

無料電子ブック

学習

Android

Free unaffiliated eBook created from
Stack Overflow contributors.

#android

1
1: Android 2
2
2
Examples 3
Android Studio 3
Android Studio 3
/ 4
4
4
Android Studio 4
4
4
API 6
11
12
16
Android 16
Android Studio 16
APK 17
IDEAndroid 18
18
Android SDK 18
18
19
21
21
22
22
22

23

AVDAndroid 24

2: 9 30

30

Examples 30

30

31

31

3: ACRA 33

33

33

33

Examples 33

ACRAHandler 33

34

34

4: ADBAndroid Debug Bridge 35

35

35

Examples 35

35

35

36

36

WiFiADB 39

39

40

USB 40

41

41

41

Wi-Fi/ 42
42
IP 42
adb/ 43
43
ADB 44
45
1adb 45
2 45
46
46
46
47
48
LinuxADB 49
Android 6.0 49
// 49
50
50
5: adb 51
51
51
51
Examples 51
ADBAndroid 51
53
API 23+ 53
54
54
chmod 55
adb/ 56
57

57
/ 57
Android 57
6: ADB 59
Examples 59
59
59
apk 59
7: AdMob 60
60
60
60
Examples 60
60
Build.gradle 60
60
XML 60
Java 61
8: AIDL 63
63
Examples 63
AIDL 63
9: AlarmManager 65
Examples 65
65
65
Android 66
API23 + DozeAlarmManager 66
10: Android JavaJNI 68
68
Examples 68

JNI ... 68
Java ... 69
JNI ... 70
11: Android NDK ... 71
Examples ... 71
Android ... 71
... 72
Android.mk ... 72
ndk ... 72
12: Android Paypal ... 73
... 73
Examples ... 73
paypal ... 73
13: Android Places API ... 75
Examples ... 75
... 75
Places API ... 76
... 77
Google ... 78
PlaceAutocomplete ... 79
14: Android Studio ... 81
... 81
Examples ... 81
... 81
... 82
... 83
15: Android StudioOpenCV ... 85
... 85
Examples ... 85
... 85
16: Android Vk Sdk ... 93
Examples ... 93

93
17: Android-x86 in VirtualBox 95
95
Examples 95
95
SDCARD 95
98
18: Android 102
102
Examples 102
102
AppCompatActivity 102
LiveDataViewModel 103
104
LiveData 106
107
19: Android 109
Examples 109
109
110
Jitpack.io 110
20: Android 112
Examples 112
112
21: Android 115
Examples 115
RAM 115
CPU 115
I / O 117
22: AndroidROM 119
Examples 119
119

Java 119
..... 119
..... 119
23: Android 121
Examples 121
API> 19 121
SoundPool 122
24: Android 124
Examples 124
UI 124
..... 125
logcat 127
/ 128
Android Studio 129
Android 131
Android Studio 132
Android Studio 132
Gradle 133
..... 134
25: AndroidCling 136
Examples 136
AndroidCling 136
NAT 136
26: Android 138
Examples 138
Android 138
..... 139
..... 139
27: Android 140
Examples 140
..... 140
Android 140

res 141
"res" 142
143
145
28: Android 149
Examples 149
149
29: AndroidJava 150
150
Examples 150
RetrolambdaJava 8 150
30: Android 153
153
153
Examples 153
CanvasSurfaceView 153
31: Android 160
160
Examples 161
Android 161
32: Android 163
Examples 163
163
33: Android/ 165
165
Examples 165
DateUtils.formatDateTime 165
Android/ 165
/ 165
34: AndroidFuseView 167
167

Examples....167
hikrandroid.view.View....167
35: AndroidJenkins CI....176
Examples....176
AndroidJenkins....176
I....176
IIAndroidJenkins....177
IIIAndroidJenkins....178
36: AndroidFileIO....180
....180
....180
Examples....180
....180
....180
....181
SD....181
"Invisible MTP files"....182
....182
37: AndroidFacebook SDK....184
....184
....184
Examples....184
AndroidFacebook....184
Facebook....186
Facebook....187
Facebook/....188
Facebook....189
38: AndroidGoogle Maps API v2....190
....190
....190
Examples....190

Google 190
Google 191
201
MapViewGoogle 202
Google 204
SH1 210
Google 211
UISettings 211
SHA1 212
InfoWindow 213
215
39: AndroidGradle 216
216
216
216
AndroidGradle - 216
Examples 217
build.gradle 217
DSL 217
217
DSL 218
218
219
signingConfig 219
220
220
221
221
BuildConfigField 221
ResValue 223
"dependencies.gradle" 224

225

226

Android Studiobuild.gradle2 226

gradle 227

Gradle 227

ID 228

APK 229

Akeystore.properties 229

B 231

"version.properties" 231

apk 232

APK 232

gradleProguard 233

GradleAndroidStudioNDK 233

MyApp / build.gradle 233

MyApp / app / build.gradle 234

235

gradle 235

"unaligned" apk 237

237

238

versionnumber / buildconfigurationsgradle.properties 238

239

240

40: API-23 + 242

242

242

Examples 243

Android 6.0 243

URI 244

245

PermissionUtil ... 247
... 248
... 249
41: AsyncTask ... 251
... 251
Examples ... 251
... 251
... 251
... 252
... 252
AsyncTask ... 253
... 253
... 254
AndroidAsyncTask ... 254
Android AsyncTask ... 254
Android AsyncTask ... 255
WeakReference ... 258
... 259
AsyncTask ... 259
THREAD_POOL_EXECUTOR ... 259
SERIAL_EXECUTOR ... 260
1 ... 261
42: AudioManager ... 263
Examples ... 263
... 263
... 263
43: AutoCompleteTextView ... 264
... 264
Examples ... 264
AutoCompleteTextView ... 264
CustomAdapterClickListenerFilter ... 264

activity_main.xml 264
row.xml 265
strings.xml 265
MainActivity.java 265
People.java 266
PeopleAdapter.java 267
44: BluetoothBluetooth LE API 269
269
Examples 269
269
Bluetooth 269
270
Bluetooth 270
Bluetooth 271
Bluetooth 272
45: Bluetooth 278
278
Examples 278
BLE 278
GATT 278
279
280
BLE 281
281
46: BroadcastReceiver 284
284
Examples 284
284
BroadcastReceiver 285
LocalBroadcastManager 285
Bluetooth 286
286

286

286

287

287

BOOT_COMPLETEDBroadcastReceiver 287

LocalBroadcastManager 288

2 289

290

290

Android 291

47: CardView 292

292

292

292

293

Examples 293

CardView 293

CardView 294

295

CardViewLollipop 295

TransitionDrawableCardView 297

48: CleverTap 298

298

298

Examples 298

SDK 298

298

49: ConstraintLayout 299

299

299

299

300

300
Examples 300
ConstraintLayout 300
301
50: ConstraintSet 303
303
Examples 303
ContraintLayoutConstraintSet 303
51: ContentProvider 304
304
Examples 304
304
52: ContentValuesSQLite 308
Examples 308
SQLite 308
308
308
53: DayNightAppCompat v23.2 / API 14+ 309
Examples 309
DayNight 309
54: Drawables 311
Examples 311
311
311
312
313
55: EditText 315
Examples 315
EditTexts 315
InputType 317
`inputype` 318

SoftKeyboard ... 319
... 320
56: ExoPlayer ... 323
Examples ... 323
ExoPlayer ... 323
ExoPlayer ... 323
TrackRenderer ... 324
57: FileProvider ... 325
Examples ... 325
... 325
... 325
FileProvider ... 325
URI ... 326
... 326
58: Firebase App Indexing ... 327
... 327
Examples ... 329
HTTP URL ... 329
AppIndexing API ... 331
59: Firebase Cloud Messaging ... 333
... 333
Examples ... 333
AndroidFirebase Cloud Messaging Client ... 333
... 333
webView ... 334
... 335
... 336
60: Firebase ... 338
Examples ... 338
Firebase ... 338
... 339

61: Firebase 340
..... 340
..... 340
Examples 340
Firebase Realtime DataBase 340
..... 341
Firebase 341
1 342
2JSON 342
3 342
4 343
..... 344
..... 344
firebase JSON 346
firebase 347
..... 349
..... 350
62: FloatingActionButton 351
..... 351
..... 351
..... 351
..... 351
..... 351
Examples 351
FAB 351
FloatingActionButton 353
FloatingActionButton 354
FloatingActionButton 357
63: Google Awareness APIs 358
..... 358
Examples 358

Snapshot API 358
Snapshot API 359
Snapshot API 359
Snapshot API 359
Snapshot API 360
Fence API 360
Fence API 361
64: GoogleAPI 363
363
363
Examples 363
AndroidGoogle 363
Google 374
DriveContent 374
375
376
65: Google 377
377
377
Examples 377
Google 377
66: Google 380
Examples 380
380
380
67: GradleApache Archiva.aar 383
Examples 383
383
68: GreenDAO 385
385
Examples 385

SELECTINSERTDELETEUPDATE 385
GreenDAO 3.X 387
GreenDao v3.X 388
69: GreenRobot EventBus 391
391
391
Examples 391
391
391
392
392
70: Gson 394
394
394
Examples 395
JSONGson 395
GsonJSONenum 396
Gson 397
AutoValueGsonJSON/ 397
GsonJSON 398
Gson 399
GsonJSON 400
Gson 400
RetrofitGson 401
Gsonjson 401
GsonJSON 402
Gson 404
71: HttpURLConnection 407
407
407
Examples 407
HttpURLConnection 407

HTTP GET 408
HTTP GET 409
multipart / form-dataHttpURLConnection 409
HTTP POST 412
HttpURLConnectionPOST 413
HTTPHttpURLConnection 414
..... 416
72: ImageView 418
..... 418
..... 418
..... 418
Examples 418
..... 418
..... 418
ImageView ScaleType - 419
ImageView ScaleType - CenterCrop 421
ImageView ScaleType - CenterInside 421
ImageView ScaleType - FitStartFitEnd 421
ImageView ScaleType - FitCenter 421
ImageView ScaleType - FitXy 421
..... 421
..... 425
MLRoundedImageView.java 426
73: IntentService 428
..... 428
..... 428
Examples 428
IntentService 428
..... 428
IntentService 429
74: JCodec 431
Examples 431
.....

431

431

75: JSON with Android with org.json 432

432

432

Examples 432

JSON 432

JSON 433

JSONObjectJSONArray 433

nullJSON 434

jsonnull-string 434

JsonReaderJSON 435

JSON 437

JSON 437

JSON 438

JSON 439

76: JUnitAndroid 440

440

Examples 440

440

440

440

Android Studio 441

Android Studio 441

Android 441

JUnit 444

444

445

446

447

448

77: KotlinAndroid 450

450

450

Examples 450

Kotlin 450

GradleKotlin 451

Kotlin 453

JavaKotlin 454

455

78: LruCache 456

456

Examples 456

456

456

457

79: MediaPlayer 458

458

458

Examples 460

460

460

461

462

462

462

1 462

MediaPlayer 463

Media Player 463

androidstudio 465

80: MediaSession 467

467

467

Examples 467

467

81: MediaStore 470

Examples 470

/ MP3 470

472

82: MultidexDex 474

474

474

dex 474

474

474

474

Examples 475

MultiDexApplicationMultidex 475

Multidex 475

476

476

MultiDex 477

Dexcount Gradle Plugin 477

MultiDexApplicationMultidex 478

83: MVP 479

479

479

MVP 479

479

Examples 479

Model View PresenterMVP 479

483

483

MVP 484

484

XML activity_login 484
LoginActivity.class 485
ILoginView 487
ILoginPresenter 487
ILoginPresenter.class 487
LoginPresenterCompl.class 487
488
UserModel.class 488
IUser.class 489
MVP 489
84: MVVM 491
491
Examples 492
MVVM 492
85: NavigationView 500
500
500
500
Examples 500
NavigationView 500
505
506
DividerItemDecoration 507
86: OkHttp 509
Examples 509
509
509
509
510
510
511

511

OkHttp ... 512

87: Okio ... 513

Examples ... 513

/ ... 513

PNG ... 513

... 514

88: OpenGL ES 2.0 ... 515

... 515

Examples ... 515

GLSurfaceViewOpenGL ES 2.0+ ... 515

GLSL-ES ... 516

89: PackageManager ... 518

Examples ... 518

... 518

... 518

... 518

PackageManager ... 519

90: Ping ICMP ... 521

... 521

Examples ... 521

Ping ... 521

91: Play ... 522

Examples ... 522

... 522

92: PorterDuff ... 524

... 524

... 524

Examples ... 525

PorterDuff ColorFilter ... 525

PorterDuff XferMode ... 525

PorterDuffXfermode ... 525

93: ProGuard - 527
Examples 527
527
ProGuard 529
529
530
ProGuard 531
94: ProGuard Android 533
533
Examples 533
533
95: RecyclerView 535
535
535
535
536
536
536
Examples 536
RecyclerView 536
537
RecyclerView 538
RecyclerView/ 539
ItemViewTypeViewHolders 542
RecyclerViewSearchView 543
recyclerView 544
546
SortedList 548
RecyclerView 550
552
552
RecyclerView 555

96: RecyclerView onClickListener 557
Examples 557
..... 557
KotlinRxJava 558
OnLongClickOnClick 559
..... 560
..... 562
Item Click Listener 563
RecyclerView 565
97: RecyclerViewLayoutManager 568
Examples 568
GridLayoutManager 568
Gridlayoutrecyclerview 570
LinearLayoutManager 572
..... 572
..... 572
..... 573
RecyclerViewViewHolder 573
..... 575
RecyclerViewPlaceListAdapter 575
..... 576
StaggeredGridLayoutManager 576
98: RecyclerView 579
..... 579
..... 579
..... 579
..... 579
..... 579
..... 579
javadoc 579
Examples 579

579
ItemDecoration 581
RecyclerView 581
DividerItemDecoration 584
RecycleViewGridLayoutManagerItemOffsetDecoration 584
99: RecyclerView 586
586
Examples 586
MainActivity.java 586
100: RenderScript 591
591
Examples 591
591
591
RenderScript 591
RenderScript 592
RenderScript 593
594
594
594
RenderScriptAPI 595
595
JavaRenderScript 596
596
597
598
598
599
601
BlurBitmapTask.java 601
602

101: RoboGuice 604
Examples 604
..... 604
Gradle 604
@ContentView 604
@InjectResource 604
@InjectView 605
RoboGuice 605
102: RxJavaRetrofit2 608
Examples 608
RxJavaRetrofit2 608
RxJava 609
..... 611
103: SearchView 613
Examples 613
Appcompat SearchView with RxBindings 613
SearchView 615
SearchView 618
104: SensorManager 620
Examples 620
..... 620
..... 621
..... 621
105: SharedPreferences 623
..... 623
..... 623
..... 623
..... 624
..... 624
Examples 624
SharedPreferences 624
..... 625

SharedPreferences....626
SharedPreferences....627
SharedPreferences....628
SharedPreferences....628
SharedPreferences....633
getPreferencesintVS getSharedPreferencesStringint....633
....634
SharedPreferences....634
SharedPreferences....635
StringSetpre-Honeycomb....636
EditTextPreference....637
106: SpannableString....639
....639
Examples....639
TextView....639
....642
107: SparseArray....644
....644
....644
Examples....645
SparseArray....645
108: SQLite....647
....647
....647
Examples....647
SQLiteOpenHelper....647
....648
onUpgrade....648
....649
AndroidSQLite....651
....655
....656

656

SQLite 657

658

661

662

109: SurfaceView 664

664

Examples 664

SurfaceView 664

110: TabLayout 670

Examples 670

ViewPagerTabLayout 670

111: TensorFlow 671

671

671

Examples 671

671

112: TextInputLayout 672

672

672

Examples 672

672

672

673

673

TextInputEditText 674

TextInputLayout 674

113: TextView 676

676

676

676

Examples 676

676

TextView 676

Spannable TextView 679

TextView 680

TextView 681

681

681

682

RelativeSizeSpan 684

TextViewPinchzoom 685

2TextView 687

114: TransitionDrawable 689

Examples 689

2 689

1XML 689

2XMLImageView 689

3onCreateXMLonClick 689

TransitionDrawable 690

115: Twitter API 691

Examples 691

Twitter 691

116: Typedef@IntDef@StringDef 693

693

Examples 693

IntDef 693

694

117: UIAutomatorUI 695

695

695

Examples 695

UIAutomator 695

UIAutomatorViewer ... 696
UIAutomator ... 697
118: UI - Android ... 698
... 698
... 698
... 698
JUnit ... 698
... 698
... 698
Examples ... 698
MockWebServer ... 698
IdlingResource ... 701
... 701
... 702
... 702
... 703
JUnit ... 703
119: UI ... 705
Examples ... 705
... 705
120: VectorDrawableAnimatedVectorDrawable ... 706
Examples ... 706
VectorDrawable ... 706
... 706
... 707
Basic AnimatedVectorDrawable ... 708
... 709
AppCompat ... 711
121: VideoView ... 712
Examples ... 712
VideoView Create ... 712

VideoViewURL ... 712
122: ViewFlipper ... 714
... 714
Examples ... 714
ViewFlipper ... 714
123: ViewPager ... 716
... 716
... 716
Examples ... 716
ViewPager ... 716
ViewPagerTabLayout ... 718
PreferenceFragmentViewPager ... 719
ViewPager ... 721
ViewPager ... 722
ViewPagerTabLayout ... 722
TabLayout ... 723
selected_dot.xml ... 723
default_dot.xml ... 723
tab_selector.xml ... 724
OnPageChangeListener ... 724
124: WebView ... 726
... 726
... 726
Examples ... 726
WebViewJavaScript - ... 726
JavascriptJavaAndroid ... 726
JavaJavascript ... 727
... 728
WebView ... 728
webviewlogcat ... 729
ChromeAndroid ... 729
AndroidUSB ... 729

Android ... 729
/ Webview ... 730
125: Wi-Fi ... 731
Examples ... 731
WEP ... 731
WPA2 ... 731
... 732
126: XMPP ... 735
Examples ... 735
XMPP ... 735
127: Xposed ... 744
Examples ... 744
Xposed ... 744
... 744
128: Youtube-API ... 747
... 747
Examples ... 747
StandAlonePlayerActivity ... 747
YouTubeBaseActivity ... 747
YoutubePlayerFragment Activty ... 748
YouTube Player API ... 751
AndroidYouTubeAPI ... 753
129: Android ... 757
... 757
Examples ... 757
... 757
TabLayout ... 760
130: AccountManager ... 762
Examples ... 762
/ ... 762
131: ... 765
... 765

765
766
766
Examples 766
766
Android 767
launchMode 770
771
771
SingleTask 771
SingleInstance 771
setContentViewUI 771
772
772
772
773
773
774
132: AlertDialog 776
776
Examples 776
... 776
133: 779
Examples 779
ImageView 779
/ 780
TransitionDrawable 780
ValueAnimator 781
ObjectAnimator 782
ViewPropertyAnimator 783
783

134: 785
Examples 785
..... 785
..... 785
1 785
2 785
3 785
4 785
5 786
6 789
In-App v3 790
135: 791
..... 791
Examples 791
..... 791
136: Google 792
..... 792
Examples 792
Google Auth 792
Google SignIn 792
137: ORMLite 794
Examples 794
Android OrmLite over SQLite 794
..... 794
..... 795
SQLite 796
138: Zip 799
Examples 799
Zip 799
139: Genymotion 801

801
Examples 801
Genymotion 801
1 - VirtualBox 801
2 - GenymotionGenymotion 801
3 - Genymotion 801
4 - Genymotion 801
5 -genymotionAndroid Studio 801
6 - Android StudioGenymotion 802
GenymotionGoogle 802
140: API 803
803
Examples 803
Android 803
Android Fingerprint API 804
141: Retrolambda 813
813
Examples 813
813
142: 815
815
815
815
815
Examples 815
815
816
816
143: 820
Examples 820
820

820
820
821
822
144: 824
824
824
825
825
825
singleTasksingleTop 825
Examples 826
826
826
OriginActivity 826
DestinationActivity 827
828
829
829
DetailActivity 830
830
URL 831
831
831
832
832
URI 832
833
CustomTabsIntent for Chrome 834
834

835

835

836

836

Google 837

837

840

840

840

841

841

843

844

145: 846

846

Examples 846

- 846

846

AppWidgetProvider 846

2 847

Android Studio/ 848

==> New ==>==> App 848

146: UI 850

850

850

850

Examples 850

850

851

DrawerLayout 851

UI 852

UI 852

853
854
854
857
857
onView 857
858
860
EditText 862
862
862
862
147: 864
864
Examples 864
864
AVD 867
868
869
148: 870
Examples 870
870
149: Windows 871
Examples 871
871
WindowManager 871
SYSTEM_ALERT_WINDOW6.0 871
150: 873
873
Examples 873
873
874

151: 875
875
Examples 875
875
876
877
878
878
AsyncTask 879
879
879
879
879
AsyncTaskLoader 879
880
880
152: 882
882
882
Examples 882
882
882
153: RangeSeekBar 884
884
884
Examples 885
7 885
154: 886
Examples 886
886
888

