mode, Zoe
110	Быстрый способ настройки Retrolambda в проекте Android.	anatoli, Md. Ali Hossain
111	вводимый коэффициент	alanv, B001, Daniel Nugent, Greg T, Hiren Patel, Jinesh Francis, Nick Cardoso, TR4Android
112	Векторные иллюстрации	Priyank Patel, ShahiM
113	Версии Android	4444, AndroidMechanic, athor, BooleanCheese, Dalija Prasnikar, Daniel Nugent, Fildor, Gabriele Mariotti, H. Pauwelyn, Matt, RediOne1, tynn
114	Версия проекта SDK	Arnav M., Ranveer, Tanis.7x
115	вибрация	cdeange, Zerntrino
116	Виджеты	4444, Alex Ershov, Daniel Nugent, Don Chakkappan, Imdad, nenofite, sun-solar-arrow
117	волчок	AndroidMechanic, Anonsage, Daniel Nugent, Vishwesh Jainkuniya
118	Время Утилиты	Burhanuddin Rashid, Mukesh Kumar Swami, Muthukrishnan Rajendran
119	Выбор даты и времени	adalPaRi, Brenden Kromhout, Daniel Nugent, Harish Gyanani, Ironman, Milad Nouri, RediOne1, Rohan Arora

120	Декорации RecyclerView	Barend, David Medenjak, Gabriele Mariotti, Muthukrishnan Rajendran, Peter Gordon, RamenChef, Stephen Leppik, Yasin Kaçmaz
121	Деятельность	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
122	Джексон	KeLiuyue
123	диалог	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
124	Дизайн материалов	Akash Patel, Aleksandar Stefanović, Alex Chengalan, AndroidMechanic, Anirudh Sharma, ankit dassor, Bartek Lipinski, Bulwinkel, cascal, Charuක, dakshbhatt21, Dan Hulme, Daniel Nugent, dev.mi, Eixx, fyfyone Google, Gabriele Mariotti, Gal Yedidovich, Guillermo García, honk, Ibrahim, Ichigo Kurosaki, Ishita Sinha, Jaiprakash Soni, jlynch630, Jon Adams, Lewis McGeary, Lucas Paolillo, Machado, mahmoud moustafa, Marina K., MathaN, Max, Menasheh, mmBs, mpkuth, N J, Nikita Kurtin, noongiya95, oshurmamadov, pavel163, Piyush, Pravin Sonawane, Rajesh, RamenChef, rciovati, Reaz Murshed, RediOne1, ridsatrio, Sagar Chavada, Sanoop, sat, Saveen, Shashanth , Simo, SimplyProgrammer, Sneh Pandya, Stephen Leppik, sud007, sudo, sukumar, Uttam Panchasara, Vasily Kabunov, vguzzi, Vivek Mishra, Willie Chalmers III, X3Btel, Xaver Kapeller, Yasin Kaçmaz, Yury Fedorov
125	Добавление FuseView в проект Android	Tudor Luca
126	Доступ к базам данных SQLite с использованием класса ContentValues	Adil Saiyad, emecas, honk
127	Жизненный цикл пользовательского интерфейса	Daniel Nugent, Dinesh Choudhary, Floern, Lewis McGeary, orelzion, R. Zagórski, Sergey Glotov
128	Закусочная	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
129	залп	2943, Ankur Aggarwal, Endzeit, Harsh Dalwadi, herrmartell , honk, Jon Adams, Pablo Baxter, RamenChef, Rubin Nellikunnathu, Rucha Bhatt, sameera lakshitha, Stephen Leppik, VISHWANATH N P
130	Захват скриншотов	Ayush Bansal, Daniel Nugent, honk, Onik, sushant kumar, W0rmH0le
131	Звук и мультимедиа Android	johnrao07, Muhammad Umair Shafique, Squidward
132	Звуковая дорожка	Ayush Bansal
133	Изменения ориентации	EmmanuelMess, k3b, Ricardo Vieira, y.feizi
134	Измерение размеров	mnoronha, stkent
135	Индикатор	Gabriele Mariotti, Hiren Patel, mpkuth, Sanoop, shtolik
136	Инструменты отчетов о сбоях	Ajit Singh, Charuᔕ, Ekin, Gabriele Mariotti, Ishita Sinha, Jason Bourne, Madhukar Hebbar, pRaNaY
137	Интеграция OpenCV в Android Studio	MashukKhan, RamenChef, ssimm
138	Интеграция входа в Google	AndiGeeky, RamenChef, Tot Zam
139	Интеграция с подписью Google на Android	jagapathi
140	Интеграция шлюза Android Paypal	A-Droid Tech
141	Интеллектуальная карточка	shadygoneinsane
142	Интерактивный пользовательский интерфейс с UIAutomator	Timo Bähr