890

CustomView ... 893

drawableRightSVG / VectorDrawable ... 895

custom_edit_drawable - c_d_e ... 895

build.gradle ... 895

c_e_d_compound_view.xml ... 895

attrs.xml ... 895

EditTextWithDrawable.java ... 896

... 897

activity_main.xml ... 897

MainActivity.java ... 897

... 897

155: ... 899

Examples ... 899

... 899

... 899

TextView ... 899

TextViewxmlJava ... 899

... 900

... 901

... 901

Android O ... 902

156: 2 API ... 904

... 904

... 904

Examples ... 904

TextureView ... 905

157: ... 914

Examples ... 914

... 914

AndroidManifest.xml ... 914

... 916

919

922

URI 922

158: 926

Examples 926

926

926

159: 928

Examples 928

Google Play 928

Google Play 928

160: 930

930

Examples 930

- 930

Fabric-Crashlytics 930

Fabric IDE 931

ACRA 935

936

Sherlock 937

161: 939

939

939

Examples 939

939

CoordinatorLayout.Behavior 939

940

XML 940

940

SwipeDismissBehavior 940

941

162: URL 943
Examples 943
Instagram OAuthURL 943
163: 945
..... 945
..... 945
..... 945
Examples 945
..... 945
164: 946
..... 946
..... 946
Examples 946
..... 946
..... 946
..... 947
..... 948
AIDL 949
..... 950
165: 954
..... 954
..... 954
..... 954
..... 954
..... 954
px 954
..... 954
mm 954
pt 955
dpdip 955
sp 955

Examples....955
....955
dpsp....956
Android....957
166: 958
....958
Examples....958
....958
....959
167: 961
....961
Examples....961
....961
TextView....961
168: 962
....962
Examples....962
....962
169: 964
Examples....964
....964
170: 965
....965
Examples....965
....965
JobService....965
AndroidManifest.xmlJobService....965
....966
171: 968
Examples....968
Android Studio....968

Android ... 968
ADB ... 969
ADBPC ... 969
... 969
172: ... **971**
Examples ... 971
... 971
... 971
... 971
173: ... **972**
... 972
... 972
... 972
... 972
Examples ... 972
Snackbar ... 972
... 973
... 974
... 974
... 975
... 975
174: ... **977**
Examples ... 977
... 977
... 977
175: ... **980**
... 980
Examples ... 980
... 980
... 982
1 ... **982**
2 ... **982**

3 983
176: 985
Examples 985
..... 985
177: 988
..... 988
Examples 988
RecyclerView 988
..... 988
178: SharedPreferences 990
..... 990
..... 990
..... 990
..... 990
Examples 990
..... 990
179: SharedPreferences 992
..... 992
..... 992
..... 992
..... 992
Examples 992
..... 992
180: 994
Examples 994
- 994
181: 995
..... 995
..... 995
Examples 995
..... 995

996

DialogFragment ... 996

DatePickerDialog ... 998

DatePicker ... 999

DatePickerDialog ... 999

AppcompatMaterial Design AlertDialog ... 1000

AlertDialogListView ... 1001

EditText ... 1001

... 1003

... 1003

182: ... 1006

Examples ... 1006

... 1006

... 1007

GetCurrentRealTime ... 1007

183: 2 ... 1009

... 1009

... 1009

Examples ... 1009

... 1009

... 1011

... 1011

@Component@Subcomponent= {...} ... 1012

build.gradleDagger 2 ... 1012

... 1013

184: ... 1016

Examples ... 1016

... 1016

185: ... 1020

... 1020

... 1020

Examples ... 1020

1020
1021
1022
186: 1023
1023
Examples 1023
1023
187: SyncAdapter 1025
1025
Examples 1025
1025
188: / 1035
1035
Examples 1035
AES 1035
189: 1037
1037
Examples 1037
1037
1038
1039
1040
1041
RecyclerView 1042
1042
XML 1042
1042
Binding 1043
1044
1046
intboolean 1047
1047

BindingAdapter 1047
190: 1049
Examples 1049
..... 1049
ConnectivityManager 1049
..... 1049
191: 1050
Examples 1050
..... 1050
..... 1050
..... 1050
API 21+ 1051
API 21+ 1051
API 23+ 1051
API 19+ 1052
API 21+ 1052
..... 1052
1 1053
..... 1054
192: 1056
..... 1056
Examples 1056
..... 1056
..... 1056
193: TTS 1058
Examples 1058
..... 1058
APITextToSpeech 1060
194: 1063
..... 1063
Examples 1063
..... 1063

1064
1064
195: 1065
Examples 1065
1065
1065
pxdpdppx 1065
196: 1066
1066
1066
1066
1066
1066
Examples 1067
1067
1067
1068
Toast 1068
1069
AsyncTask 1070
197: 1071
1071
1071
1071
1071
Examples 1072
1072
198: 1074
1074
Examples 1076
1076
Android 1077

199: QR 1078
..... 1078
Examples 1078
QRCodeReaderViewZxing 1078
..... 1078
..... 1078
200: 1080
..... 1080
..... 1080
Examples 1080
Parcelable 1080
ParcelableParcelable 1081
ParcelableEnum 1082
201: /PTTLWP 1084
..... 1084
Examples 1084
Sonim Devices 1084
PTT_KEY 1084
YELLOW_KEY 1084
SOS_KEY 1084
GREEN_KEY 1084
..... 1084
RugGear 1085
PTT 1085
202: 1086
Examples 1086
AES 1086
203: 1089
..... 1089
..... 1089
..... 1089

Examples 1089
ButterKnife 1089
ButterKnife 1091
1092
1092
1092
1092
ViewHolder 1093
1093
1093
1094
ButterKnife 1094
ButterKnife 1095
Android Studio ButterKnife 1096
204: 1097
1097
Examples 1097
ViewHolderView 1097
205: 1098
1098
Examples 1098
1098
HandlerThreads 1098
1098
UI 1098
Runnable 1098
HandlerThread 1099
1099
javax.swing.Timer 1100
206: 1102
1102

1102

Examples 1102

AndroidPicasso 1102

Gradle **1102**

Maven **1102**

..... 1102

..... 1103

..... 1103

Picasa 1105

..... 1105

..... 1105

..... 1106

PicasaHtml.fromHtmlImageGetter 1106

..... 1108

207: **1110**

..... 1110

..... 1110

..... 1110

Examples 1110

LRU 1110

208: **1112**

..... 1112

..... 1112

Examples 1112

Android 1112

209: **1115**

..... 1115

..... 1115

Firebase - 1115

..... 1115

Examples 1115

Firebase 1115
Firebase 1116
Firebase 1118
Firebase 1119
1120
Firebase 1121
Firebase 1123
Firebase Cloud Messaging 1129
FirebaseFCM SDK 1129
1129
AndroidFirebase 1131
Firebase 1131
SDK 1131
Firebase/ 1132
FCM 1134
Firebase 1143
210: 1145
1145
1145
Examples 1145
FastjsonJSON 1145
JSON String 1147
211: 1148
1148
1148
Examples 1148
1148
OkHttp 3 1149
DraweeControllerJPEG 1149
212: 1151
1151

Examples 1151
ProgressBar 1151
ProgressBar 1151
1153
Tinting ProgressBar 1156
1157
1158
1158
1159
1159
1160
213: SDK 1162
1162
1162
1162
Examples 1162
SDK 1163
214: 1164
1164
Examples 1164
1164
1164
1164
1165
1165
1165
215: 1167
1167
1167
1167
Examples 1168

VectorDrawable 1168
VectorDrawable xml 1169
VectorDrawableSVG 1169
216: 1172
Examples 1172
1172
ChuynchuiTingthunhchuikhôngdu 1172
217: 1173
1173
Examples 1173
onClickListener 1173
1173
XML1 1173
1174
1174
1174
1175
1176
218: 1181
1181
1181
Examples 1181
GoogleBottomSheetBehavior 1181
1188
1188
BottomSheetDialogFragment 1189
BottomSheetDialog 1190
ExpandedBottomSheet DialogFragment 1190
219: 1192
1192
1192
1192

Examples 1192
1192
BottomNavigationView 1193
/ 1194
3 1194
220: 1196
1196
1196
1196
1196
1196
Examples 1197
GETStringRequest 1197
1197
NetworkImageView 1197
JSON 1199
[] 1199
1200
POSTStringRequest 1201
HTTPVolley 1203
json request with volley 1205
JSONArray 1206
221: 1208
1208
Examples 1208
1208
1209
222: 1210
1210
1210
1210
Examples 1210

1210

1210

1211

1. 1211

2. 1212

1212

1213

223: 1215

Examples 1215

1215

1. 1215

2. Context 1215

3. 1215

4. 1216

5. 1216

6. java.util.Observer 1216

AsyncTask 1216

1217

1218

LeakCanary 1219

1220

1 1222

2 1223

1225

224: 1227

1227

1227

Examples 1227

JSONJava 1227

JavaJSON 1227

1227

225: 1229

1229
Examples 1229
1229
1229
226: 2 1231
1231
1231
Dagger 2 API 1231
1231
Examples 1231
@Module@Singleton 1231
1232
@Modules@Inject 1232
@Component 1233
227: Maven 1234
Examples 1234
Maven.aar 1234
228: 1236
1236
1236
Examples 1236
CursorAdapter 1236
ArrayAdapter 1237
ArrayAdapterListView 1238
229: 1239
Examples 1239
1239
1240
1240
1241
1242
1242

1243

1244

/ 1245

strings.xml 1246

1246

1247

1247

9 1250

UI92011518 1250

1253

strings.xml 1254

230: Android 1256

1256

Examples 1256

1256

build.gradle 1257

231: 1259

1259

Examples 1259

LooperThread 1259

HandlerThread 1259

232: 1260

1260

1260

1260

LayoutParamsLayout_ Attributes 1260

RelativeLayouts 1261

Examples 1262

LinearLayout 1262

RelativeLayout 1263

1265

1268

1270

FrameLayout....1271

....1272

....1273

....1275

LinearLayout....1277

LayoutParams....1278

233: 2....1282

....1282

....1282

Examples....1282

GET....1282

Retrofit2....1285

....1286

OkHttp....1286

....1287

Retrofit....1288

Retrofit2....1290

Stetho....1292

2Xml....1293

GSONPOST....1295

Retrofit 2XMLURL....1297

234: 1300

....1300

....1300

Examples....1300

....1300

....1300

RealmList 1301

....1303

....1303

....1303

RxJava 1304
1304
1304
1305
1305
1306
1307
1307
235: 1309
1309
1309
1309
1309
Examples 1310
AsyncTaskLoader 1310
AsyncTaskLoader 1311
1312
1312
236: 1314
1314
1314
1314
1314
1314
1315
Examples 1315
logcat 1315
1317
1317
1318
1318

1319
1319
1319
1319
1319
Logcat 1319
1320
1321
Android Studio 1322
1325
237: 1326
1326
1326
LocationManager 1326
FusedLocationProviderApi 1327
1329
Examples 1335
API 1335
w / LocationRequest 1335
w / PendingIntentBroadcastReceiver 1337
LocationManager 1340
LocationManager 1341
1343
Geocoder 1346
BroadcastReceiver 1346
238: 1348
1348
Examples 1348
1348
1348
1348

SDK 1348
1349
1349
239: 1350
Examples 1350
API 1350
1352
240: 1353
Examples 1353
NetworkOnMainThreadException 1353
ActivityNotFoundException 1354
OutOfMemoryError 1354
DexException 1355
1355
1356
241: 1358
1358
1358
Examples 1358
2 1358
242: 1361
1361
1361
Examples 1361
1361
1362
Android - 1363
SDSDDB 1368
1368
243: 1371
1371

1371
Examples 1371
StrictMode for Thread Policies 1371
SQLLite 1371
244: 1372
Examples 1372
1372
245: I18NL10N 1378
1378
1378
Examples 1378
RTL 1378
RTL 1378
RTL 1379
1380
1380
1381
246: 1382
Examples 1382
1382
1382
1382
1383
1383
247: 1384
1384
1384
1385
1385
Examples 1385
newInstance 1385
1386

1388
1388
1388
1388
1389
1390
248: 1395
Examples 1395
DatePicker 1395
1397
249: 1399
1399
1399
1399
Examples 1399
1399
1400
250: VideoView 1401
1401
Examples 1401
ListViewVideoView 1401
251: 1413
1413
1413
Examples 1413
AppCompat 1413
1414
FloatingActionButtonFAB 1416
1416
TextInputLayout 1418
TabLayout 1419
RippleDrawable 1421

1425
1428
1429
DialogFragment 1430
1431
252: 1434
1434
Examples 1434
@NonNull 1434
1434
1435
253: 1437
1437
Examples 1437
Google Play ActivityRecognitionAPI 1437
PathSense 1439
254: 1442
1442
1442
1442
<intent-filter> 1442
<data> 1442
1443
Examples 1443
1443
1 1443
1444
httphttps 1444
1445
pathPrefix 1445
255: 1447

1447
1447
Examples 1447
1447
1448
ImageView 1448
RecyclerViewListView 1448
ImageView 1449
1450
1450
1451
1452
ImageView 1452
1453
256: 1454
1454
1454
Examples 1454
Android 1454
257: 1455
Examples 1455
1455
258: / 1458
Examples 1458
Android Nougat 1458
259: 1460
Examples 1460
1460
UI 1460
260: 1462
1462
1462

Examples 1462
xml 1462
1462
graditeLintOptions 1463
lint.xml 1464
JavaXMLlint 1464
Java 1464
XMLlint 1465
1465
261: 1466
Examples 1466
1466
262: 1467
1467
Examples 1467
1467
263: 1469
1469
Examples 1469
CrashlyticsBetaFastfile 1469
Fastfile 1471
264: 1473
Examples 1473
1473
1473
1473
1473
1473
Heads Up NotificationAndroid Marshmallow 1474
TicketerAndroid KitKat 1475
Android 6.0 Marshmallow 1475

Android 4.4.x KitKat ... 1476
... 1477
... 1478
- ... 1479
... 1479
... 1479
`Picasso` ... 1480
... 1481
... 1481
265: Android O ... **1483**
... 1483
... 1483
... 1483
Examples ... 1483
... 1483
266: ... **1489**
Examples ... 1489
... 1489
... 1489
267: ... **1490**
... 1490
Examples ... 1490
+17861234 5678 ... 1490
268: ... **1491**
Examples ... 1491
Google ... 1491
... 1492
... **1494**

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

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

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

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

1: Androidをいめる

Android Gradle Pluginのについてしくりたいは、 android-gradleドキュメントをごください。

あなたがエミュレータにがあるなら、あなたはGenymotionをることができます。のプランをし、なRAMのはなくてみます。

バージョン

バージョン	APIレベル	バージョンコード	
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	20111018
4.0.3	15	ICE_CREAM_SANDWICH_MR1	2011-12-16
4.1	16	JELLY_BEAN	2012-07-09
4.2	17	JELLY_BEAN_MR1	2012-11-13
4.3	18	JELLY_BEAN_MR2	2013724
4.4	19	KITKAT	20131031

バージョン	APIレベル	バージョンコード	
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 マシュマロ	2015-10-05
7.0	24	N Nougat	2016-08-22
7.1	25	N_MR1 N_MR1 MR1	2016104
8.0	26	O けプレビュー4	2017-07-24

Examples

Android Studioをする

Android Studioは、Googleがにサポートし、するAndroidIDEです。 Android Studioには、 Android SDK Managerがしています 。これは、アプリのをするためになAndroid SDKコンポーネントをダウンロードするツールです。

Android StudioとAndroid SDKツールのインストール

1. Android Studioをダウンロードしてインストールします 。
2. Android Studioをき、 Android SDKツールのアップデートにって、のSDKツールとSDKプラットフォームツールをダウンロードしてください。のしたパッケージをインストールするがあります。

   いSDKバージョンをしてビルドされたいプロジェクトでするがあるは、これらのバージョンもダウンロードするがあります

Android Studio 2.2、のOpenJDKのコピーがインストールにバンドルされており、すべてのAndroid StudioプロジェクトでされるJDK Java Development Kitです。これにより、OracleのJDKパッケージをインストールするがなくなります。バンドルされたSDKをするには、のをします。

1. Android Studioでプロジェクトをき、メニューバーの[ファイル]> [プロジェクトの]をします。
2. [**SDK**の]ページと[**JDK**の]で 、[めみ**JDK**をする]チェックボックスをオンにします。
3. [**OK**]をクリックします。

Android Studioをする

Android Studioでは、[ヘルプ]メニューから2つのファイルにアクセスできます。

- studio.vmoptions ヒープサイズやキャッシュサイズなど、StudioのJava Virtual MachineJVMのオプションをカスタマイズします。 Linuxマシンでは、Android Studioのバージョンにじて、このファイルのは*studio64.vmoptions*になることにしてください。
- idea.propertiesAndroid Studioのプロパティプラグインのフォルダパスやサポートされているファイルサイズなどをカスタマイズします。

テーマの*I*

あなたのみとしてすることができます。 `File->Settings->Editor->Colors & Fonts->`とあなたからしいテーマをダウンロードすることができtheme.Alsoをhttp://color-themes.com/あなたがダウンロードしたら`.jar.zip`ファイルを、にく`File -> Import Settings...`をし、ダウンロードしたファイルをします。

アプリのコンパイル

しいプロジェクトをするか、Android Studioでのプロジェクトをき、のボタンをします  ツールバーのにあるツールバーをしてします。グレーの、Android Studioがいくつかのファイルのインデックスをしくできるようにするには、1つがあります。そのはのステータスバーにされます。

シェルからプロジェクトをするは、Android Studioによってにされる`local.properties`ファイルがあることをしてください。 Android Studioをせずにプロジェクトをするがあるは、 `sdk.dir=`まり、SDKのインストールのパスがです。

シェルをき、プロジェクトのディレクトリにします。 `./gradlew aR`としてEnterキーをします。 `aR`は`assembleRelease`ショートカットで、あなたのためのすべてのをダウンロードし、アプリケーションをビルドします。なAPKファイルは`ProjectName/ModuleName/build/outputs/apk`あり、`ModuleName-release.apk`とばれます。

しいプロジェクトをする

Android Studioをする

Android Studioをしてからきます。、あなたはのAndroidアプリをるがいました

> このガイドはAndroid Studio 2.2につていますが、のバージョンのプロセスはにじです。

プロジェクトをする

しいプロジェクトをするには、の2つのがあります。

- ようこそから[`Start a New Android Studio Project`クリックします。
-

にプロジェクトがいているは、 `File` → `New Project`ナビゲートし`File`。

に、いくつかのフィールドをしてアプリケーションをするがあります。

1. アプリケーション - このはユーザーにされます。

   `Hello World`。 で`AndroidManifest.xml`ファイルでいつでもできます。

2. ドメイン - これはプロジェクトのパッケージのです。

   `stackoverflow.com`。

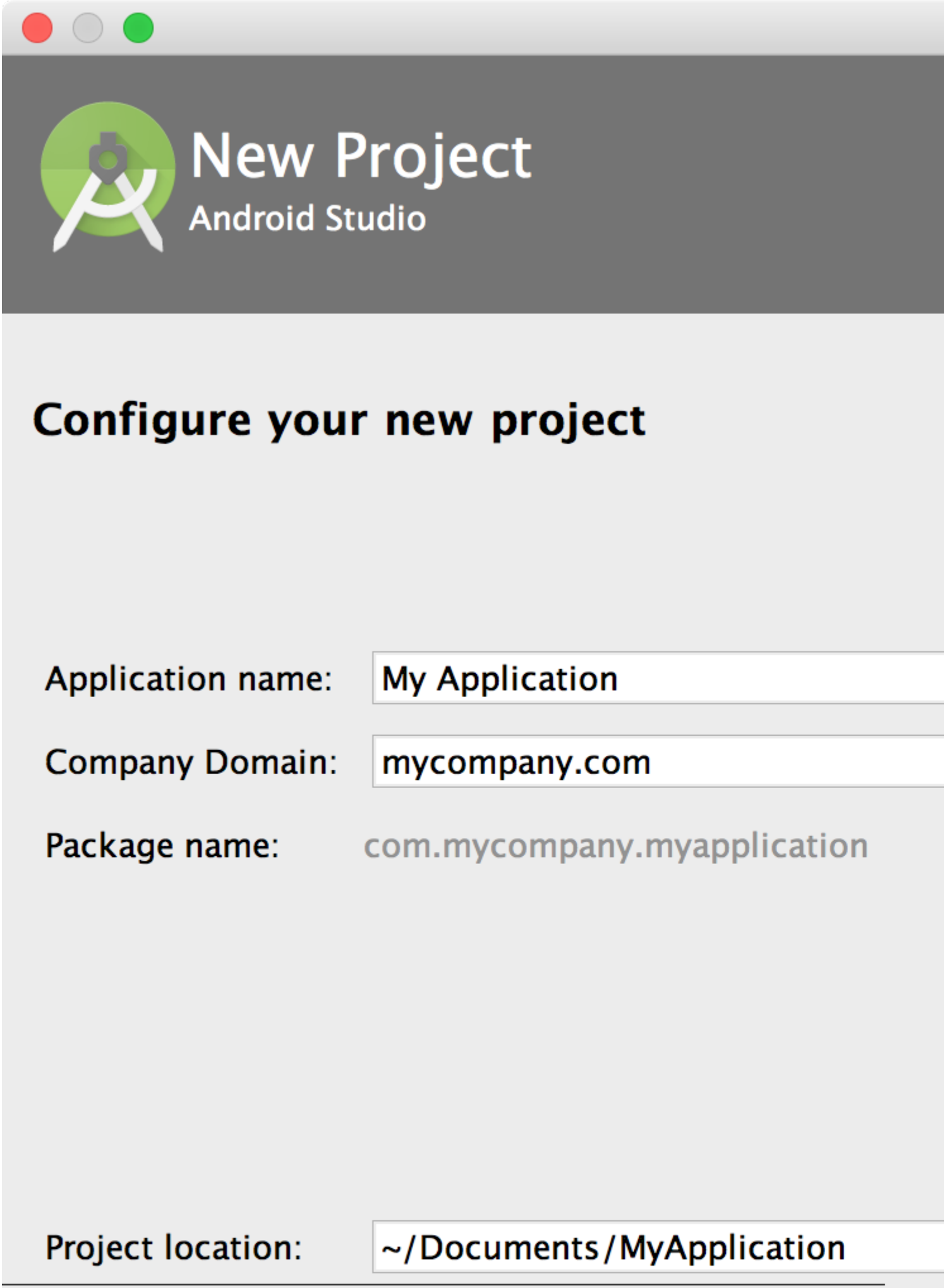
3. パッケージ `applicationId` - これはプロジェクトパッケージです。

   これはドメインネーム リバース*DNS* トップレベルドメインにうべきです。 ドメイン。 [セグメント。] アプリケーション。

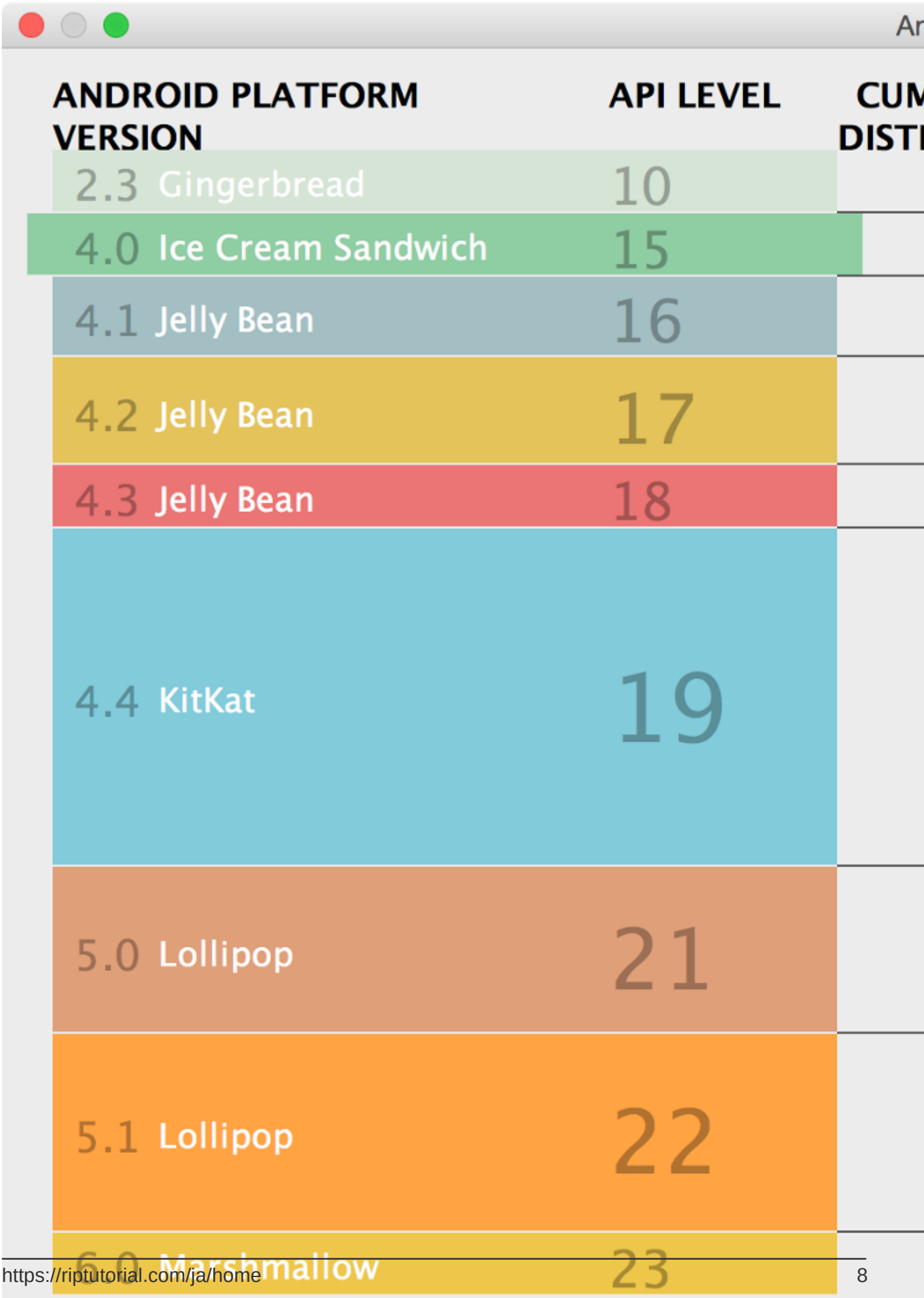
   `com.stackoverflow.android.helloworld`または`com.stackoverflow.helloworld`。 あなたのアプリケーション IDは、いつでもあなたのgradleファイルできることでできます 。

   アプリケーションをGoogle Playストアにしないをき、デフォルトの「com.example」はしないでください。パッケージはGoogle Playののアプリ**ID**になります。

4. プロジェクトの - プロジェクトがされるディレクトリです。


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

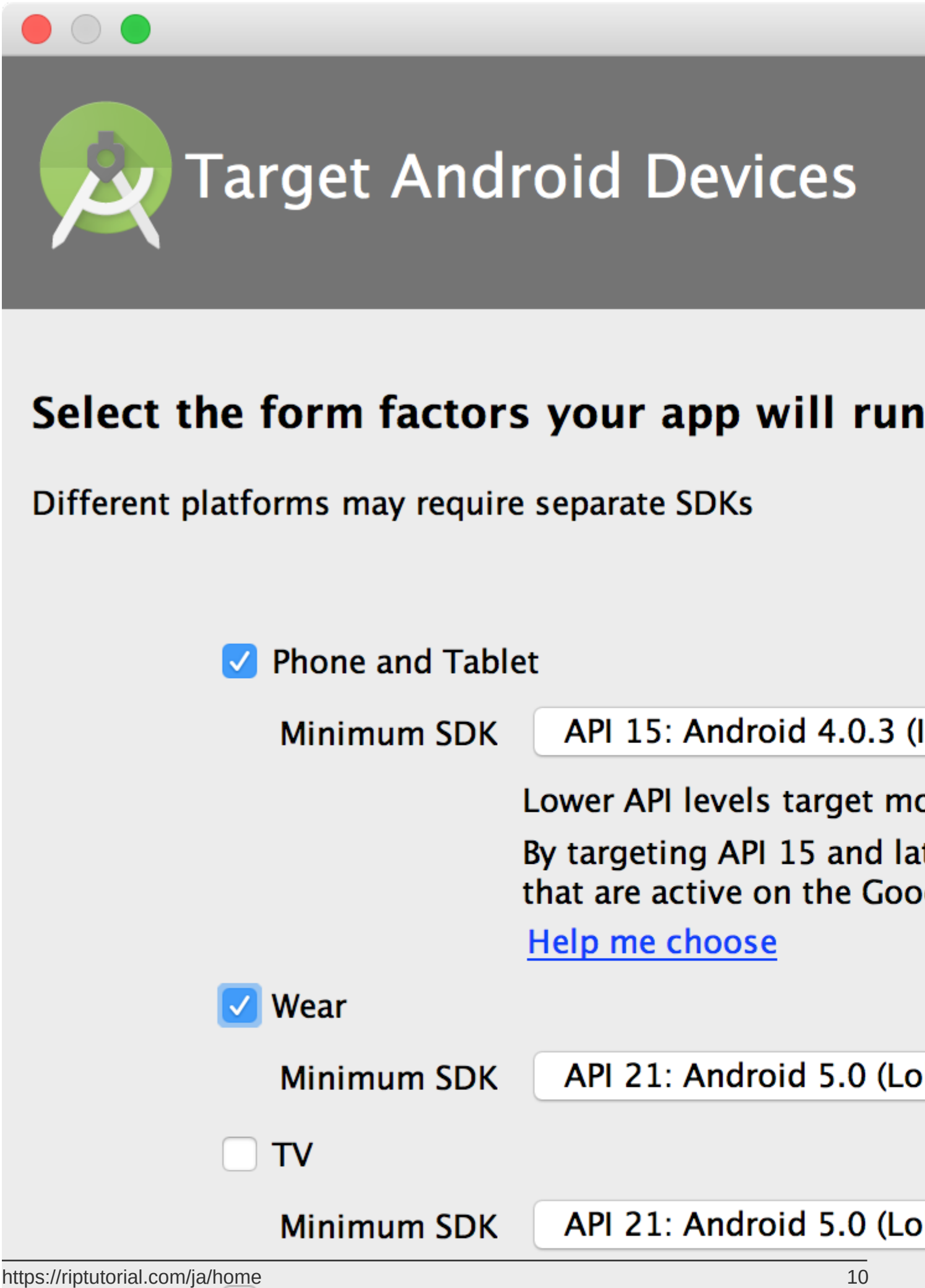
などのフォームファクタをできます。したフォームファクタは、プロジェクトのアプリモジュールになります。フォームファクタごとに、そのアプリのAPIレベルをすることもできます。をするには、[ヘルプを]をクリックします。


ANDROID PLATFORM VERSION
API LEVEL
2.3 Gingerbread
10
4.0 Ice Cream Sandwich
15
4.1 Jelly Bean
16
4.2 Jelly Bean
17
4.3 Jelly Bean
18
4.4 KitKat
19
5.0 Lollipop
21
5.1 Lollipop
22
23

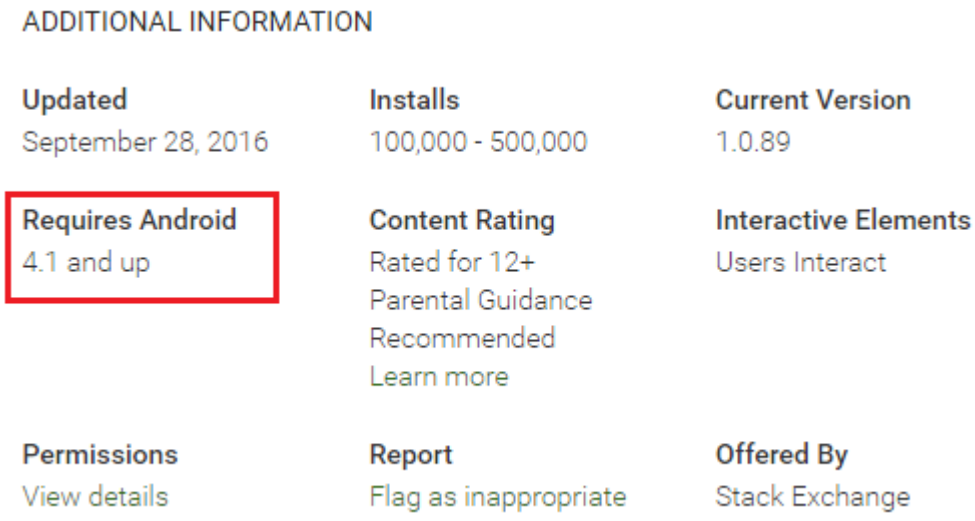
したヘルプをクリックするとされるの*Android*バージョンの

Android Platform Distributionウィンドウには、2にすように、Androidのバージョンをするモバイルデバイスのがされます。するAndroidバージョンでされたのリストをするには、APIレベルをクリックしてください。これにより、アプリになすべてのをえたのAPIレベルをできるので、できるだけくのにアクセスできます。に[**OK**]をクリックします。

に、アプリケーションがサポートするAndroid SDKのプラットフォームとバージョンをします。


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

するためにするの1つです。たとえば、 Stack Exchangeのアプリは Android 4.1をサポートしています。



Android Studioは、されたSDKがあれば、サポートされるデバイスのをおおよそえてくれます。

> APIレベルがいほど、よりくのデバイスをターゲットにできますが、できるはなくなります。

SDKをするには、 Dashboardsのをするがあります。このは 、Google Playストアをにしたデバイスにするバージョンをします。



Androidデベロッパーのウェブサイトのダッシュボード

アクティビティをする

に、アプリケーションのデフォルトのアクティビティをします。 Androidでは、 `Activity`はユーザーにされるのです。アプリケーションはのアクティビティをし、それらのをナビゲートできます。このでは、「 `Empty Activity`をし、「へ」をクリックします。

ここでは、にじて、アクティビティとレイアウトのをできます。いはすることです`Activity`アク

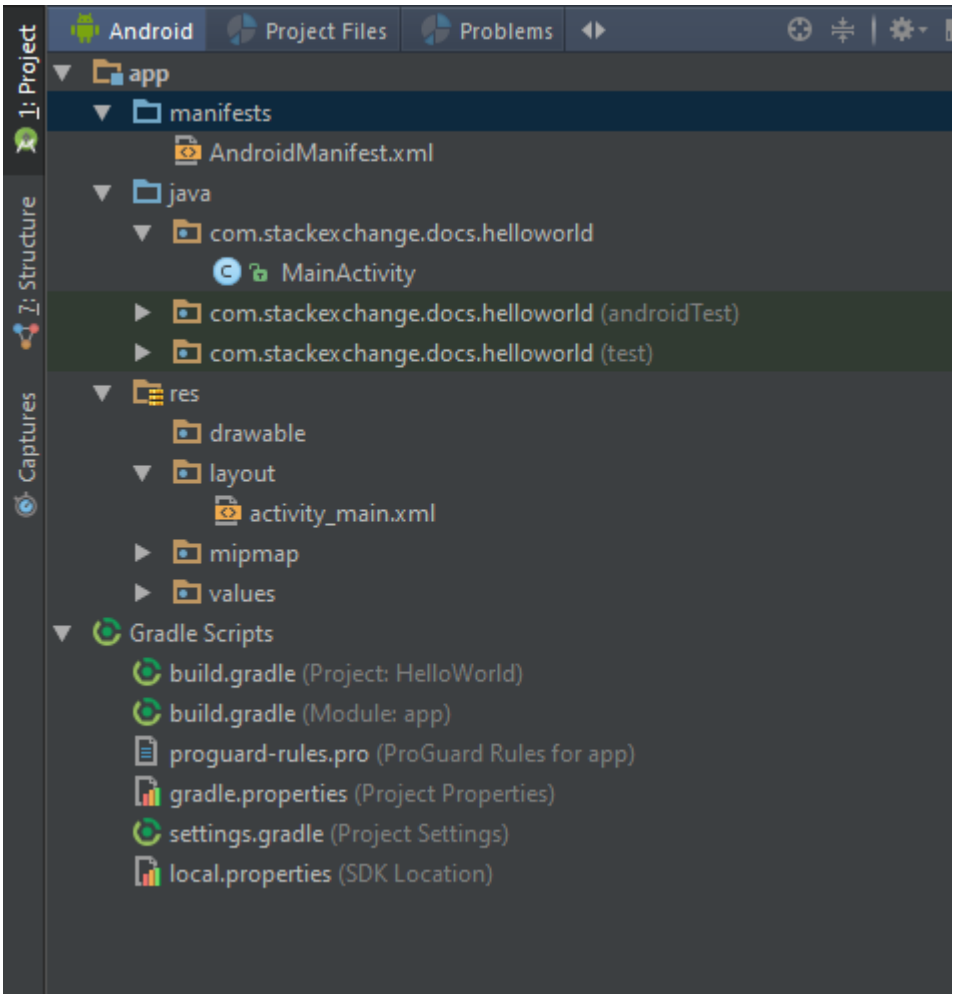
ティビティのとして、び`activity_`レイアウトのとして。これらをデフォルトのままにすると、Android Studioは`MainActivity`という`activity_main`と`activity_main`というレイアウトファイルをします。 `Finish`クリックします。

Android Studioはプロジェクトをしてしますが、システムによってはがかかることがあります。

プロジェクトの

Androidのみをするために、たちのためにされたファイルのいくつかをてみましょう。

Android Studioのペインには、Androidアプリケーションのがされます 。



まず、 `AndroidManifest.xml`をダブルクリックしてきましょう。 Androidマニフェストファイルには、Androidアプリケーションにするのがされています。これには、のだけでなく、いくつかのなコンポーネントがまれています。

アプリケーションがでされているにアクセスするがあるは、マニフェストの`<uses-permission>`でそのがであることをするがあります。に、アプリケーションがデバイスにインストールされると、インストーラは、アプリケーションのにしたをチェックし、によってはユーザにいわせることによって、されたをえるかどうかをします。また、アプリケーションは、のコンポーネントアクティビティ、サービス、ブロードキャストレシーバ、およびコンテンツプロバイダをアクセ

スでることもできます。 Androidでされたandroid.Manifest.permissionにリストされているをすることも、のアプリケーションによってされたをすることもできます。それはのものをすることもできます。

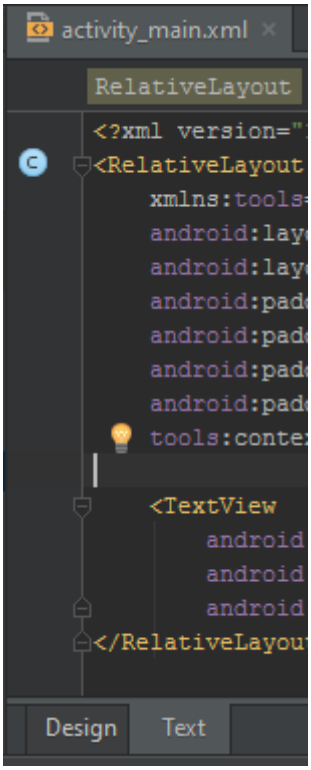
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.stackoverflow.helloworld">

    <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>
```

に、 `app/src/main/res/layout/`ある`activity_main.xml`をき`app/src/main/res/layout/`。 このファイルには、MainActivityのビジュアルコンポーネントのがまれています。ビジュアルデザイナーがされます。これにより、をドラッグしてしたレイアウトにドロップできます。

Android Studioのにある[テキスト]をクリックすると、XMLレイアウトデザイナーにりえることもできます。




```
<?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>
```

このレイアウトに`TextView`というウィジェットがされ、 `android:text`プロパティが「Hello World」にされてい`android:text`。これは、ユーザーがアプリケーションをするときにユーザーにされるテキストブロックです。

レイアウトとについてしくむことができます 。

に、 `MainActivity`てみましょう。これは`MainActivity`にされたJavaコードです。

```
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);
    }
}
```

Androidマニフェストでされているように、 `MainActivity`は、ユーザーが`HelloWorld`アプリケーションをするとデフォルトでします。

に、 `app/`ある`build.gradle`というのファイルをきます。
Android Studioは、ビルドシステム**Gradle**をして、Androidアプリケーションとライブラリをコンパイルしてビルドします。

```
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'
}
```

このファイルには、ビルドとアプリケーションのバージョンにするがまれています。また、このファイルをしてライブラリにをすることもできます。のところ、もしないでください。

ににできるのバージョンをすることをおめします。

- buildToolsVersion 26.0.0
- com.android.support:appcompat-v726.0.02017 7
- 11.0.420178

compileSdkVersion

`compileSdkVersion`は、あなたのアプリをコンパイルするためのAndroid SDKのバージョンを*Gradle*にえるです。しいAndroid SDKをするには、そのレベルでされたしいAPIをするがあります。

`compileSdkVersion`をしてものはされないことにしてください。 `compileSdkVersion`すると、しいコンパイラ/エラーがされることがありますが、 `compileSdkVersion`はAPKにはまれていません。コンパイルににされます。

したがって、にのSDKでコンパイルすることをくおめします。のコードにするしいコンパイルチェックのをすべてて、たにのAPIをけ、しいAPIをできるようにします。

minSdkVersion

`compileSdkVersion`ができるのAPIをしている、 `minSdkVersion`はアプリのです。 `minSdkVersion`は、Google Playストアがユーザーののうちどれにアプリをインストールできるかをするためにするの1つです。

また、になをたします。デフォルトでは、lintはプロジェクトにしてされ、 `minSdkVersion`のAPIをするとし、しないAPIをびそうとするののをします。しいバージョンのプラットフォームでのみAPIをするは、にシステムのバージョンをするのがなです。

targetSdkVersion

`targetSdkVersion`は、 `targetSdkVersion`がされないり、のをしないことによってAndroidがをするなです。これにより、のをするにしいAPIをすることができます。のSDKをターゲットにするためのは、すべてのアプリにとってです。されたすべてのをするはなく、テストせずに`targetSdkVersion`をにするはありません。

targetSDKVersionはなツールのであるAndroidのバージョンです。 targetSDKVersionが23の、アプリがAPI 23でされているでも、アプリはインスタンスのにアクセスをリクエストするはありません。 TargetSDKVersionは、したAndroidバージョンをえるAndroidバージョンがアプリをすることをげません 。

Gradleプラグインのはのとおりです。

- な
- アンドロイドとラッパーのためのGradleプラグインの
- build.gradleとDSLメソッドのの

アプリケーションの

では、HelloWorldアプリケーションをしましょう。 Androidデバイスのでする Androidスタジオの AVDマネージャをしてできますをするか、USBケーブルをしてAndroidデバイスをします。

Androidの

AndroidデバイスのAndroidスタジオからアプリケーションをするには、デバイスので`Developer Options`で`USB Debugging`をにするがあります。

`Settings > Developer options > USB debugging`

`Developer Options`がにされていないは、 `About Phone` にし、 `Build Number` 7タップします。これにより、 `Developer Options`にされます。

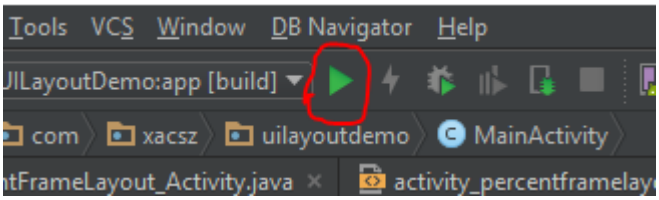
`Settings > About phone > Build number`

また、デバイスにあるバージョンをビルドするために`build.gradle`をするがあるかもしれません。

Android Studioからする

Android Studioのにあるツールバーのの「 `Run`ボタンをクリックします。されるウィンドウで、アプリケーションをするデバイスをしますにじてAndroidデバイスをするか、 するがあるはAVD

Androidデバイスをするを。[OK]をクリックします。



Android 4.4KitKatのデバイスでは、USBデバッグをするポップアップがされます。 OKをクリックしてけれます。

アプリケーションがAndroidデバイスまたはエミュレータにインストールされ、されます。

APKファイルの

リリースにアプリケーションをするときは、アプリケーションのリリースバージョンを、、およびテストします。はで、なコードクリーンアップとアプリケーションのにつコードがまれています。ビルドプロセスは、デバッグビルドプロセスとており、JDKおよびAndroid SDKツールをしてできます。テストタスクはなチェックとしてし、アプリケーションがのでどおりにすることをします。リリースのためのアプリケーションのがすると、みのAPKファイルがされます。このファイルは、ユーザーにしたり、Google Playなどのアプリケーションマーケットプレイスですることができます。

Androidスタジオ

のでは、Gradleがされているため、されたAPKファイルの`<Your Project Location>/app/build/outputs/apk/app-debug.apk`はのとおりです。 `<Your Project Location>/app/build/outputs/apk/app-debug.apk`

IntelliJ

StudioにりえるにIntelliJをしていて、IntelliJプロジェクトをインポートしているは、もされません。アウトプットのはのりです

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

これは々1.0のまわりでされる

Eclipse

Android Eclipseプロジェクトをインポートするは、これをわないでくださいプロジェクトjarsまたはLibrary Projectsにするとすぐに、これはせず、プロジェクトはにセットアップされません。がない、apkはEclipseでつけられるのとじにあります