143	Интернационализация и локализация (I18N и L10N)	Ankur Aggarwal
144	Интерфейс Java Java Native (JNI)	Doron Yakovlev-Golani, Muthukrishnan Rajendran, samgak
145	Интерфейсы	appersiano, Brenden Kromhout, Daniel Nugent, RediOne1
146	Инфраструктура приложений Firebase	shalini, tynn
147	Исключения	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
148	Как безопасно хранить пароли	honk, Jaggs
149	Как использовать SparseArray	honk, Robert Banyai
150	Камера и галерея	Ahmad Aghazadeh, carvaq, Daniel Nugent, Hiren Patel, johnrao07, RediOne1, Squidward, Yasin Kaçmaz
151	Картина холста с использованием SurfaceView	davidgiga1993
152	Кинжал 2	Aurasphere, Cabezas, David Medenjak, EpicPandaForce, honk, mattfred, Tomik
153	Кинжал библиотеки 2: Инъекция зависимостей в приложениях	Er. Kaushik Kajavadara, honk
154	клавиатура	Hiren Patel, Kayvan N
155	кнопка	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
156	Компоненты	DeKaNszn

	архитектуры Android	
157	контекст	Will Evers
158	КоординаторLayout и поведение	Adarsh Ashok, Gabriele Mariotti, honk, RamenChef, Stephen Leppik
159	Коснитесь событий	Yvette Colomb
160	Кэш Bitmap	Lokesh Desai
161	Локализация ресурсов на Android	AndroidMechanic, electroid, Fabio, Gubbel, Harish Gyanani, honk, Jinesh Francis, mpkuth, RamenChef, USKMobility
162	Локализованная дата / время в Android	Geert, honk, mnoronha
163	Макеты	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
164	Мгновенный запуск в Android Studio	AndroidMechanic, Daniel Nugent, ridsatrio, Zoe
165	Медиа-плейер	Ahmad Aghazadeh, Carlos Vázquez Losada, hello_world, Makille, R. Zagórski, Redman
166	Меню	Bhargavi Yamanuri, Chip, Daniel Nugent, Hi I'm Frogatto, honk, Iman Hamidi
167	Место нахождения	Alex Chengalan, Aryan, BadCash, Daniel Nugent, Hiren Patel, Mahmoud Ibrahim, MidasLefko, Pablo Baxter, RamenChef, Stephen Leppik
168	Модульное тестирование в Android с помощью JUnit	abhi, Andre Perkins, AndroidMechanic, Eugen Martynov, honk, Lewis McGeary, N J, Namnodorel, Patrick Dattilio, Rolf ツ
169	Мультидекс и предел метода Dex	Adarsh Ashok, Ben, bigbaldy, cdeange, Daniel Nugent, Gabriele Mariotti, Mike, Pongpat, R. Zagórski, Shirane85

170	Написание UI-тестов - Android	Atif Farrukh, Daniel Nugent, Gabriele Mariotti, honk, Jon Adams, originx
171	Настройка Jenkins CI для Android-проектов	honk, Ichthyocentaurs
172	Начало работы с OpenGL ES 2.0+	MarGenDo
173	Неявные намерения	Blundering Philosopher, Daniel Nugent, mnoronha, Pratik Butani, SoroushA
174	Нижние листы	Daniel Nugent, Gabriele Mariotti, Magesh Pandian, MiguelHincapieC, RamenChef, Stephen Leppik, sud007, Zarul Izham
175	Низкая энергия Bluetooth	Roberto Betancourt
176	Нить	Daniel Nugent, PRIYA PARASHAR, RamenChef
177	Нож для масла	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
178	область	bdash, Dan, EpicPandaForce, Hi I'm Frogatto, iurysza, null pointer, RamenChef, Stephen Leppik, sukumar
179	Обнаружение жеста	mpkuth
180	Обнаружение события Shake в Android	N-JOY, tynn, Xiaozou
181	Обработка глубоких ссылок	Doron Yakovlev-Golani, Harsh Sharma, mnoronha, Tanis.7x
182	Обработка событий касания и движения	honk, Zoe
183	Обработчик аннотации	krishan
184	Обратный вызов	Atif Farrukh, honk, RamenChef, Stephen Leppik
185	обслуживание	adao7000, AndroidMechanic, Apoorv Parmar, BadCash, B-GangsteR, Daniel Nugent, g4s8, Hiren Patel, JonasCz, Lazai, Lucas Paolillo, Michael Spitsin, Nougat Lover,