```
bin/...
```

これは、もなAndroidツールのみをする、のHello Worldのです。

と

- Oracle JDK 1.7
- Android SDKツール コマンドラインツールのみ

このでは、Linuxをしています。のプラットフォームのをするがあるかもしれません。

Android SDKの

SDKリリースの

1. SDKマネージャをしてパッケージをインストールします。バンドルされたReadme.txtのにって、 `android update sdk --no-ui` をしないでください。なファイルを30 GBダウンロードします。わりに、インタラクティブなSDKマネージャー`android sdk`をして、パッケージのをしてください。

2. のJDKおよびSDKディレクトリをPATHにします。これはオプションですが、のでしています。

   - JDK / bin
   - SDK /プラットフォームツール
   - SDK /ツール
   - SDK / build-tools / LATEST *1*でインストールしたもの

3. Androidデバイスをします。インタラクティブなAVDマネージャ `android avd` をしてください。ちょっとをかけてアドバイスをすがあるかもしれません。 オンサイトのはずしもではありません。

   のデバイスをすることもできます

4. デバイスをします。

   ```
   emulator -avd DEVICE
   ```

5. デバイスのがロックされているようにえるは、スワイプしてロックをします。

   アプリをコーディングしているはしておいてください。

6. のディレクトリにします。

7. ソースファイルをする

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

コンテンツ

```
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. マニフェストをする

```
touch AndroidManifest.xml
```

コンテンツ

```
<?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. されたリソースのサブディレクトリをします。

```
mkdir res
```

はにしておいてください。

コードの

10. リソースのソースをします。ここでは、 **SDK**へのしいパスと、ビルドするためにインストールされた**API**「android-23」などをきえます。

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

リソースでさらにするは、はオプションです。 res /がまだの、のびしはもしません。

11. ソースコードをJavaバイトコード.java→.classにコンパイルする

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

12. バイトコードをJavaからAndroidにする.class→.dex

まずJill.class→.jayceをって

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

その、ジャック.jayce→.dex

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

Androidバイトコードは「Dalvikコード」とばれていたので、「dex」です。

あなたがきなら、ステップ11と12をジャックへの1のびしできえることができます。 Javaソース.java→.dexからコンパイルできます。しかし、 `javac`コンパイルすることにはがあります。よりよくられており、よりよくされ、よりくなツールです。

13. マニフェストをむリソースファイルをパッケージします。

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

その、なAPKファイルAndroidアプリケーションパッケージがされます。

14. `ApkBuilder`ツールをしてなAPKを`ApkBuilder`ます。

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

これはされています。 "このツールはされていません。は--helpをしてください。" `--help`が`ArrayIndexOutOfBoundsException`でした`ArrayIndexOutOfBoundsException` 、わりにをしません

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

これは、CLI `ApkBuilderMain` がJava API `ApkBuilder` をびすためにされることをしています。コマンドラインからそれをうがわかっているは、このをしてください。

15. APKのデータアライメントをする される

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

インストールと

16. Androidにアプリをインストールする

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

17. アプリをする

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

それはし、こんにちはうがあります。

それでです。それはなAndroidツールをってこんにちはとうのです。

リソースの

このセクションはオプションです。な "hello world"アプリケーションでは、リソースはありません。あなたのアプリケーションにもでないは、ステップ10をし、ステップ13からのres /ディレクトリへのをすることで、いくらかのビルドをすることができます。

それのは、リソースをするとそののなをします。

18. リソースファイルをする

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

コンテンツ

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

19. XMLマニフェストからリソースをします。これはなスタイルです。

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

20. Javaソースからじリソースをしてください。これはのです

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

21. アプリを、インストール、して、のをテストします1017。

    それはし、"このアプリは、こんにちはとわれています"とうがあります。

アプリのアンインストール

```
adb uninstall dom.domain
```

もしてください

- の - このをしたの
- - のコマンドをするビルドスクリプト

アプリケーションの

Androidアプリは Javaでかれています。 Android SDKツールは、コード、データ、およびリソースファイルをAPKAndroidパッケージにコンパイルします。に、1つのAPKファイルにはアプリのすべてのコンテンツがまれています。

それぞれのアプリはのマシンVMでし、アプリをのアプリからしてできます。 Androidシステムはのでします。アプリケーションは、をうためになコンポーネントだけにアクセスできます。しかし、アプリでLinuxユーザーIDをするなど、アプリがのアプリとデータをするや、アプリがSDカード、などのデバイスデータにアクセスするをリクエストすることができます。

アプリコンポーネント

アプリコンポ―ネントは、Androidアプリのビルディングブロックです。それぞれのコンポ―ネントは、なをたし、なライフサイクルコンポ―ネントがされ、されたとタイミングをつAndroidアプリでのをたします。 4のアプリコンポ―ネントがあります

1. アクティビティアクティビティは、ユ―ザ―インタ―フェイスUIをつのをします。 Androidアプリにのアクティビティがあるがあります。 えば、メ―ルアプリは、すべてのメ―ルをリストする1つのアクティビティ、メ―ルのをするアクティビティ、しいメ―ルをするアクティビティを1つずつつことができます。
2. サ―ビスサ―ビスはバックグラウンドでされ、されるをしたり、リモ―トプロセスのをしたりします。サ―ビスはUIをせず、バックグラウンドでのみユ―ザ―のでされます。 えば、ユ―ザ―がのAppにいるにサ―ビスがバックグラウンドでされたり、Androidデバイスとのユ―ザ―のやりとりをげることなくインタ―ネットからデ―タをダウンロ―ドするなど。
3. コンテンツプロバイダコンテンツプロバイダはアプリケ―ションデ―タをします。アプリケ―ションにデ―タをするには4つのがありますファイルにきんでファイルシステムにしたり、SQLiteデ―タベ―スにまたはしたり、ウェブにしたり、アプリケ―ションがアクセスできるのなにすることができます。コンテンツプロバイダ―をじて、のAppsはデ―タのやをうことができます。 えば、Androidシステムは、ユ―ザのをするコンテンツプロバイダをし、をつのアプリケ―ションがにいわせることができます。コンテンツプロバイダは、デ―タのをめるために、アプリケ―ションのデ―タをするためにもできます。
4. は、システムのにするえば、がえたこと、がっていることなどをらせる、またはAppsからするえば、のアプリケ―ションに、デバイスにダウンロ―ドされ、できるようになります。ブロ―ドキャストにはUIはありませんが、ステ―タスバ―にをしてユ―ザ―にすることができます。、は、アプリののコンポ―ネントへのゲ―トウェイとしてされ、にアクティビティやサ―ビスからされます。

Androidシステムのユニ―クなの1つは、どのアプリものアプリのコンポ―ネントをできることですたとえば、をかけたり、SMSをしたり、ウェブペ―ジをいたり、をしたいは、じタスクのためのしいアクティビティをするわりに、それをする。

システムがコンポ―ネントをすると、そのアプリケ―ションのプロセスがされますまだされていない、つまり、Androidシステムではいつでもアプリケ―ションごとに1つのフォアグラウンドプロセスしかできません。したがって、コンポ―ネントはそれがするAppのプロセスでされます。したがって、のシステムのアプリとはなり、Androidアプリには1つのエントリポイントがありません`main()`メソッドはありません。

システムはアプリケ―ションを々のプロセスでするため、1つのアプリでのアプリのコンポ―ネントをアクティブすることはできませんが、Androidシステムではです。したがって、のアプリケ―ションのコンポ―ネントをするには、1つのアプリケ―ションが、そのコンポ―ネントをするをするメッセ―ジをシステムにしなければなりません。システムは、そのコンポ―ネントをします。

コンテキスト

クラス`android.content.Context`インスタンスは、アプリケ―ションをするAndroidシステムへのをします。コンテキストのインスタンスは、プロジェクトのリソ―スとアプリケ―ションのにする

グローバルなにアクセスするためにです。

なをげましょうあなたはホテルにいるとえて、かべたいとっています。あなたはルームサービスにをして、をってるか、をきれいにするようします。ではこのホテルをAndroidアプリとしてえています。をアクティビティとんでいます。ルームサービスのは、ルームサービス、などのホテルリソースへのアクセスをするコンテキストです。

さらにのとして、あなたはテーブルにっているレストランにいて、テーブルにはアテンダントがいます。アテンダントがあなたのをし、あなたのべがあなたのテーブルにまれます。このでも、レストランはAndroid App、テーブルまたははAppコンポーネント、アイテムはAppリソース、アテンダントはコンテキストなので、アイテムなどのリソースにアクセスするをします。

のコンポーネントのいずれかをにするには、コンテキストのインスタンスがです。だけでなく、ほぼすべてのシステムリソースビューをしたUIの、システムサービスのインスタンスの、しいアクティビティやサービスの - はすべてコンテキストをとします。

よりなはここにかれています 。

AVDAndroidデバイスの

TL; DRには、のデバイスをシミュレートし、のデバイスなしでアプリをテストすることができます。

Androidデベロッパーけドキュメントによると、

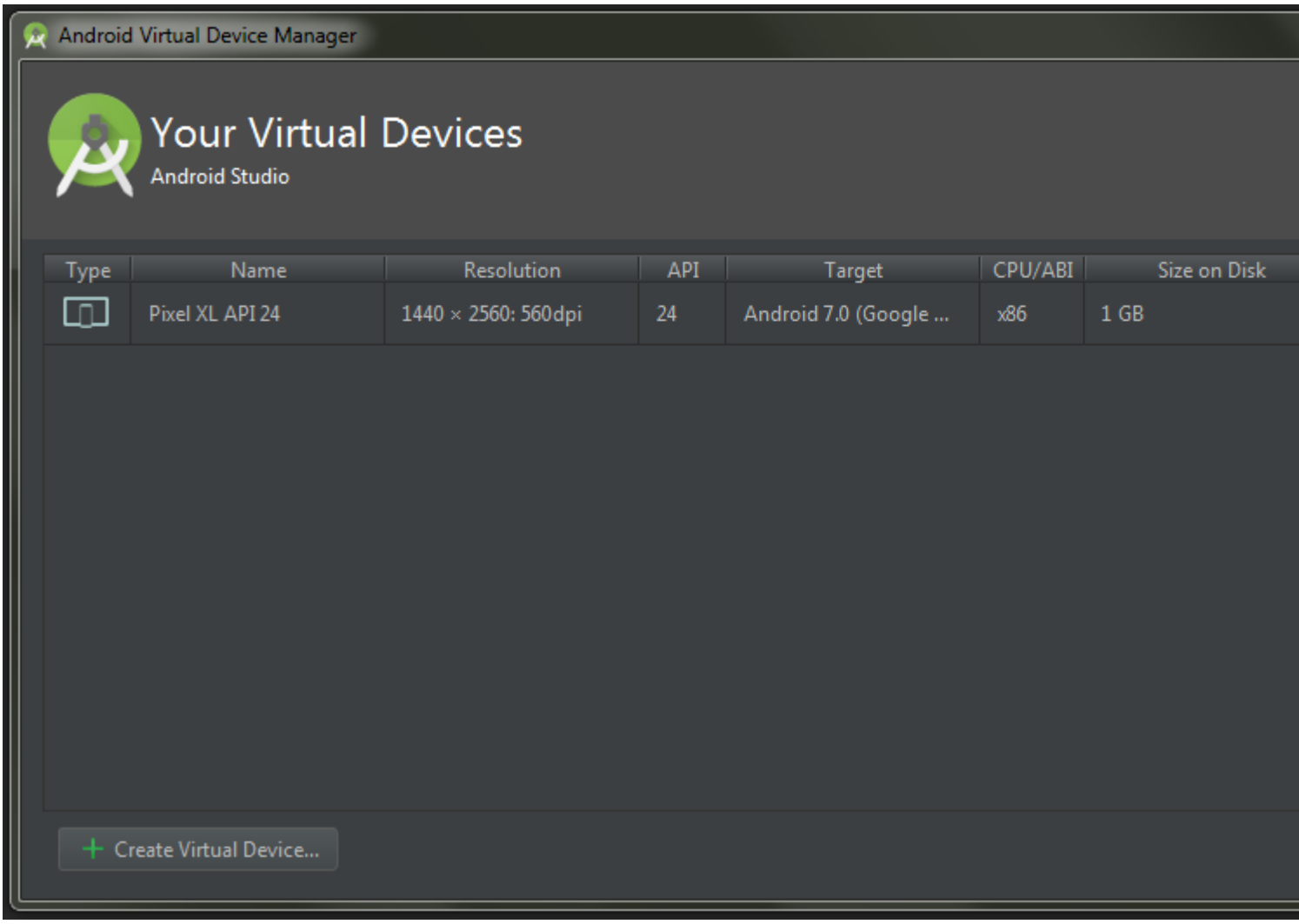
> ***Android***デバイス***AVD***では、Androidエミュレータでシミュレートするandroid Phone、Tablet、Android Wear、またはAndroid TVデバイスのをできます。 AVDマネージャをすると、AVDをにおよびできます。

AVDをするには、のをします。

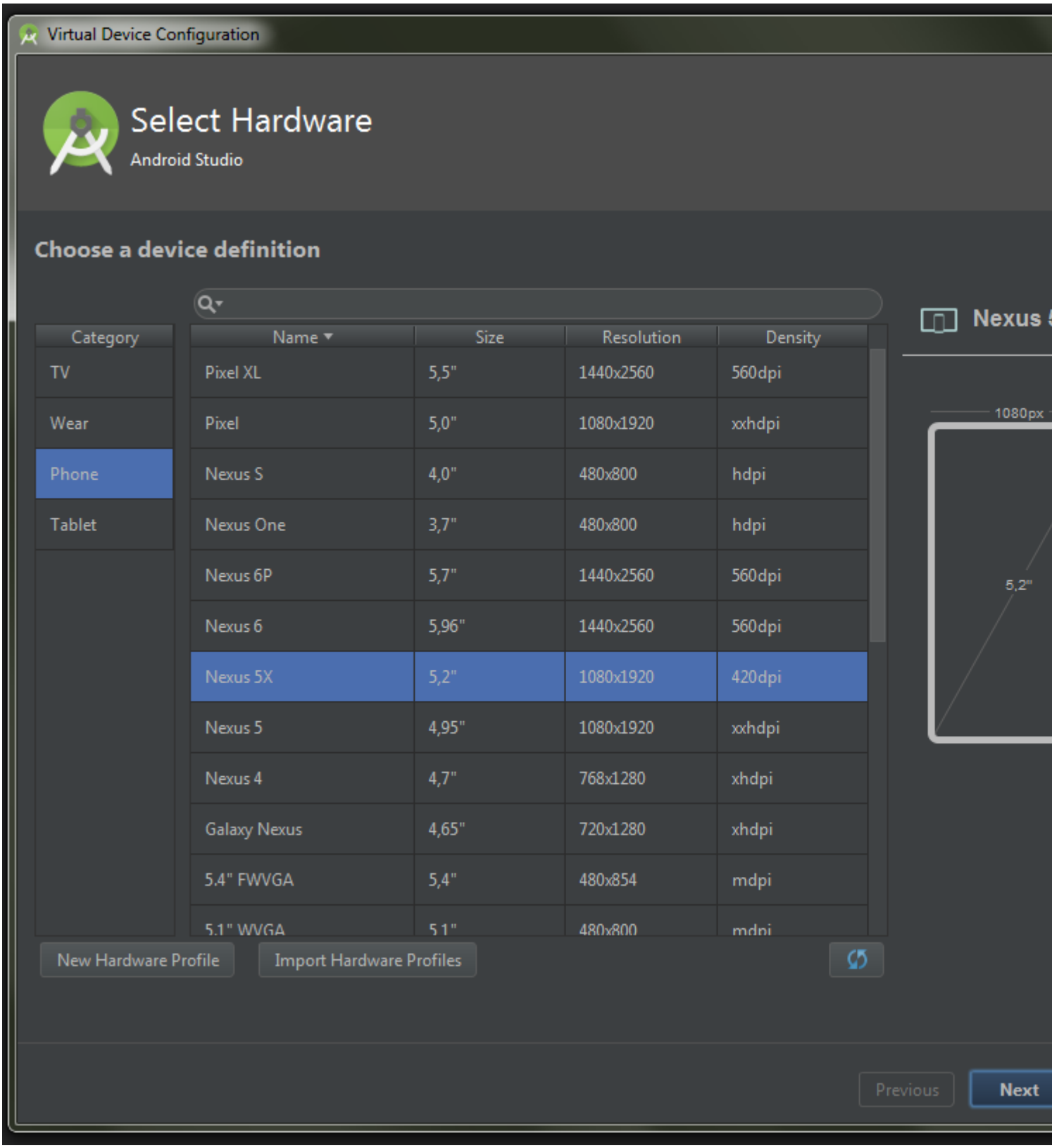
1.このボタンをクリックすると、AVD Managerがします。



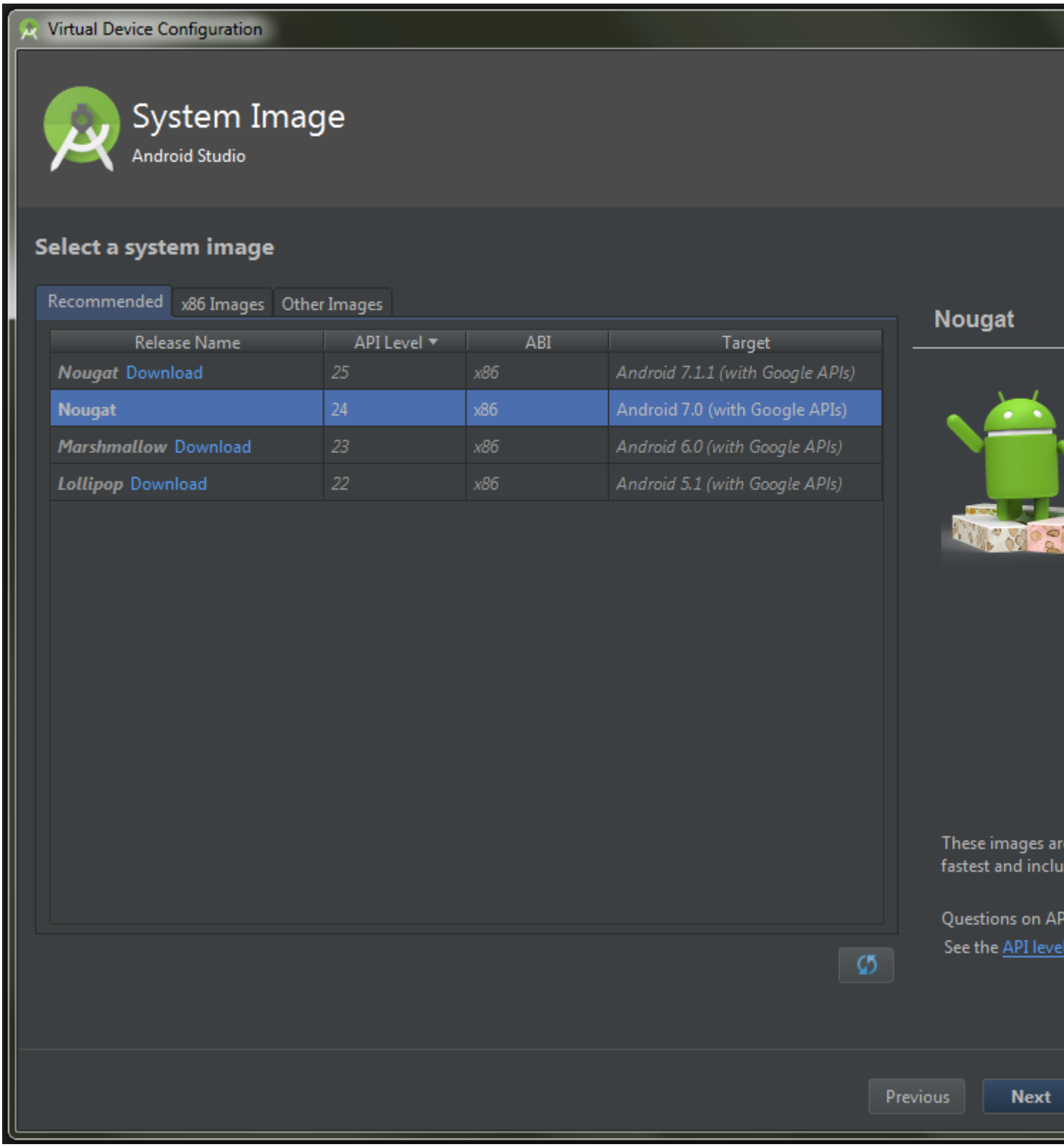
2.のようなダイアログがされます。



3. [`+ Create Virtual Device...`]ボタンをクリックします。これにより、Virtual Device Configuration Dialogがされます。



4.なデバイスをし、[`Next`]をクリックします。



5.エミュレータのAndroidバージョンをするがあります。また、クリックしてにそれをダウンロードするがあるかもしれません`Download`。バージョンをしたら、[`Next`]をクリックします。



6.ここでは、エミュレーターの、の、およびフレームのりにフレームをするかどうかをします。これらをすべてしたら、[`Finish`]をクリックします。

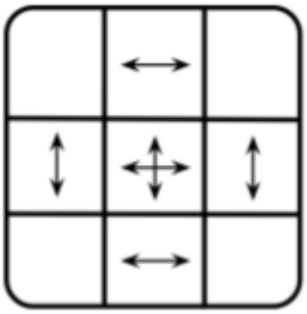
7.これであなたのアプリをするためのしいAVDがされました。

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk
	Nexus 5X API 24	1080 × 1920: 420dpi	24	Android 7.0 (Google ...	x86	650 MB

オンラインでAndroidをいめるをむ https://riptutorial.com/ja/android/topic/85/androidをいめる

2: 9パッチ

9パッチファイルはにフォーマットされたファイルであるため、のどのができるかどうかはAndroidにされます。あなたのイメージを3x3グリッドにします。コーナーはスケーリングされていないままで、サイドはにスケーリングされ、センターはのディメンションでスケーリングされます。



Nine Patch9-Patchイメージは、イメージに1つのピクセルのをつビットマップです。のコーナーにある4つのピクセルをする。このは、ビットマップのメタデータをします。はのでされます。

9パッチは`.9.png`されます。

のは、にびるをします。のは、にびるをします。

のは、にパディングをします。のは、のパディングをします。

パディングボーダーは、テキストをするをするためにされます。

これらのファイルのをにするれたツールがGoogleによってされています。

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

Examples

ない

しくストレッチするためのは、とにあります。

のはのストレッチをし、のはのストレッチをします。

このでは、トーストにしたいコーナーをします。



ののに、 のにあるのは、すべてののスペースをめるためにしていきます。

このは、にすように、サイズのすべてのみわせにびます。



スピナー

`Spinner`は、Nine Patchをしてのスタイルにってスキンします。

として、このNine Patchをしてください。



あなたがることができるように、それは3つのにさなストレッチのをマークしています。

のにはアイコンのマークがっています。これは、アイコンにするまで、Drawableのなが`Spinner`ビューをめるようにすることをしています。

のには、マークされたアイコンのとになセグメントがマークされています。これは、とのが`Spinner`ビューのサイズにすることをします。これにより、アイコンがのにかれます。

Nine Patchメタデータなしのイメージの



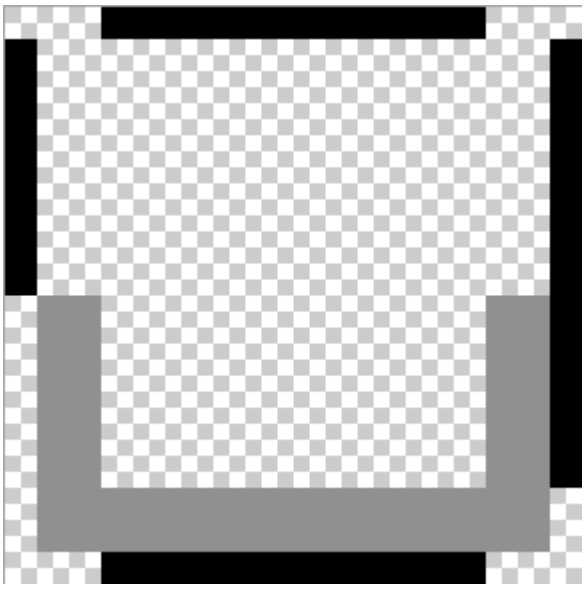
Nine Patchメタデータでをする



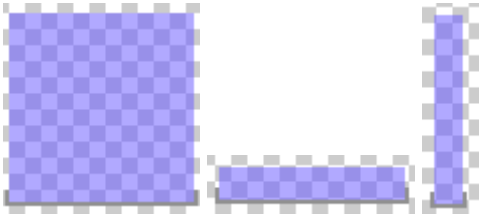
オプションのパディングライン

9つのパッチは、のパディングラインののをにする。パディングラインは、とのラインです。

ビューが9パッチをそのとしてする、パディングラインをしてビューのコンテンツのスペースをしますたとえば、 `EditText`テキスト。パディングがされていないは、わりにとのがされます。



きばしたのコンテンツは、のようになります。



オンラインで9パッチをむ https://riptutorial.com/ja/android/topic/461/9パッチ

3: ACRA

- androidname = "。ACRAHandler"
- ACRA.initthis、config;
- パブリッククラスACRAHandlerはアプリケーション{

パラメーター

パラメータ	
@ReportCrashes	レポートする、カスタムコンテンツなどのACRAをします。
formUri	クラッシュをするファイルへのパス

- ACRAはGoogleフォームをサポートしていないため、バックエンドがです https://github.com/ACRA/acra/wiki/Backends

Examples

ACRAHandler

レポートをするアプリケーションクラス