		rakeshdas, Vinícius Barros
186	Окио	Adhikari Bishwash
187	Определить значение шага (приращение) для пользовательского RangeSeekBar	Romu Dizzy
188	Оптимизация производительности	honk, Jonas Köritz
189	Оптимизация ядра Android	honk, Sneh Pandya
190	Оптимизированный видеообзор	Chip
191	Опубликовать в Play Store	Carlos Borau, Fabio, mnoronha, Zoe
192	Отображение портов с использованием библиотеки Cling на Android	Shinil M S
193	Переходы общего элемента	noongiya95
194	Пикассо	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
195	Планирование работы	RamenChef, reflective_mind
196	погрузчик	chandsie, g4s8, jefry jacky, Marcus Becker, RamenChef, Stephen Leppik
197	Поддержка экранов с различными разрешениями, размерами	Eduardo, Guilherme Torres Castro, kalan, mpkuth, Onur, ppeterka
198	Подключение Wi-Fi	4444, AndroidMechanic, Daniel Nugent, gus27
199	Подпишите свое	Gabriele Mariotti, M M

	приложение для Android	
200	Пожарная авария	AndiGeeky, Gabriele Mariotti, honk, RamenChef, Stephen Leppik, Zarul Izham
201	Пожарная безопасность Firebase	Gabriele Mariotti, shikhar bansal, Shubham Shukla, Zarul Izham
202	Показатели отображения устройства	Daniel Nugent, Hiren Patel, Talha, W3hri
203	Покрасить	Nicolas Maltais
204	Получение имен системных шрифтов и использование шрифтов	Adil Saiyad, honk
205	Пользовательские шрифты	Daniel Nugent, Erik Ghonyan, Gabriele Mariotti, Hiren Patel, honk, kit, Nougat Lover, Simon Schubert, Stanojkovic, Sujith Niraikulathan
206	Посмотреть список	A.A., brainless, Daniel Nugent, Diti, Douglas Drumond, Fabian Tamp, Gabriele Mariotti, Hiren Patel, Mr.7, Ruben Pirotte, Saeed-rz, shaonAshraf, Squidward
207	Поставщик услуг	Andrew Siplas, cdeange, Daniel Nugent, Dinesh Choudhary, Lewis McGeary, RamenChef
208	Преобразование vietnamese строки в английскую строку Android	1SStorm
209	Преобразование речи в текст	Hitesh Sahu, honk, RamenChef, Stephen Leppik
210	Проведите по экрану	Chirag SolankI, Daniel Nugent, Gabriele Mariotti, Malek Hijazi, RamenChef, Stephen Leppik
211	Проверить подключение к данным	sukumar, Suresh Kumar
212	Проверка подключения к Интернету	AndiGeeky, Bill, Daniel Nugent, gbansal, Ichigo Kurosaki, Jon Adams, sukumar, TameHog, Yousha Aleayoub

213	Проверка электронной почты	Hiren Patel, honk, iravul, Nicolas Maltais
214	Программирование на Android с Kotlin	Gian Patrick Quintana, Govinda Paliwal, Oknesif, Zarul Izham
215	Просмотр объявлений Google	Egek92, RamenChef, ReverseCold, Stephen Leppik, Zarul Izham
216	Публикация библиотеки в репозитории Maven	Farid
217	Публиковать файл .aar для Apache Archiva с Gradle	Marian Klühspies
218	Разархивировать файл в Android	Arth Tilva, Daniel Nugent, mnoronha
219	Разбиение страницы в RecyclerView	Muhammad Younas
220	Разработка Android-игр	Zoe
221	Разрешения времени выполнения в 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
222	Распознавание активности	Pablo Baxter
223	Регистрация и использование 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
224	Редактировать текст	Daniel Nugent, Gabriele Mariotti, Kaushik NP, Muthukrishnan Rajendran, Rubin Nellikunnathu, Yousha Aleayoub