```
@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);


    }

}
```

マニフェストの

```
<?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>
```

インストール

Maven

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

け

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

オンラインでACRAをむ https://riptutorial.com/ja/android/topic/1324/acra

4: ADBAndroid Debug Bridge

き

ADBAndroid Debug Bridgeは、エミュレータインスタンスまたはされたAndroidデバイスとするためにされるコマンドラインツールです。

ADBの

このトピックのはadbシェルにされました

adbシェルにされたサンプルのリスト

- API 23+ののとりし
- ADBをしてテキスト、キー、タッチイベントをAndroidデバイスに
- パッケージをする
- ディスプレイの
- オープンオプション
- adbで/をする
- chmodコマンドをしてファイルをする
- 「ブート」ブロードキャストの
- アプリケーションデータをする
- /セカンダリストレージコンテンツの
- http://stackoverflow.com/documentation/android/9408/adb-shell/29140/adb-shell
- Androidデバイスのプロセスをする

Examples

されたデバイスのなリストをする

adbにされているすべてのデバイスのなリストをするには、にのコマンドをします。

```
adb devices -l
```

```
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
```

- のはデバイスのシリアルです。 `emulator-` - でまる、このデバイスはエミュレータです。
- `usb:` USBサブシステムのデバイスのパス。
- `product:`デバイスのコード。これはメーカーのもので、のArchosデバイス`A50PL`にられるよう

に、にすることができます。

- `model`:デバイスモデル。 `product` 、にすることができます。
- `device`:デバイスコード。これはメーカーのものでもあり、でもかまいません。

デバイスをみむ

にのコマンドをきます

```
adb shell getprop
```

なすべてのがキーとのペアのでされます。

のキーのをコマンドにすることによって、のをみることができます。えば

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

ここでは、あなたががられるいくつかのいがあります

- `ro.product.model` デバイスのモデルNexus 6P
- `ro.build.version.sdk` デバイスのAPIレベル23
- `ro.product.brand` デバイスのブランドSamsungなど

な

```
[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]
```

WiFiでADBをデバイスにする

のADBには、デバイスへのUSBがまれます。
にじて、TCP / IPモードにりえて、WiFiでADBにすることもできます。

ルーツのないデバイス

1. じネットワークをする
    - おいのデバイスとコンピュータがじネットワークにあることをしてください。
2. デバイスを**USB**ケーブルでホストコンピュータにします。
3. ネットワークでデバイスに**adb**をする

    デバイスがUSBで`adb`されているは、のコマンドをしてポートのTCP / IPをちけますデフォルトは5555。

    - `adb tcpip <port>`し`adb tcpip <port>` TCP / IPモードにりえます。
    - ターゲットデバイスからUSBケーブルをします。
    - `adb connect <ip address>:<port>`としますポートはオプションで、デフォルトは5555です。

    えば

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

    デバイスのIPがわからないは、のをできます。

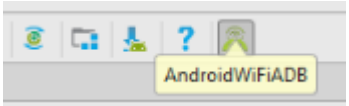
    - おいのデバイスのWiFiでIPをしてください。
    - ADBをしてIPをするUSB
        1. USBでデバイスをコンピュータにする
        2. コマンドラインで、 `adb shell ifconfig`をし、デバイスのIPアドレスをコピーします

    USBでデバッグにすには、のコマンドをします。

    ```
    adb usb
    ```

    また、Android Studioにプラグインをインストールすることで、WiFiでADBにすることもで

きます。これをうには、 *[]> [プラグインとリポジトリの]*にし、 *ADB WiFi*をしてインストールし、Android Studioをきます。のにすように、ツールバーにしいアイコンがされます。デバイスをUSBでホストコンピュータにし、この*AndroidWiFiADB*アイコンをクリックします。デバイスがされているかどうかをすメッセージがされます。されたら、USBのプラグをくことができます。



ルーツきデバイス

ルートされているデバイスのには 、にPlayストアのADB WiFi Appをしてこれをにすることができます。また、のデバイスにCyanogenMod ROMをしているデバイスでは、このオプションは、[ ]の[オプション]にあります。これをにすると、 `adb connect <ip address>:<port>`するだけで、 `adb` にするためになIPアドレスとポートがえられます。

ルーツをつデバイスがあるのに**USB**ケーブルにアクセスできない

このプロセスはのえでしくされています http : //stackoverflow.com/questions/2604727/how-can-i-connect-to-android-with-adb-over-tcp/3623727#3623727もなコマンドにします。

デバイスのをき、のようにします。

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

えば

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

あなたのコンピュータで

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

えば

```
adb connect 192.168.1.2:5555
```

オフにするには

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

タイムアウトをける

デフォルトでは、 adbは5000ミリにタイムアウトします。これは、WiFiやなAPKなど、のケースでするがあります。

Gradleのをにすると、このトリックをうことができます

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

デバイスからにファイルをプルプッシュする

のコマンドをして、デバイスからファイルをダウンロードすることができます。

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

えば

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

また、コンピュータからデバイスにファイルをプッシュアップロードすることもできます

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

えば

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

デバイスからデータベースをする

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

デバイスをする

のコマンドをすると、デバイスをできます。

```
adb reboot
```

ブートローダをするには、のコマンドをします。

```
adb reboot bootloader
```

リカバリモードでする

```
adb reboot recovery
```

デバイスがにシャットダウンされないことにしてください。

Wi-Fiをオン/オフにする

オンにする

```
adb shell svc wifi enable
```

す

```
adb shell svc wifi disable
```

なデバイスをする

コマンド

```
 adb devices
```

の

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

の - デバイスのシリアル

2のステータス

Androidのドキュメント

IPでデバイスをする

Androidのでこれらのコマンドをします

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

その、 **CMD**と**ADB**をして、のコマンドをしてできます

```
adb connect 192.168.0.101.5555
```

あなたはそれをにして、ADBをUSBでリッスンすることができます

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

コンピュータから、すでにUSBアクセスがあるルートはありません

すでにUSBをおちのは、Wi-Fiをするようにりえることもさらにです。デバイスがUSBでされているコンピュータのコマンドラインから、コマンドをします

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

*192.168.0.101*をデバイス*IP*にきえる

adbの/

ADBをする

```
adb kill-server
```

ADBをする

```
adb start-server
```

ログキャストをる

`logcat`をadbコマンドとしてすることも、エミュレータまたはされたデバイスのシェルプロンプトですることもできます。 `adb`をしてログをするには、SDKプラットフォームのtools-tools /ディレクトリにし、のコマンドをします。

```
$ adb logcat
```

あるいは、デバイスへのシェルをし、にすることができます。

```
$ adb shell
$ logcat
```

1つのなコマンドはのとおりです。

```
adb logcat -v threadtime
```

これにより、、びし、、タグ、およびメッセージをするスレッドのPIDとTIDがいメッセージでされます。

フィルタリング

Logcatログはログレベルとばれます。

V - 、 **D** - デバッグ、 **I** - 、 **W** - 、 **E** - エラー、 **F** - 、 **S** - サイレント

logcatをログレベルでフィルタリングすることもできます。たとえば、デバッグレベルのみをするはのようにします。

```
adb logcat *:D
```

Logcatはパッケージでフィルタリングできますが、もちろんログレベルフィルタとみわせることもできます

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

grepをしてログをフィルタリングすることもできますlogcatのはここでフィルタリングします 。

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

Windowsでは、findstrをしてfilterをできます。たとえば、のようになります。

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

ログバッファ[main | events | radio]をするには、 `-b`オプションを`logcat`して`logcat`をします。

```
adb logcat -b radio
```

をファイルにする

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

ファイルをながらをする

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

ログのクリーニング

```
adb logcat -c
```

マルチデバイスののデバイスへのADBコマンド

1.デバイスをシリアルでターゲティングする

`-s`オプションのにデバイスをして、 `adb`コマンドをするデバイスをします。 `-s`オプションは、コマンドのに 1にするがあります。

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

```
adb devices

List of devices attached
emulator-5554       device
02157df2d1faeb33    device

adb -s emulator-5554 shell
```

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. 1つのデバイスタイプのみがされている、デバイスをターゲットにする

のエミュレータのみを-eオプションでできます

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

または、されているのUSBデバイスを-d

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

デバイスのディスプレイからスクリーンショットとビデオをするキットキットのみ

スクリーンショットオプション1なadb

`shell` adbコマンドをすると、デバイスのみみシェルをしてコマンドをできます。 `screencap`シェルコマンドは、デバイスにされているコンテンツをキャプチャし、 `/sdcard/screen.png` / `/sdcard/screen.png`のイメージファイルにします。

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

に、pullコマンドをして 、デバイスからコンピュータののディレクトリにファイルをダウンロードできます。

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

スクリーンショットオプション2

の1をします。

マシュマロとそれ

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

Nougat

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

`-p`フラグは、 `screencap`コマンドのをstdoutにリダイレクトします。これがパイプされたPerlは、マシュマロとそれのいくつかのをします。ストリームはのディレクトリの`screen.png`というのファイルにきまれます。については、 このとこのをしてください。

ビデオ

これはKitKatとADBのみでのみします。これはKitkatではしません。デバイスののをするには、のコマンドをします。

`adb shell screenrecord /sdcard/example.mp4`このコマンドは、デフォルトをしてデバイスののをし、デバイスの`/sdcard/example.mp4`ファイルのファイルにのビデオをします。

がしたら、コマンドプロンプトウィンドウでCtrl + CLinuxのはzをして、をします。したにファイルがあります。のは、おいのコンピュータではなく、デバイスのストレージにされます。

デフォルトのでは、デバイスのをし、4Mbpsのビットレートでビデオをエンコードし、を180にします。できるコマンドラインオプションのについては、のコマンドをしてください。

`adb shell screenrecord -help` 、これはデバイスをすることなくします。おにてれば。

アプリケーションデータのクリア

`adb`をしてのアプリのユーザーデータをすることができます。

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

これは、のをし、アプリをし、データボタンをすのとじです。

- `pm`はデバイスのパッケージマネージャをびします
- `clear`は、パッケージにするすべてのデータをします。

をする

BroadcastReceiverにadb BroadcastReceiverをすることはです。

このでは、アクションcom.test.app.ACTIONとstring extraをバンドル'foo'='bar'ブロードキャストをしてい'foo'='bar'。

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

だけでなく、のサポートされているもバンドルできます。

--ez - boolean
--ei -
--el - long
--ef - フロート
--eu - uri
--eia - int '、'でられた
--ela - い '、'でられた
--efa - '、'でられた
--esa - '、'でられています

インテントをのパッケージ/クラスにするには、 -nまたは-pパラメータをできます。
パッケージにする

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

のコンポーネント com.test.app package SomeReceiverクラスへの

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

な

- 「ブート」ブロードキャストをする
- adbコマンドでをしたに「された」ブロードキャストをする

アプリケーションのインストールと

APKファイルをインストールするには、のコマンドをします。

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

アプリがしていてインストールしたい

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

アプリケーションをアンインストールするには、そのパッケージをするがあります

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

されたパッケージまたはアプリののアクティビティでアプリをするには、のコマンドをします。

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

たとえば、Wazeをするには

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

バックアップ

`adb backup`コマンドをしてデバイスをバックアップすることができます。

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

`-f <filename>` のディレクトリに*backup.ab*をし ます。

`-apk|noapk`バックアップを/にするデフォルト *-noapk*

`-obb|noobb`ファイルのバックアップをまたはにするデフォルト *-noobb*

`-shared|noshared`バックアップされていないバックアップデバイスのストレージ/ SDカードのデフォルト *-noshared*

`-all`バックアップインストールされているすべてのapllications

`-system|nosystem`システムアプリケーションのデフォルトがまれています。

`<packages>`バックアップするパッケージのリストcom.example.android.myapp- `-all`がされているは

すべてをむなデバイスバックアップのは、

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

フルバックアップをするとがかかることがあります。

バックアップをするには、

```
adb restore backup.ab
```

LinuxシステムにADBをインストールする

あなたのディストリビューションのリポジトリをってでLinuxシステムにAndroid Debugging BridgeADBをインストールする。

aptをしてUbuntu / Debianシステムにインストールする

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

yumでFedora / CentOSシステムにインストールする

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

portageをってGentooシステムにインストールする

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

zypperをってopenSUSEシステムにインストールする

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

pacmanをってArchシステムにインストールする

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

Android 6.0のユーザーからをとするすべてのアクセスをする

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

アプリのデータデータ/データ/ をデバイスに

まず、アプリが`AndroidManifest.xml`にバックアップできることをします。つまり、`android:allowBackup`は`false`はありません。

バックアップコマンド

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

ddコマンドでtarをする

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

ターールをする

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

その、されたコンテンツをすることができます。

アクティビティスタックをする

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

`watch` unixコマンドとにするとにです

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

アプリのキャッシュファイルをおよびプルする

このコマンドをして、のデバッグなapkのファイルをリストすることができます。

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

そして、このスクリプトをキャッシュからきすために、これはにsdcardにコンテンツをコピーし、にそれをりしてします

```
#!/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'"
```

に、のようにファイルをキャッシュからりすことができます

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

ADBをしてデータベースファイルをする

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

オンラインでADBAndroid Debug Bridgeをむ https://riptutorial.com/ja/android/topic/1051/adb-android-debug-bridge-

5: adbシェル

き

`adb shell`はターゲットデバイスまたはエミュレータでLinux `adb shell`きます。これは、 `adb`でAndroidデバイスをするもでなです。

このトピックはADBAndroid Debug Bridgeからされていました。のにし、そのくに`adb shell`コマンドがまれていました。

- adbシェル[-eエスケープ] [-n] [-Tt] [-x] [コマンド]

パラメータ

パラメータ	
-e	エスケープ、または「なし」をします。デフォルト "
-n	スタンダードからない
-T	PTYりてをにする
-t	PTYりて
-バツ	リモートコードとstdout / stderrをにする

Examples

ADBをしてテキスト、キー、タッチイベントをAndroidデバイスに

フォーカスがあるビューにテキストをするには、のコマンドをしますテキストをサポートしている

6.0

SDK 23でテキストをする+

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

に`adb`でデバイスにされている

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

6.0

SDK 23のにテキストをする

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

スペ―スはとしてけれられません。sできえてください。

イベントをする

ハ―ドウエアキ―をすことをシミュレ―トするには

```
adb shell input keyevent 26
```

あるいは

```
adb shell input keyevent POWER
```

ハ―ドウエアキ―をっていなくても、キ―keyeventをしてのをできます

```
adb shell input keyevent CAMERA
```

タッチイベントをとしてする

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

スワイプイベントをとしてする

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

* DURATIONはオプションですデフォルト= 300ms。 ソース

のオプションでポインタのをにすることでXとYのポイントをします。

ADBサンプルシェルスクリプト

> Ubuntuでスクリプトをするには、script.shをして、そのファイルをクリックして、みきをし、プログラムとしてファイルにチェックをれます。
>
> タ―ミナルエミュレ―タをき、コマンド./script.shをします。

Script.sh

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

イベントのなについては

- いくつかのいイベントのリストADB Shell Input Events
- ドキュメント https://developer.android.com/reference/android/view/KeyEvent.html#KEYCODE_POWER

パッケージをする

パッケージに<FILTER>のテキストがまれているものだけを、オプションですべてのパッケージをします。

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

All <FILTER>

adb shell pm list packages
```

するファイルをするには`-f`をします。

`-i`パッケージのインストーラをしてください。

`-u`には、アンインストールされたパッケージもまれます。

`-u`アンインストールされたパッケージもめます。

フィルタリングする

なパッケージのは`-d`。

なパッケージのは`-e`です。

システムパッケージのは`-s`です。

`-3`をします。

`--user <USER_ID>` するのユーザー・スペース。

API 23+ののとりし

なアクセスのまたはりしにつ1つのライナー。

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

  ```
  adb shell pm revoke <sample.package.id> android.permission.<PERMISSION_NAME>
  ```
- インストールににすべてのアクセスをする**-g**
  ```

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

アプリケーションデータをする

このコマンドは、するすべてのアプリケーションデータをします。

- バージョンコード
- バージョン
- されたアクセスAndroid API 23
- ..

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

ディスプレイの

4.4

Android 4.4APIレベル19をするのディスプレイの

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

Ctrl + Cをしてをする

デバイスからファイルをダウンロードする

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

> Ctrl-Cをしてをします。そうしないと、はに3に`--time-limit`。タイムリミットは`--time-limit`され`--time-limit`。

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

サイズを1280x720にします。デフォルトは、デバイスのネイティブディスプレイサポートされている、1280x720でないです。のをるには、おいのデバイスのAVCAdvanced Video Codingエンコーダでサポートされているサイズをしてください。

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

ビデオのビデオビットレートをメガビット/でします。デフォルトは4Mbpsです。ビデオをさせるためにビットレートをげることができますが、その、ムービーファイルがきくなります。のでは、ビットレートを5Mbpsにしています。

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

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

をでします。デフォルトとは1803です。

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

を90します。このはです。

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

コマンドラインにログをします。このオプションをしないと、ユーティリティはにをしません。

これはのデバイスではしないがあります。

4.4

スクリーンレコーディングコマンドは、4.4よりのアンドロイドバージョンとがありません

screenrecordコマンドは、Android 4.4APIレベル19をするデバイスのをするシェルユーティリティです。このユーティリティは、アクティビティをMPEG-4ファイルにします。

chmodコマンドをしてファイルをする

は、ファイル***prmissions***をするためには、おいのデバイスがいするがあること、 *su*バイナリは、のデバイスがしていません

コンベンション

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

ユーザー、グループ、ワールドのセクションからされたアクセス。

たとえば、すべてのがみり、きみ、およびにするためにファイルをするは、これがあなたのコマンドになります。

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

または

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

あなたがそれへのアクセスをしようとするならば。

1 - ユーザーをします。 **2** - グループをします。 **3** - そののアクセスをします。

アクセス

```
--- :   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)
```

adbで/をする

6.0

デフォルトのSETは`MMDDhhmm[[CC]YY][.ss]` それぞれ2です

たとえば、のをせずに7171010をするには、のようにします。

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

ヒント1のはちにされず、システムクロックがのにむとながわれます。
`TIME_SET`したブロードキャストをあなたのびしにけることによって、をすることができます

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

ヒント2 Androidのをローカルマシンとする

Linux

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

WindowsPowerShell

```
$currentDate = Get-Date -Format "MMddHHmmyyyy.ss" # Android's preferred format
adb shell "date $currentDate"
```

のヒントをに

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

6.0

デフォルトのSETは 'YYYYMMDD.HHmmss'

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

ヒント Androidのをローカル Linuxベースのマシンとさせるには

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

オープンオプション

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

デバイス/エミュレータを`Developer Options`セクションにします。

「ブート」ブロードキャストの

これは、 `BootListener`をするアプリケーションにします。あなたのアプリをすことによってあなたのアプリをテストし、にテストする

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

`your.package/your.app.BootListener`をなに`your.package/your.app.BootListener`てください。

/セカンダリストレージコンテンツの

コンテンツをする

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

ビューパス

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

Androidデバイスのプロセスをする

AndroidのlogcatIは、あなたがしていないプロセスからのエラー、バッテリーの、またはコードをデバッグするのをしくするなど、にされていることがあります。

デバイスをせずにをするなは、のとなっているプロセスをしてさせることです。

Logcatから

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

プロセスにく1550

これで、シエルをいてプロセスをできます。 `root`プロセスをすることはできません。

```
adb shell
```

シェルのでは、プロセスのをすることができます

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

たちがむならそれをす

```
kill -9 1550
```

オンラインでadbシェルをむ https://riptutorial.com/ja/android/topic/9408/adbシェル

6: ADBをしたアプリのインストール

Examples

アプリをインストールする

にのコマンドをきます

```
adb install [-rtsdg] <file>
```

あなたのコンピュータではなく、あなたのデバイスにあるファイルをさなければならないことにしてください。

に`-r`をけると、のするapkがきされます。それの、コマンドはエラーでします。

`-g`はすべてのアクセスをにします。

`-d`はバージョンコードのダウングレードをにしますデバッグなパッケージでのみ。

SDカードにアプリケーションをインストールするには、 `-s`をします。

`-t`では、テストアプリケーションをできます。

アプリをアンインストールする

にのコマンドをして、されたパッケージでアプリをアンインストールします。

```
adb uninstall <packagename>
```

すべてのapkファイルをディレクトリにインストールする

Windows

```
for %f in (C:\your_app_path\*.apk) do adb install "%f"
```

Linux

```
for f in *.apk ; do adb install "$f" ; done
```

オンラインでADBをしたアプリのインストールをむ
https://riptutorial.com/ja/android/topic/5301/adbをしたアプリのインストール

7: AdMob

- 'com.google.firebasefirebase-ads10.2.1'をコンパイルしてください。//をにする
- `<uses-permission android:name="android.permission.INTERNET" />`をするためにです
- AdRequest adRequest = new AdRequest.Builder。build; //バナー
- AdView mAdView =AdViewfindViewByIdR.id.adView; //バナー
- mAdView.loadAdadRequest; //バナー

パラメーター

Param	
adsadUnitId = "@ string / main_screen_ad"	のID。 admobサイトからあなたのIDをしてください。「ではありませんが、リソースファイルにユニット*ID*のをすることはいです。アプリがし、のがするがあるため、*ID*をするがあります。ファイルをしてコードをするはありません」[1]

- なAdMobアカウントがです
- admobポリシーをんでください。あなたはあなたのadmobアカウントをさせることができるもしないでください

Examples

このでは、な*Admob*アカウントとな*Admob*コードがです。

アプリレベルでBuild.gradle

のはバージョンにします。

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

マニフェスト

データにアクセスするにはインターネットアクセスがです。このアクセスは、のアクセスでありではないため、API 23+をしてリクエストするはありません。

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

XML

のXMLは、バナーをしています。

```
<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" />
```

のタイプのコードについては、 Google AdMobヘルプをごください。

Java

のコードは、バナーのです。のタイプでは、なるがながあります。

```
// 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);
```

`onResume()` 、 `onPause()` 、および`onDestroy()`メソッドに`AdView`ライフサイクルメソッドをします。

```
@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();
}
```

オンラインでAdMobをむ https://riptutorial.com/ja/android/topic/5334/admob

8: AIDL

き

AIDLはAndroidインターフェースです。

どうしてどうやって

それはのサービスです。このAIDLサービスは、クライアントのなくとも1つがするまでアクティブになります。マーシャリングとアンマーシャリングのコンセプトについてします。

どうしてリモートアプリケーションはサービス+マルチスレッドリモートアプリケーションにアクセスできます。

どうやって .aidlファイルをするインターフェイスをするインターフェイスをクライアントにする

Examples

AIDLサービス

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;
        }
    };

}
```

サービス

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

オンラインでAIDLをむ https://riptutorial.com/ja/android/topic/9504/aidl

9: AlarmManager

Examples

インテントをでする

1. をします。このクラスはインテントをけり、あなたがむようにそれをします。

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

2. インテントをAlarmManagerにします。このでは、1にAlarmReceiverにされるインテントがトリガーされます。

```
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 );
```

アラームをキャンセルする

アラームをキャンセルするには、アラームのにされたのPendingIntentへのがないは、PendingIntentをにしたときとじようにするがあります。

インテントは、AlarmManagerによってしいとみなされます 。

> そのアクション、データ、タイプ、クラス、およびカテゴリがじである。これはインテントにまれるなデータをしません。

、アラームのコードはとしてされます。

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

したがって、このようにされたなアラームの

```
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);
```

しいAlarmManagerでアラームをキャンセルするためにできるしいPendingIntentをするはのとおりです。

```
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);
}
```

すべてのAndroidバージョンでなアラームをする

がつにつれて、Androidシステムにバッテリのがますますんでいるため、 `AlarmManager`のメソッドもなタイミングをにするためににされています。しかし、のアプリケーションでは、すべてのAndroidバージョンでできるだけにするがあります。のヘルパーは、すべてのプラットフォームでなもなメソッドをして、 `PendingIntent`をスケジュールし`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);
    }
}
```

API23 + DozeモードはAlarmManagerとします

Android 6API23はDozeモードをし、AlarmManagerとします。のメンテナンスウィンドウをしてアラームをするため、 `setExactAndAllowWhileIdle()`をしても、なでアラームがすることをすることはできません。

あなたのの `Settings/General/Battery & power saving/Battery usage/Ignore optimizations`などをしてアプリのこのをオフにすることができます

あなたのアプリのでこのをチェックすることができます...

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

...そしてにそれぞれのダィアログをします

```
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);
```

オンラインでAlarmManagerをむ https://riptutorial.com/ja/android/topic/1361/alarmmanager

10: Android Javaネイティブインタフェースの JNI

き

JNI Java Native Interfaceは、AndroidがアプリケーションでNDKをしてC ++ネイティブコードをできるなツールです。このトピックでは、Java < - > C ++インタフェースのについてします。

Examples

JNIインタフェースをしてネイティブライブラリのをびす

Javaネイティブインタフェース JNIをすると、Javaコードからネイティブをびすことができます。このでは、JNIをしてネイティブをロードおよびコールするをします.JNIをしてネイティブコードからJavaメソッドおよびフィールドにアクセスすることはありません。

`project/libs/<architecture>`フォルダに`libjniexample.so`というのネイティブライブラリがあり、`com.example.jniexample`パッケージの`JNITest` Javaクラスからをびしたいとします。

JNITestクラスで、のようなをします。

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

ネイティブコードで、のようなをします。

```
#include <jni.h>

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

`pEnv`は、JavaオブジェクトおよびクラスのメソッドおよびフィールドにアクセスするためにJNIにすことができるJNIへのポインタです。 `thiz`ポインタは、ネイティブメソッドがびされたJavaオブジェクトメソッドのはクラスへの`jobject`です。

Javaコードの`JNITest` 、のようにライブラリをロードします。

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

の`lib`にしてください。ファイルのにある`.so`はされています。

Javaのネイティブをのようにびします。

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

ネイティブコードからJavaメソッドをびす

Javaネイティブインタフェースを使用すると、Javaをネイティブコードからびすことができます。これをうのなをにします。

Javaコード

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

ネイティブコード

```
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);
}
```

JNIメソッドシグネチャからJavaへの

JNI	Javaタイプ
Z	ブール
B	バイト
C	チャー
S	ショート
	int
J	いです
F	く

JNI	Javaタイプ
D	ダブル
Lクラス。	クラス
[タイプ	タイプ[]

したがって、このでは、ZIをしました。つまり、はブーールをし、intをします。

JNIにおけるユーティリティメソッド

このメソッドは、JavaをC ++からするのにちます。

```
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;
    }
```

このメソッドは、jbyteArrayをcharにするのにちます

```
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;
}
```

オンラインでAndroid Javaネイティブインタフェースjniをむ
https://riptutorial.com/ja/android/topic/8674/android-javaネイティブインタフェース-jni-

11: Android NDK

Examples

Androidのネイティブファイルの

プロジェクト/ jni / main.c

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

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

project / jni / Android.mk

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

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

プロジェクト/ jni / Application.mk

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

Androidバージョンが5.0API 21のをサポートするには、 `APP_PLATFORM`をいAPIにしてバイナリをコンパイルするがあります `android-8` 。 これは、Android 5.0がバイナリ PIEをしたですが、いデバイスはずしもPIEをサポートしていません。したがって、デバイスのバージョンにじて、PIEまたはPIEのいずれかをするがあります。 Androidアプリケーションからバイナリをするは、APIレベルをチェックしてしいバイナリをするがあります。

`APP_ABI`は、 `armeabi`などののプラットフォームにして、これらのアーキテクチャのバイナリのみをビルドすることができます。

の、アーキテクチャごとにPIEとPIEののバイナリndk-r10eをする14のなるバイナリがあります。

ファイルをビルドするには

```
cd project
ndk-build
```

バイナリは`project/libs/<architecture>/hello_world`ます。 あなたは`ADB` パーミッションで`push`と`chmod`

またはアプリケーションからパーミッションでそれをしてchmodすることができます。

CPUのアーキテクチャをするには、ビルドプロパティのro.product.cpu.abiなアーキテクチャまたはのためro.product.cpu.abilistサポートされているアーキテクチャのなリストについては、しいデバイスに。アプリケーションからandroid.os.Buildクラスをするか、ADBでgetprop <name>をしてこれをうことができます。

ビルドのクリーンアップ

ビルドをきれいにするがある

```
ndk-build clean
```

Android.mkのメイクファイルのい

ndk-build NDK_PROJECT_PATH = PROJECT_PATH APP_BUILD_SCRIPT = MyAndroid.mk

ndkへのログイン

まず、 Android.mkファイルのロギングライブラリにリンクされていることをしてください

```
LOCAL_LDLIBS := -llog
```

に、の__android_log_print()びしのいずれかをします。

```
#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)
```

または、するマクロをすることで、よりなでそれらをします。

```
#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__)
```

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

オンラインでAndroid NDKをむ https://riptutorial.com/ja/android/topic/492/android-ndk

12: Android Paypalゲートウェイの

Paypalはたちのライブラリをいのためにしているので、ではアプリケーションでするのがにでです。はなステップです。

Examples

あなたのアンドロイドコードでpaypalをする

1まず、Paypal DeveloperのWebサイトにアクセスし、アプリケーションをします。

2マニフェストファイルをき、のをえます

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

3そしていくつかのなとサービス -

```
 <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あなたのアクティビティクラスをき、あなたのアプリケーションのをします。

```
//set the environment for production/sandbox/no netowrk
 private static final String CONFIG_ENVIRONMENT = PayPalConfiguration.ENVIRONMENT_PRODUCTION;
```

5これで、Paypalアカウントのクライアントidをします。 - プライベートCONFIG_CLIENT_ID = "あなたのクライアントidをしてください"; 6onCreateメソッドで、Paypalサービスをびします。 - Intent intent = new Intentthis、PayPalService.class; intent.putExtra PayPalService.EXTRA_PAYPAL_CONFIGURATION、config; startServiceintent;

7すぐあなたはボタンをすだけでいをするがいました。いアクティビティ -

```
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そしてにonActivityResultからいをします。

```
 @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.");
                   }
               }


           }
```

オンラインでAndroid Paypalゲートウェイのをむ
https://riptutorial.com/ja/android/topic/5895/android-paypalゲートウェイの

13: Android Places API

Examples

プレ イスピッカーの

Place Pickerは、Places APIによってされるとてもシンプルなUIウィジェットです。それは、、、くの、とオートコンプリートをします。

これは、Place Picker UIウィジェットのです。

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

Places APIをしたのの

GoogleプレイスAPIをすると、のとユーザーのローカルのをできます 。

まず、 `PlaceDetectionApi.getCurrentPlace()`メソッドをびして、のビジネスやそののをするがあります。このメソッドは、す`PlaceLikelihoodBuffer`リストがまれるオブジェクト`PlaceLikelihood`オブジェクトを。に、 `PlaceLikelihood.getPlace()`メソッドをびして`Place`オブジェクトをできます。

あなたのアプリがなにアクセスできるようにするには、 **`ACCESS_FINE_LOCATION`**をリクエストしてするがあります。

```
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());
}
```

プレイスオートコンプリートインテグレーション

AndroidGoogleプレイスAPIのオートコンプリートは、ユーザーにプレースをします。ユーザーがボックスにしている、オートコンプリートはユーザーのクエリについてをします。

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"/>
```

の**Google**アクティビティをしています。