225	Режим PorterDuff	Adarsh Ashok, AndroidMechanic, Knossos, PhilLab, S.D., Vasily Kabunov
226	Режим дозировки	Daniel Nugent, Fabio, honk, NitZRobotKoder, RamenChef, Rosário Pereira Fernandes, Rupali
227	Ресурсы	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
228	Сжатие изображения	Hiren Patel, mnoronha
229	Синхронизация данных с адаптером синхронизации	Arpit Gandhi, mnoronha
230	скольжение	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
231	Создание класса Singleton для сообщения Toast	Emad, Ishan Fernando
232	Создание обратных совместимых приложений	Jon Adams, mnoronha, RamenChef, SoroushA
233	Создание окна Overlay (всегда на вершине) Windows	honk, mnoronha, NitZRobotKoder, Rupali, Sujith Niraikulathan
234	Создание пользовательских ПЗУ Android	honk, Pradumn Kumar Mahanta
235	Создание пользовательских представлений	AndroidMechanic, Barend, Bartek Lipinski, Charuක, 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

236	Создание собственных библиотек для приложений Android	EpicPandaForce, honk, mnoronha
237	Создание экрана заставки	honk, Kiran Benny Joseph, Zoe
238	Сплит-экран / многоэкранная деятельность	Vishal Puri
239	Строгий режим политики: инструмент, чтобы поймать ошибку во время компиляции.	Shekhar
240	Таймер обратного отсчета	privatestaticint
241	Текст для речи (TTS)	Ahmad Aghazadeh, honk, Jordan, Lukas, nibarius, Peter Taylor, RamenChef, Stephen Leppik
242	Тема DayNight (AppCompat v23.2 / API 14+)	Ishita Sinha
243	Тема, Стиль, Атрибут	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
244	Тестирование пользовательского интерфейса с помощью эспрессо	Daniel Nugent, Gabriele Mariotti, Jason Robinson, Michael Vescovo, Milad Faridnia, N J, RamenChef, Víctor Albertos
245	Тост	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
246	Уведомления	alexey polusov, bricklore, Da-Jin C, Daniel Nugent, Dus, gbansal, Jeeter, piotrek1543, RediOne1, Rupali, TR4Android, weston

247	укротитель	Daniel Nugent, Floern, Hasif Seyd, Lewis McGeary, Mike Scamell, Muhammed Refaat, Sweeper, Tomik, TR4Android
248	Улучшение диалогов оповещений	Adil Saiyad, honk
249	Улучшение производительности Android с помощью знаковых шрифтов	Beto Caldas, honk, Neeraj
250	умысел	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, Charuಎ, 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, noɥʎdʎzɐɔ, 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, ɥolɐəz əɥɥ qoq, Tyler Sebastian, vasili111, Vasily Kabunov, Vinay , Vivek Mishra, Xaver Kapeller, younes zeboudj, Yury Fedorov, Zoe
251	Универсальный загрузчик изображений	Greg T, honk, Jon Adams, priyankvex, Stephen Leppik
252	Установка приложений с помощью ADB	Ahmad Aghazadeh, fyfyone Google, Laurel, Xaver Kapeller

253	Утечки памяти	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
254	Учетные записи и учетные записи	gaara87, systemovich
255	Файл манифеста	John Snow, Jon Adams, kit, mayojava, Menasheh
256	Форматирование строк	Beena, Daniel Nugent, gaara87, Greg T, Michele, RamenChef, Suresh Kumar
257	Форматирование телефонных номеров с рисунком.	Pedro Varela
258	Фрагменты	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
259	фреска	Alexander Oprisnik, Daniel Nugent, honk, Nilesh Singh, Zarul Izham
260	Хранение файлов во внутреннем и внешнем хранилищах	Amit Vaghela, Andrew Brooke, AnV, Daniel Nugent, Gabriele Mariotti, Nickan B, Uttam Panchasara
261	Цвета	Carlos Borau, Dalija Prasnikar, Daniel Nugent, Erfan Mowlaei, Jon Adams, N J, Sujith Niraikulathan
262	Чтение штрих-кода и QR-кода	FlyingPumba
263	Что такое ProGuard? Что используется в Android?	Ayush Bansal, Daniel Nugent, Ghanshyam Sharma, Pratik Butani
264	Шаблоны проектирования	Adhikari Bishwash, honk, Steve.P
265	Шифрование /	honk, HoseinIT, Robert

	дешифрование данных	
266	Эмулятор	Ahmad Aghazadeh, Dan Hulme, fyfyone Google, honk, rekire, Rubin Nellikunnathu, ThomasThiebaud
267	Эффективная загрузка растровых изображений	iDevRoids