```
 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.
           }
        }
    }
```

PlaceAutocompleteのプレースタイプフィルタの

によっては、 **PlaceAutocomplete**によってのにされているをりんだり 、のみをしたりすることもできます。これは、インテントにオートコンプリートフィルタをすることでできます。たとえば、REGIONののみをし、インドにしているは、のようにします。

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がにし、からされたにのみする**REGION**のをできます。インテントは、ボタンをクリックしてすることもできます。

オンラインでAndroid Places APIをむ https://riptutorial.com/ja/android/topic/4111/android-places-api

14: Android Studioでインスタント

インスタント·ランは、アプリケーションのコードでされるeevryのなビルドとインストールをとせずに、よりなデバッグをにするrunおよびdebugコマンドのです。

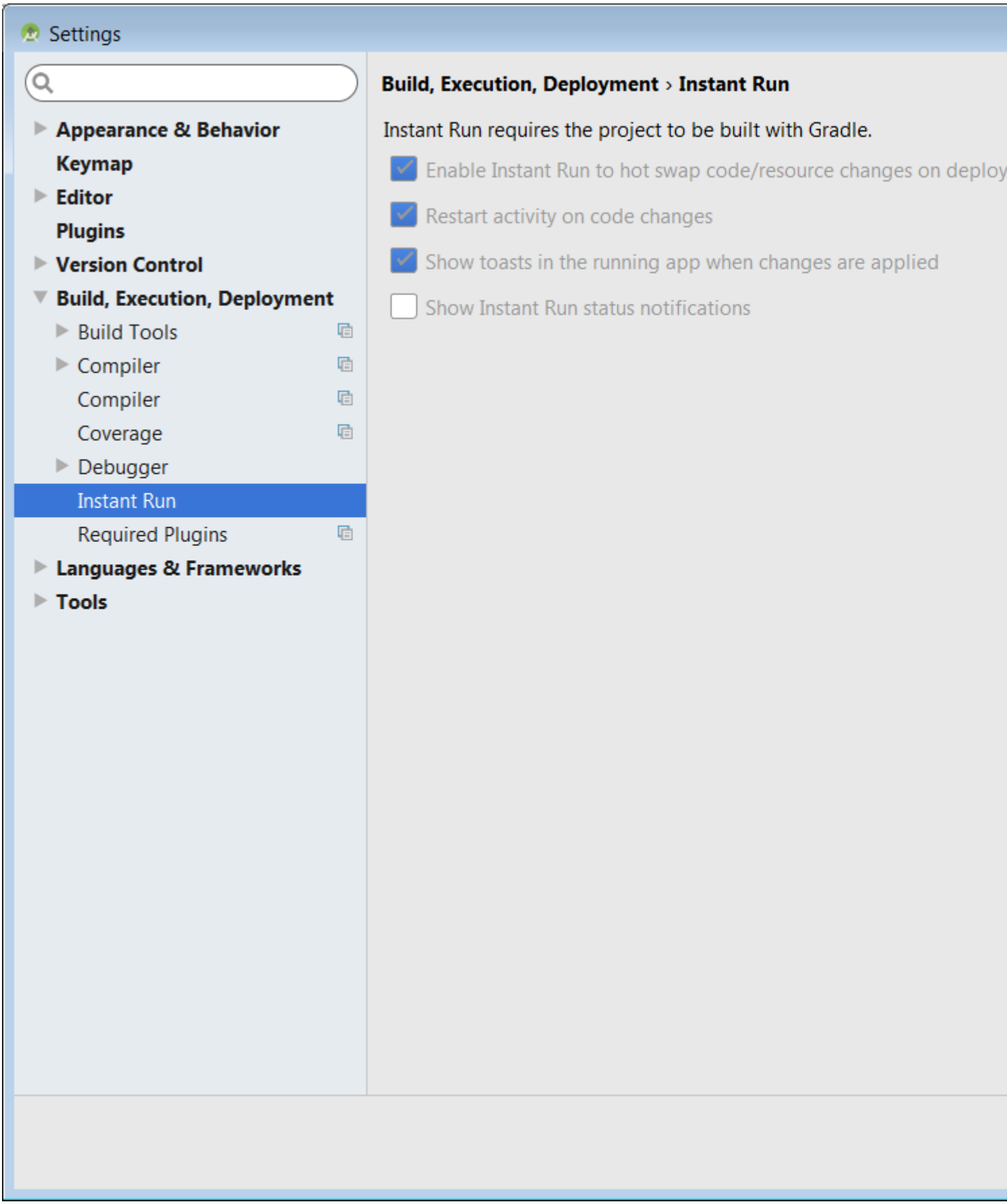
> Android Studio 2.0でされたInstant Runは、アプリのをにするRunおよびDebugコマンドのです。のビルドがするまでにがかかることがありますが、Instant Runは、しいAPKをすることなく、そののアップデートをアプリにプッシュするので、がはるかににされます。
>
> インスタントは、デバッグビルドバリアントをデプロイし、Androidプラグインを Gradleバージョン2.0.0でし、アプリのモジュールレベルbuild.gradleファイルで minSdkVersionを15にしたにのみサポートされます。のパフォーマンスをるには、 minSdkVersionを21にします。
>
> アプリをデプロイすると、[]ボタンまたは[デバッグ]ボタンにさなのアイコンがされ、にボタンをクリックしたときにインスタント·ランがをプッシュするができていることがされます。しいAPKをするのではなく、しいをプッシュするだけでなく、によってはするもなく、すぐにそのコードのをすることができます。
>
> Instant Runはホットスワップ、ウォームスワップ、またはコールドスワップをして、されたコードとリソースをされたデバイスまたはエミュレータにプッシュします。のタイプにづいてするスワップのタイプをにします。のビデオは、これがどのようにしてボンネットのでするかについてのいをします。ただし、のコードをターゲットデバイスにプッシュするにInstant Runがどのようにするかのなについては、のをしてください。

ドキュメンテーション

Examples

インスタントをまたはにする

1. またはダイアログをきます
   - WindowsまたはLinuxのは、メインメニューから[`File Settings` > `Settings`をします。
   - Mac OSXでは、メインメニューから`Android Studio` > `Preferences`します。
2. `Build, Execution, Deployment` > `Compiler`ナビゲートします。
3. [コマンドラインオプション]のにあるテキストフィールドに、コマンドラインオプションをします。
4. [OK]をクリックしてしてします。



のオプションはインスタントです。そのチェックボックスをオンまたはオフにします。

ドキュメンテーション

でのコードスワップの

インスタントでは、Android Studioのコードからデバッグをし、アプリケーションをするための3のコードスワップがあります。

- ホットスワップ
- ウォームスワップ
- コールドスワップ

これらのスワップのそれぞれがいつトリガされますか

HOT SWAPは、のメソッドのがされたときにトリガーされます。

WARM SWAPは、のリソースがまたはされたときにされますresフォルダのすべてのもの

あなたのアプリのコードにコードがされたときにいつでもスワップする

1. 、、または

-
- インスタンスフィールド
- フィールド
- メソッドシグネチャ
- インスタンスメソッドのシグネチャ

2. のクラスがするクラスをする
3. されたインターフェースのリストをする
4. クラスのをする
5. リソースIDをするレイアウトのべえ

コードスワップがこるとどうなりますか

HOT SWAPのは、がされたメソッドへののびしがわれるとすぐにされます。

WARM SWAPはのアクティビティをします

COLD SWAPはアプリをしますインストールはありません

インスタントをするとサポートされていないコードがされる

インスタントがそのトリックをやっていない、いくつかのがあり、インスタントランがまれるにこったのとじようにあなたのアプリのためのなビルドとインストールがわれます。

1. アプリのマニフェストをする
2. アプリマニフェストがするリソースをする
3. AndroidウィジェットのUIをするクリーンアンドが

ドキュメンテーション

オンラインでAndroid Studioでインスタントをむ
https://riptutorial.com/ja/android/topic/2119/android-studioでインスタント

15: Android StudioにOpenCVをする

Open CVライブラリは、Webでエンジンをしてつけることができます。

ゴッチャ

- あなたのタ—ゲットプラットフォ—ムをKitKatにげると、OpenCVライブラリのいくつか、に*org.opencv.android.Camera2Renderer*およびのクラスにするクラスはしなくなります。このをするには、なOpenCV .javaファイルをするだけです。
- タ—ゲットプラットフォ—ムをLollipopにした、ファイルパスのにわされてしまったため、のロ—ドではファイルがみまれないがあります。だから、ギャラリ—やのからファイルをロ—ドするためにサンプルをしなければならないかもしれません。にかふくのがあります。

Examples

AS v1.4.1でテストされましたが、しいバ—ジョンでもするはずです。

1. プロジェクトウィザ—ドをしてしいAndroid Studioプロジェクトをしますメニュ—/ファイル/プロジェクト

   - それを " **cvtest1** "
   - フォ—ムファクタ **API 19**、Android 4.4**KitKat**
   - **MainActivity**というアクティビティ

   このプロジェクトがされている*cvtest1*ディレクトリがです。 Androidスタジオのタイトルバ—には、プロジェクトをいたときのcvtest1のがされます

2. あなたのアプリがしくすることをしてください。 "Hello World"テキストのようなものをして、ビルド/テストサイクルがであることをしてください。 はAPI 19デバイスのエミュレ—タでテストしています。

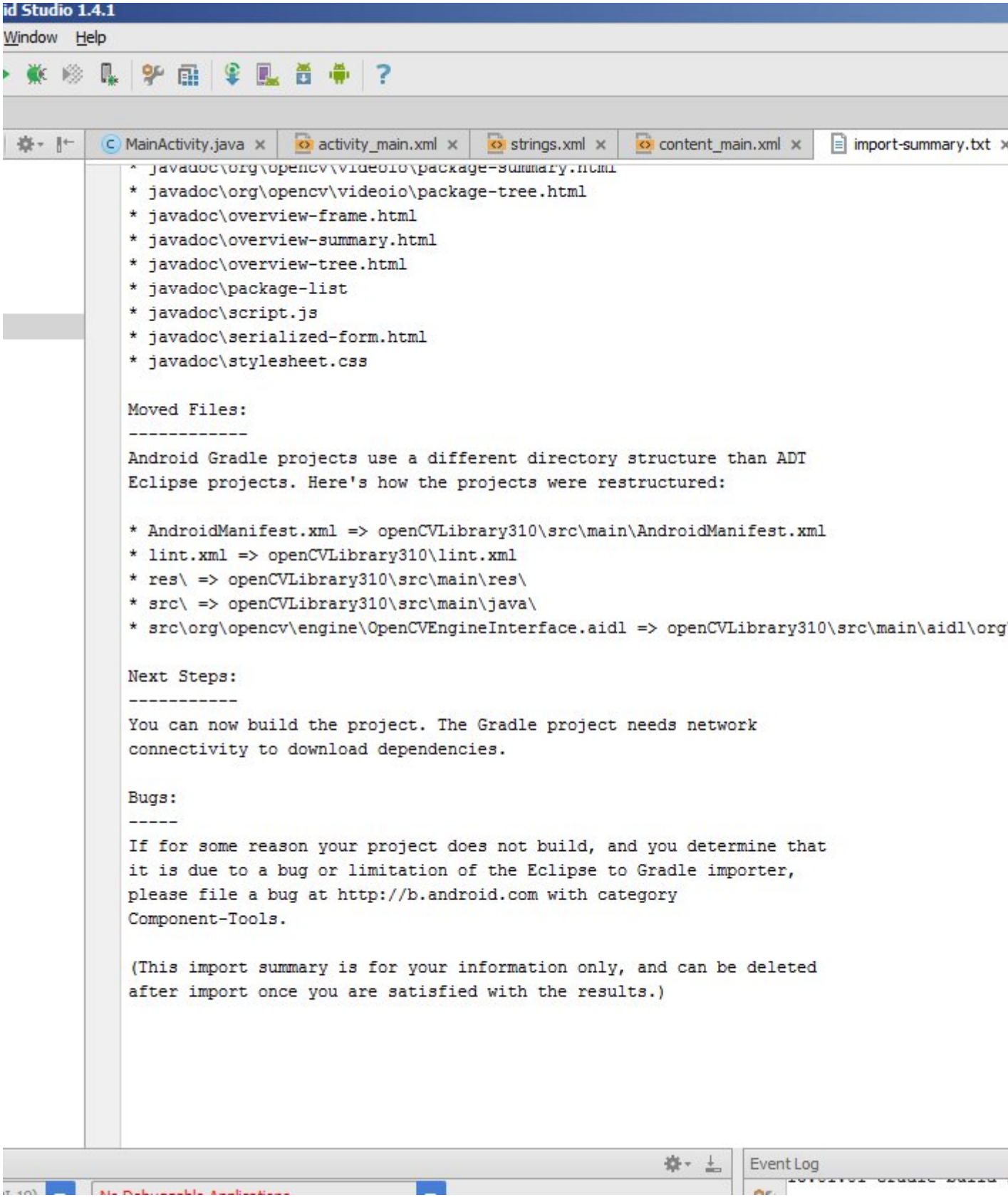
3. Android v3.1.0のOpenCVパッケ—ジをダウンロ—ドし、どこかのディレクトリにしてください。 これはJavaパッケ—ジのためののAndroidパッケ—ジだけではなく、OpenCVのであることをしてください。は、「 **-dir**の 」**-dir**のに、このディレクトリふことにします、あなたがでまるサブディレクトリと**SDK /**ネイティブ**/ libs**ディレクトリをつがあります**arm** ...、**mips** ...、 **x86**のようなもの...Androidがっている "ア—キテクチャ"のタイプごとに1つ

4. Android StudioからプロジェクトとしてOpenCVをモジュ—ルとしてインポ—ト **Menu/ File / New / Import_Module**

   - ソ—スディレクトリ **{unzip-dir} / sdk / java**
   - モジュ—ルAndroidスタジオはにこのフィ—ルドを**openCVLibrary310**でめますなはありませんが、これにいます。
   - へをクリックします。 jar、ライブラリ、およびインポ—トオプションにする3つのチ

エックボックスとがあるがされます。 3つすべてをチェックするがあります。 **Finish** をクリックします。

Android Studioからモジュールのインポートがされ、 **import-summary.txt**ファイルがされます。このファイルには、インポートされなかったものにjavadocファイルやそののがされます。



しかし、ハッシュ **'android-14'**をってターゲットをつけることができなかったというエラーメッセージがされます。これは、あなたがダウンロードしたOpenCV zipファイルのbuild.gradleファイルがアンドロイドAPIバージョン14をしてコンパイルするようにするためです。これは、デフォルトではAndroid Studio v1.4.1にはありません。

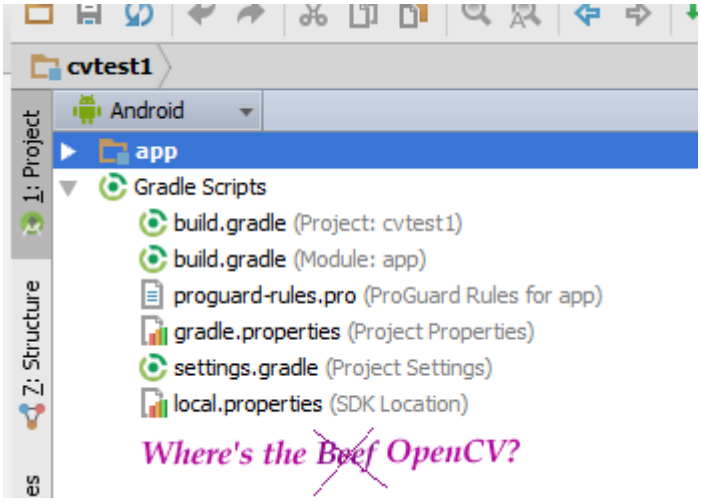


5. プロジェクトダイアログをきます メニュー**/**ファイル**/**プロジェクト。「app」モジュールをし、「**Dependencies**」タブをクリックして、モジュールのとして**openCVLibrary310**をします。 **Add / Module_Dependency**をすると、できるモジュールのリストにされます。これはとしてされますが、イベントログにはくの**can not-find-android-14**エラーがされます。

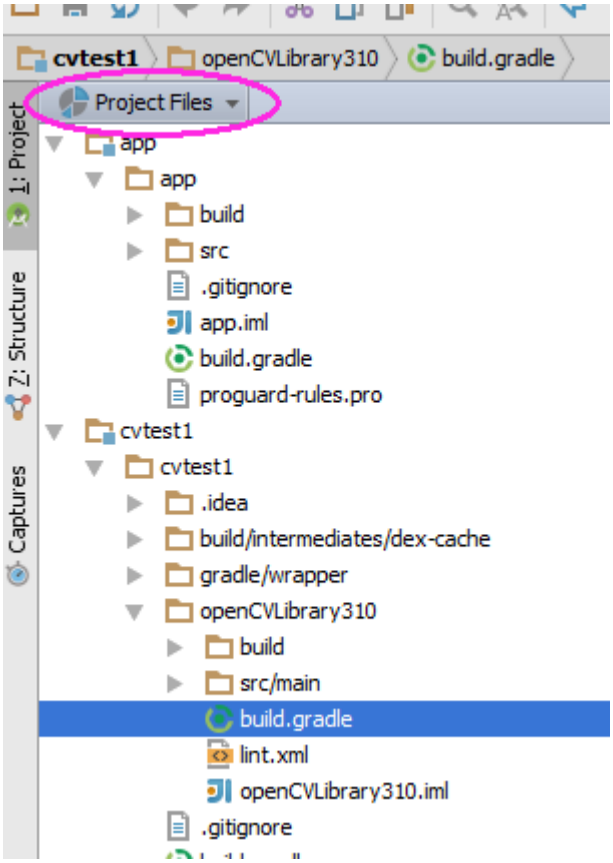
6. あなたのアプリケーションモジュールの**build.gradle**ファイルをてください。 Androidプロジェクトにはのbuild.gradleファイルがあります。なものは**cvtest1 / app**ディレクトリにあり、プロジェクトビューからは**build.gradleModuleapp**のようにえます。これらの4つのフィールドのにしてください。

   - compileSDKVersionのう23
   - buildToolsVersionのう23.0.2
   - minSdkVersionのう19
   - targetSdkVersionのう23

7. プロジェクトに**cvtest1 / OpenCVLibrary310**ディレクトリがされましたが、プロジェクトビューからはされません。

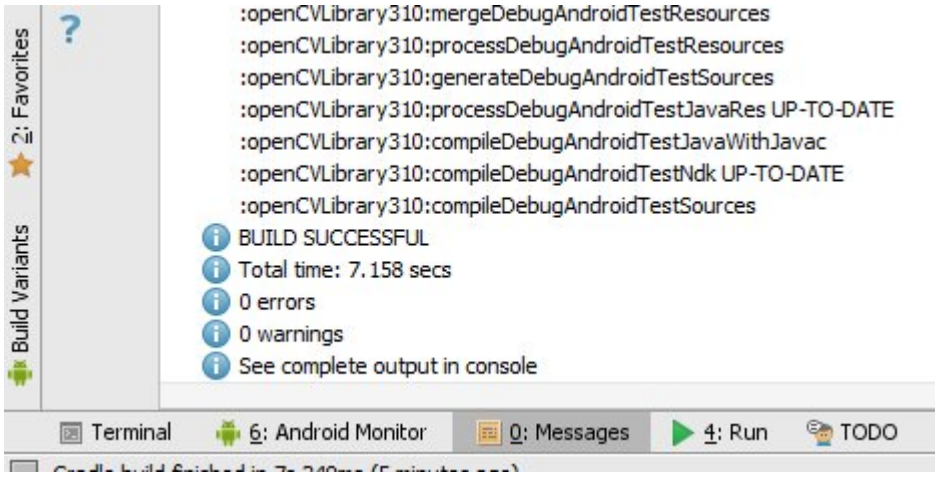


のツールファイルマネージャなどをして、このディレクトリにします。プロジェクトビューを**Android**から**Project**ファイルにりえることもできます。このスクリーンショットにすように、このディレクトリをつけることができます



にはの**build.gradle**ファイルのスクリーンショットでされていますがあります。このファイルを6の4つのでします。

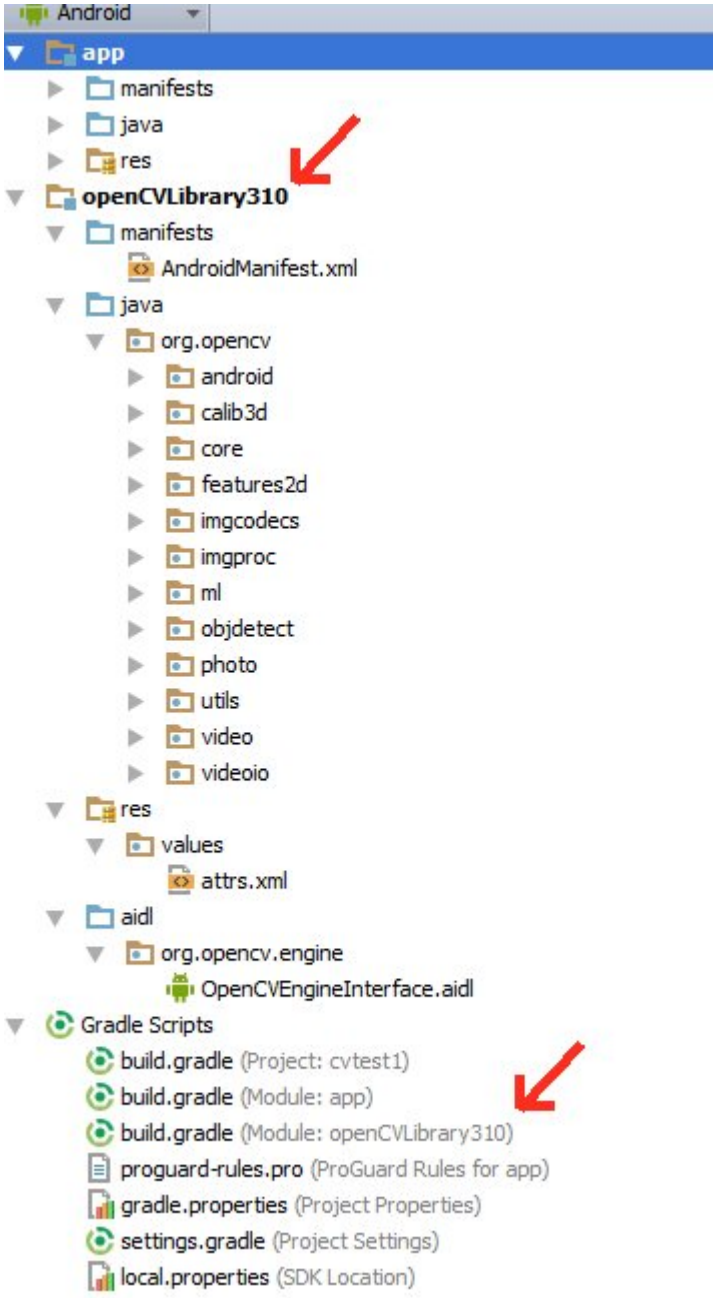
8. プロジェクトをして、それをクリーニング/します。 **Menu/ Build / Clean_Project** エラーをさせずにクリーンアップしてビルドするがあります。0**Messages**の**openCVLibrary310**へのがくあります。



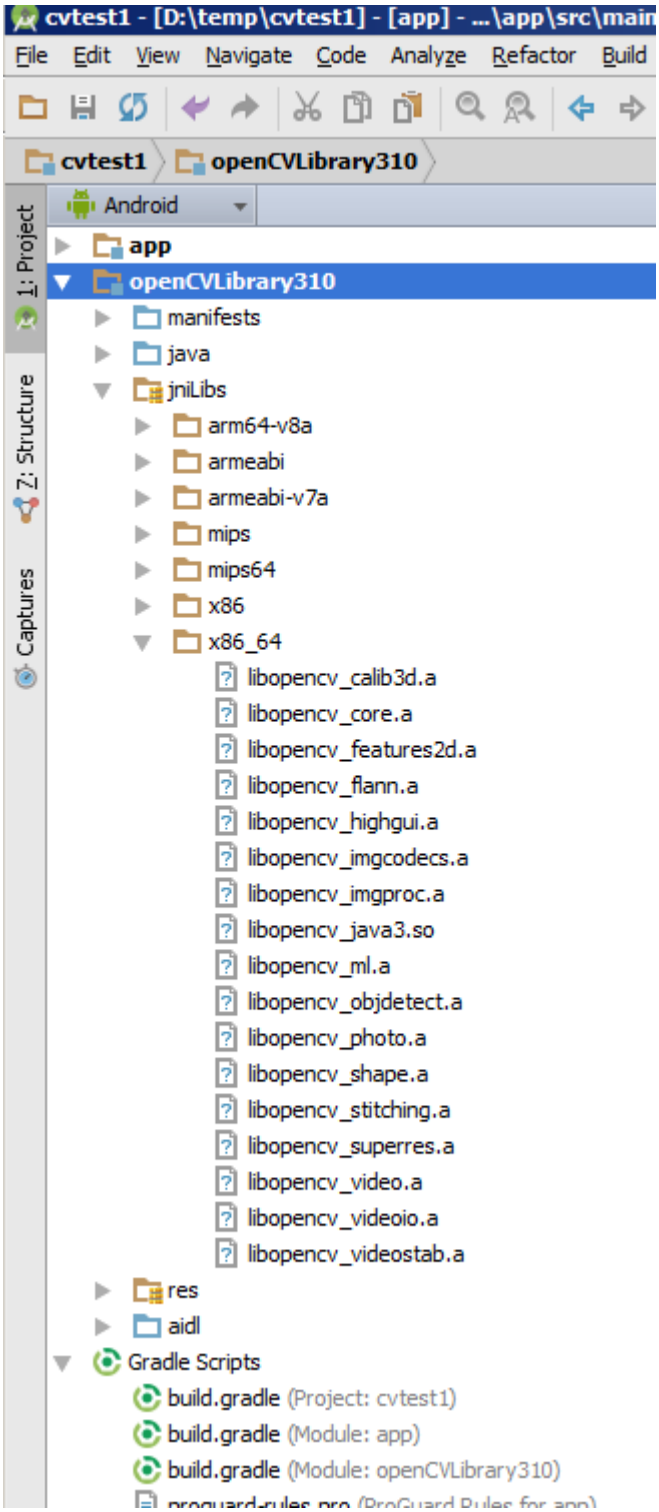
こので、モジュールはアプリケーションのようにプロジェクトに**openCVLibrary310**としてされます。 そのさなドロップダウンメニューでは、は**Project View**から**Android View**にったことにしてください。また、「Gradle Scripts」のに**build.gradle**ファイルがされているはずですが、Androidスタジオのインターフェースがちょっとです。ですから、Android Studioの、クリーニング、までおしください。

このスクリーンショットのように、javaのにあるすべてのOpenCVをむopenCVLibrary310モ

ジュールがされます。



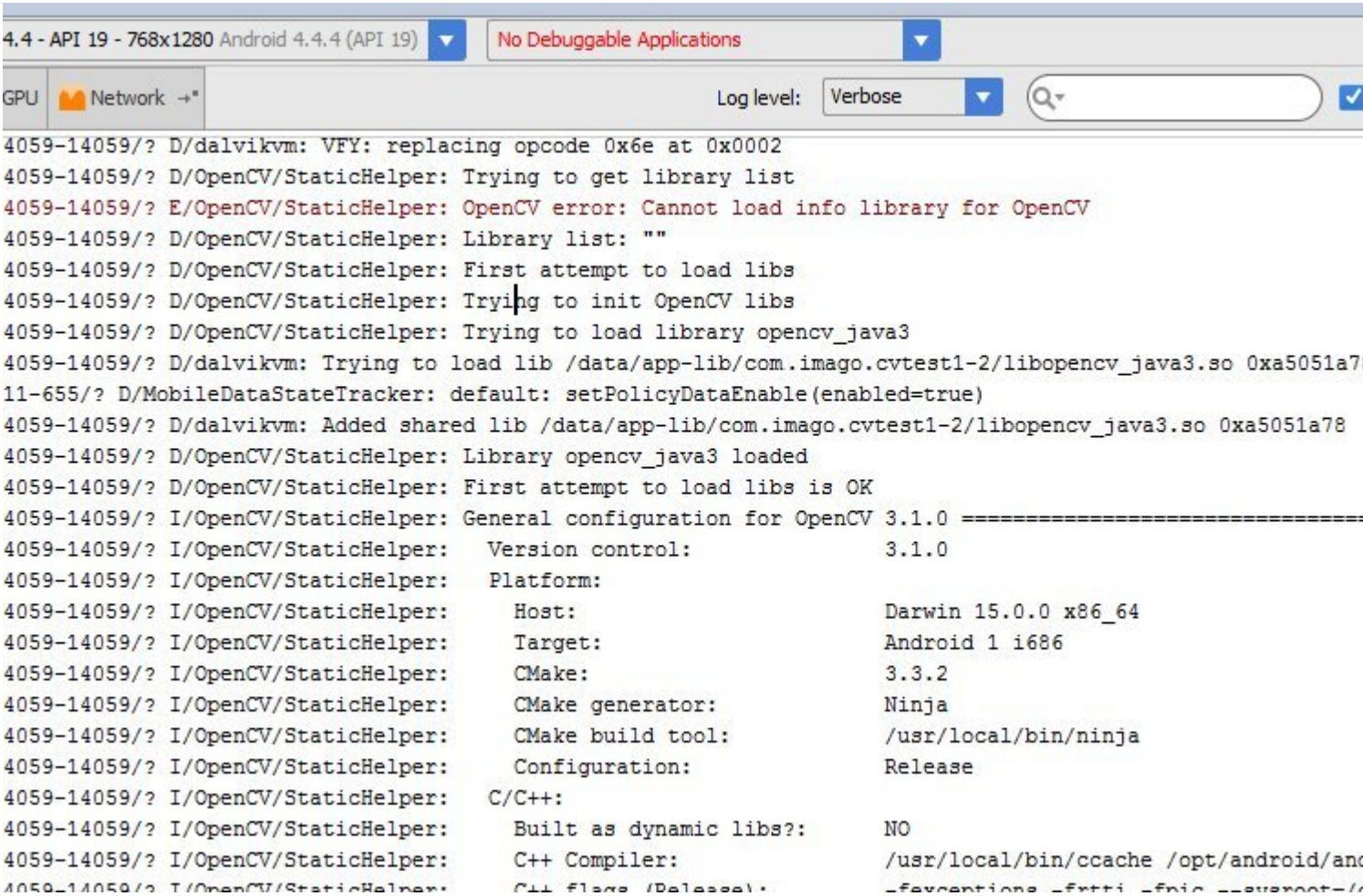
9. Androidプロジェクトに**{unzip-dir} / sdk / native / libs**ディレクトリおよびそののすべてをコピーして、 **cvtest1 / OpenCVLibrary310 / src / main /**にコピーし、あなたのコピーのを**libs**から**jniLibs**にします。これで、 **cvtest1 / OpenCVLibrary310 / src / main / jniLibs**ディレクトリがされます。プロジェクトをすると、このディレクトリはプロジェクトビューの**openCVLibrary310**にされます 。



10. *MainActivity.java*のonCreateメソッドにし、のコードをします。

```
if (!OpenCVLoader.initDebug()) {
        Log.e(this.getClass().getSimpleName(), "  OpenCVLoader.initDebug(), not working.");
    } else {
        Log.d(this.getClass().getSimpleName(), "  OpenCVLoader.initDebug(), working.");
    }
```

その、アプリケーションをします。 Android Monitorでは、のようながされます。



なぜはそのにエラ―メッセ―ジがあるのかわかりません

11. にopenCVコードをにしようとします。のでは、アンドロイドエミュレ―タのcvtest1アプリケ―ションのキャッシュディレクトリに.jpgファイルをコピ―しました。のコ―ドはこのをみみ、エッジアルゴリズムをし、そのをじディレクトリの.pngファイルにきします。

```
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. アプリケーションをします。あなたのエミュレータは、の「エッジ」イメージをするがあります。 Androidデバイスモニタをしてをしたり、アクティビティをきんですることができます。

オンラインでAndroid StudioにOpenCVをするをむ
https://riptutorial.com/ja/android/topic/7068/android-studioにopencvをする

16: Android Vk Sdk

Examples

とログイン

1. ここにしいアプリケーションをする アプリケーションをする
2. スタンドアロンアプリケーションをし、SMSでアプリのをします。
3. のパッケージとして**Android**の**Package name**をします。 にで、あなたのパッケージをアンドロイドマニフェストファイルのにれることができます。
4. シエル/ cmdでのコマンドをして、 のフィンガープリントをします。

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

また、SDKでこのフィンガープリントをすることもできます。

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

5. したを**Android**のフィンガープリントにする Vkアプリのフィールドパッケージをした

6. にこれをgradleファイルにします

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

8. のでにSDKをします。のは、アプリケーションのonCreateメソッドでびすことです。

```
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);
}
```

これがVKSdkをするのです。 VK_IDがstrings.xmlのにかれるべきところにmethidをしないでください。なぜなら、apiはそのにしくしないからです。

9. ステップは、vksdkをしてログインすることです。

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

オンラインでAndroid Vk Sdkをむ https://riptutorial.com/ja/android/topic/6046/android-vk-sdk

17: Android-x86 in VirtualBox

き

このセクションのアイデアは、デバッグのでAndroid-x86でVirtualBoxをインストールしてするをカバーすることです。バージョンにいがあるため、これはしいです。では、はがをしなければならなかったものである6.0をカバーするつもりです。そして、々はをつけなければなりません。

VirtualBoxやLinuxについてしくはれていませんが、それがさせるためにしたコマンドをしています。

Examples

マシンのセットアップ

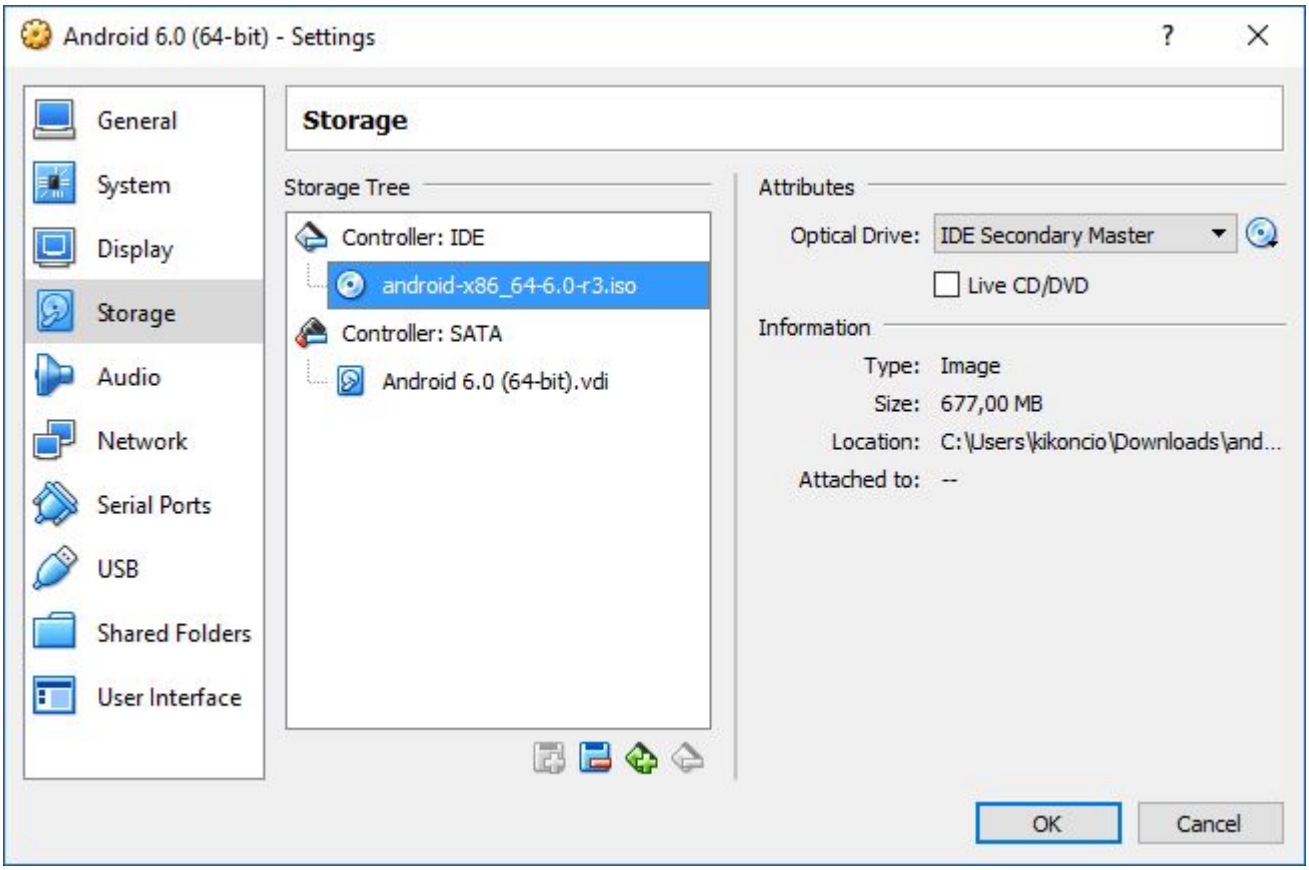
はのVirtualBoxのです

- OSのLinux 2.6のコンピュータはそれをサポートできるのでは64ビットのユーザをっています
- ハードドライブのサイズ4Gb
- RAMメモリ2048
- ビデオメモリ8M
- サウンドデバイスSound Blaster 16
- ネットワークデバイスNATにされたPCnet-Fast III。ブリッジアダプタをすることもできますが、ごのにDHCPサーバーがです。

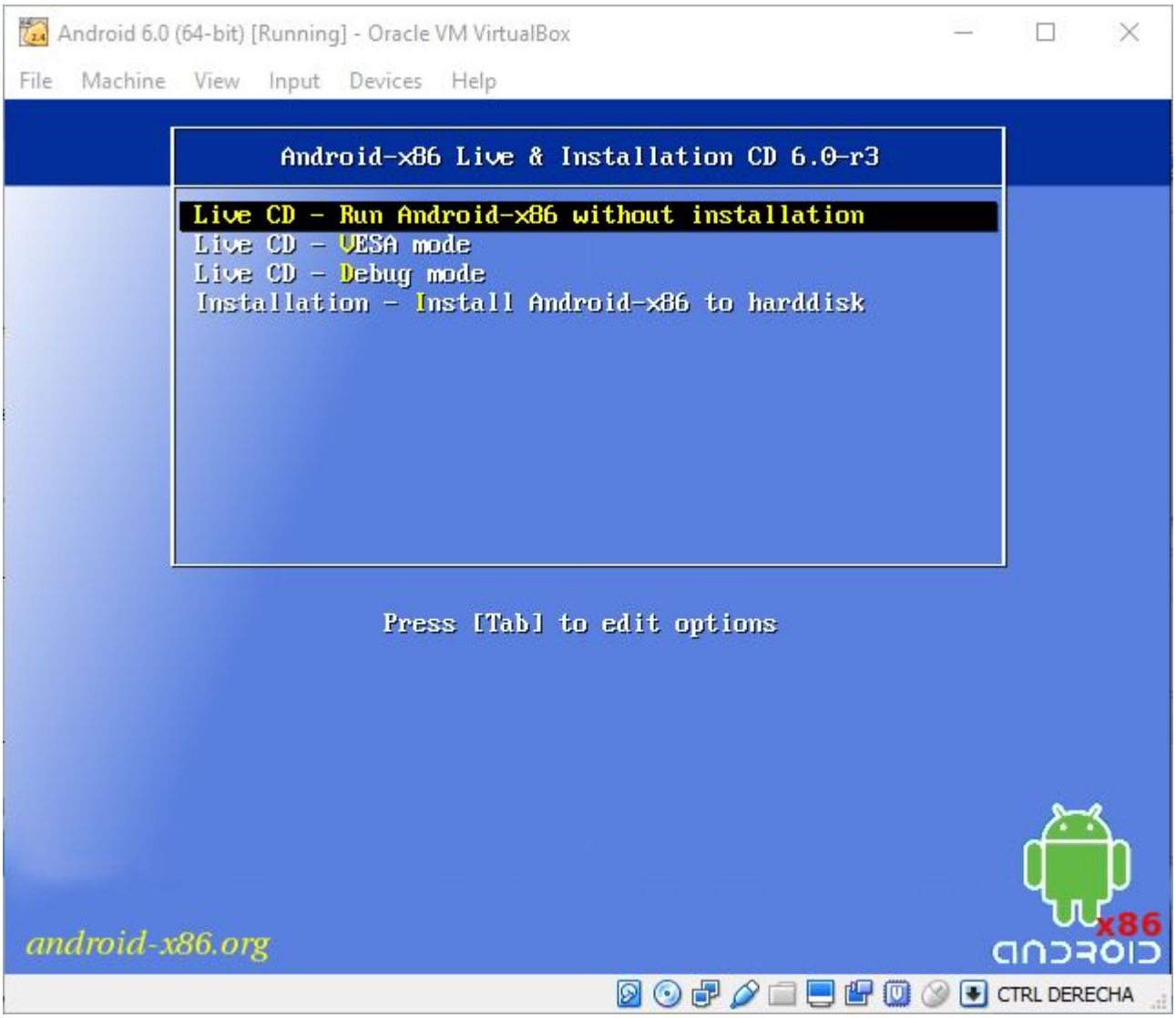
このでされるは、 http //www.android-x86.org/downloadからダウンロードされたandroid-x86_64-6.0-r3.iso64ビットです。はそれも32ビットでするといます。

SDCARDサポートのハードドライブのセットアップ

ハードドライブをした、アンドロイド-X86イメージをしてマシンをオプティカルドライブでします。



すると、Live CDのgrubメニューがされます



デバッグモードオプションをすると、シェルプロンプトがされます。これはビジーボックスシェルです。コンソールAlt-F1 / F2 / F3をりえることで、よりくのシェルをにれることができます。

fdiskで2つのパーティションをしますのバージョンではcfdiskをします。それらをext3にフォーマットします。その、します。

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

にタイプする

"n"しいパーティション

"p"プライマリパーティション

"1"1のパーティション

"1"1

"261"シリンダをすると、ディスクの50を2のパーティションにします

"2"2のパーティション

「262」262

"522"のシリンダを

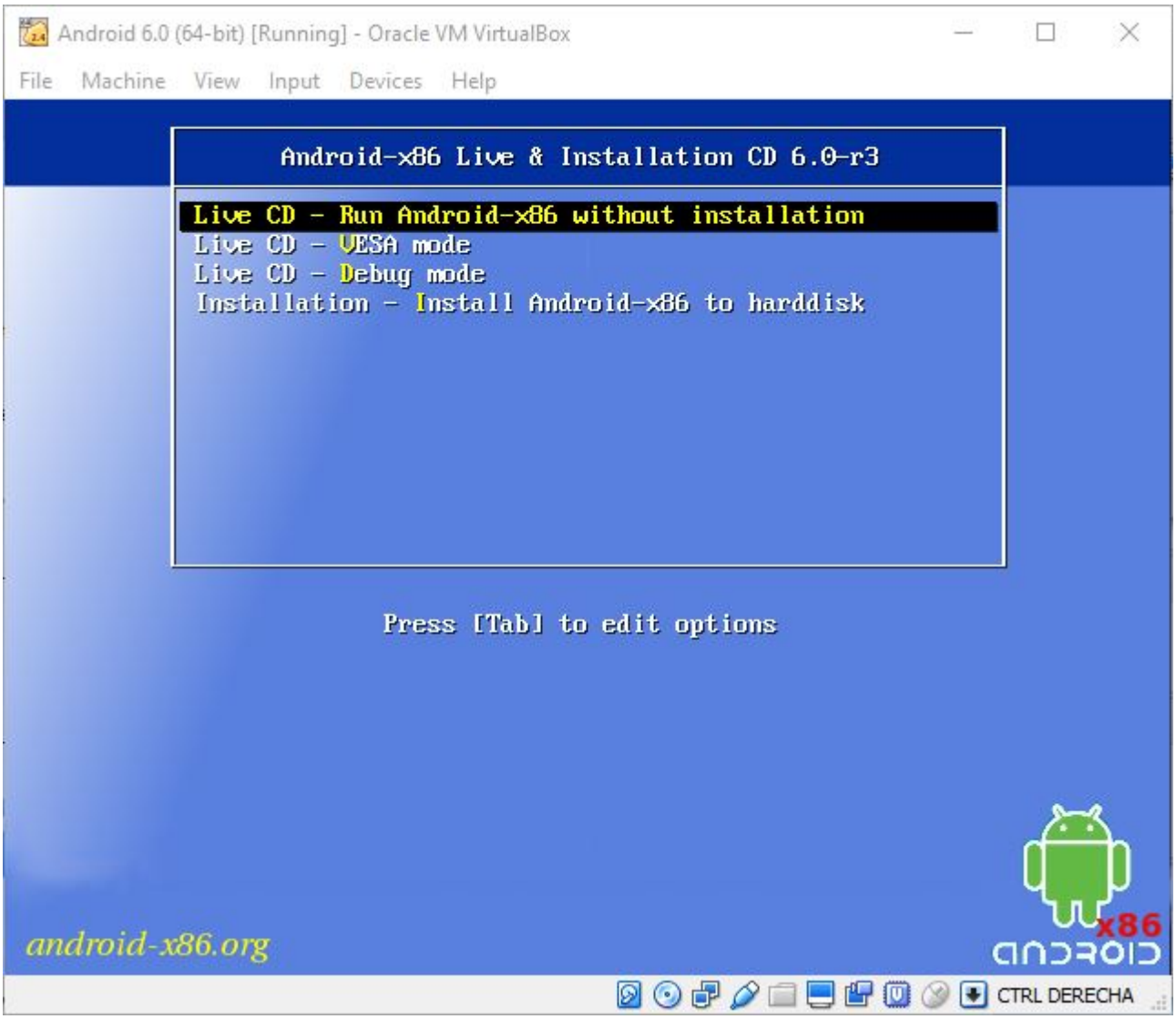
"w"パーティションにきむ

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

バーチャルマシンをすると、GRUBメニューがされ、カーネルのブートラインをできるようになります。つまり、 `DATA=sda1 SDCARD=sda2`オプションをして、sdcardまたはデータパーティションをできます。

パーティションへのインストール

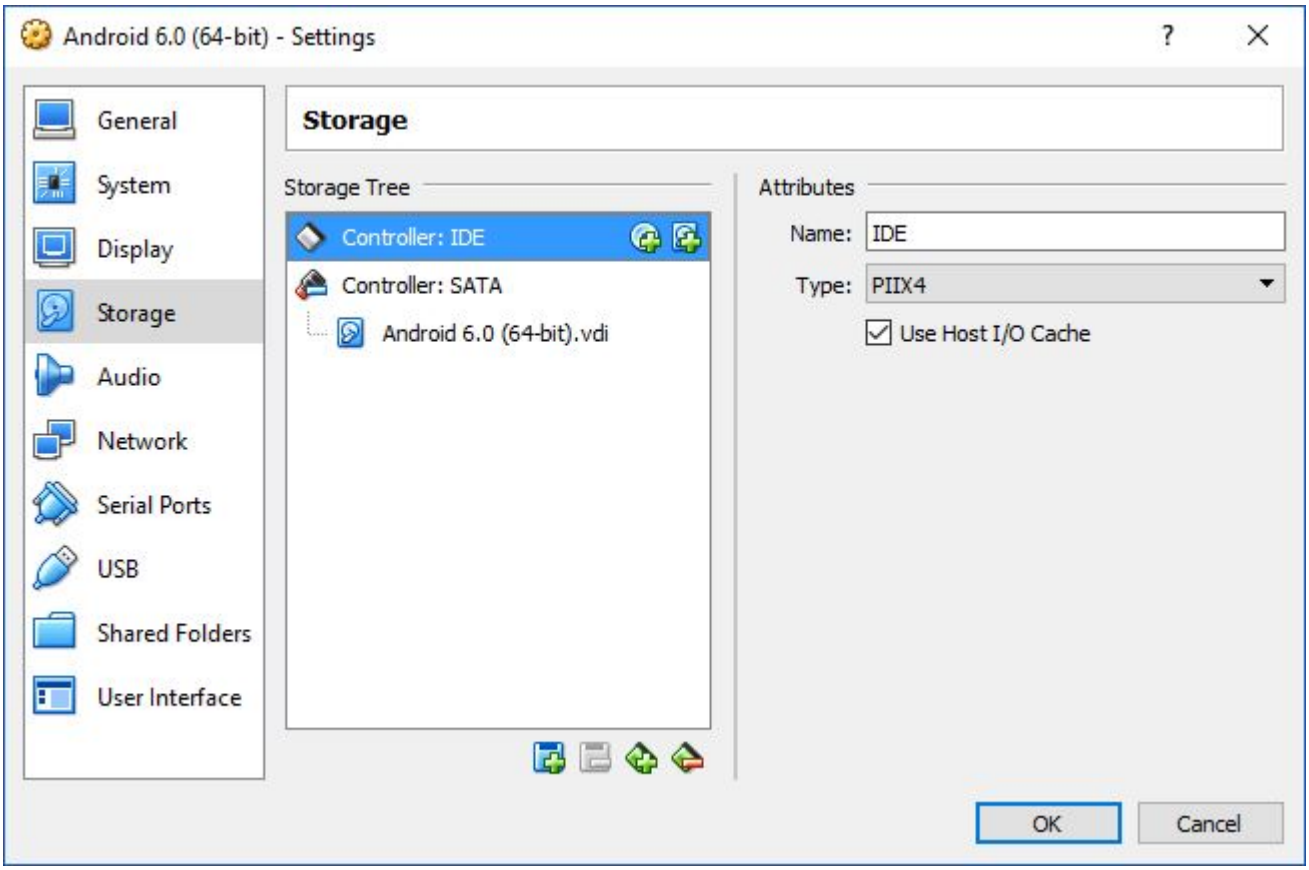
したハードドライブをして、アンドロイド-X86イメージをオプティカルドライブとしてします。



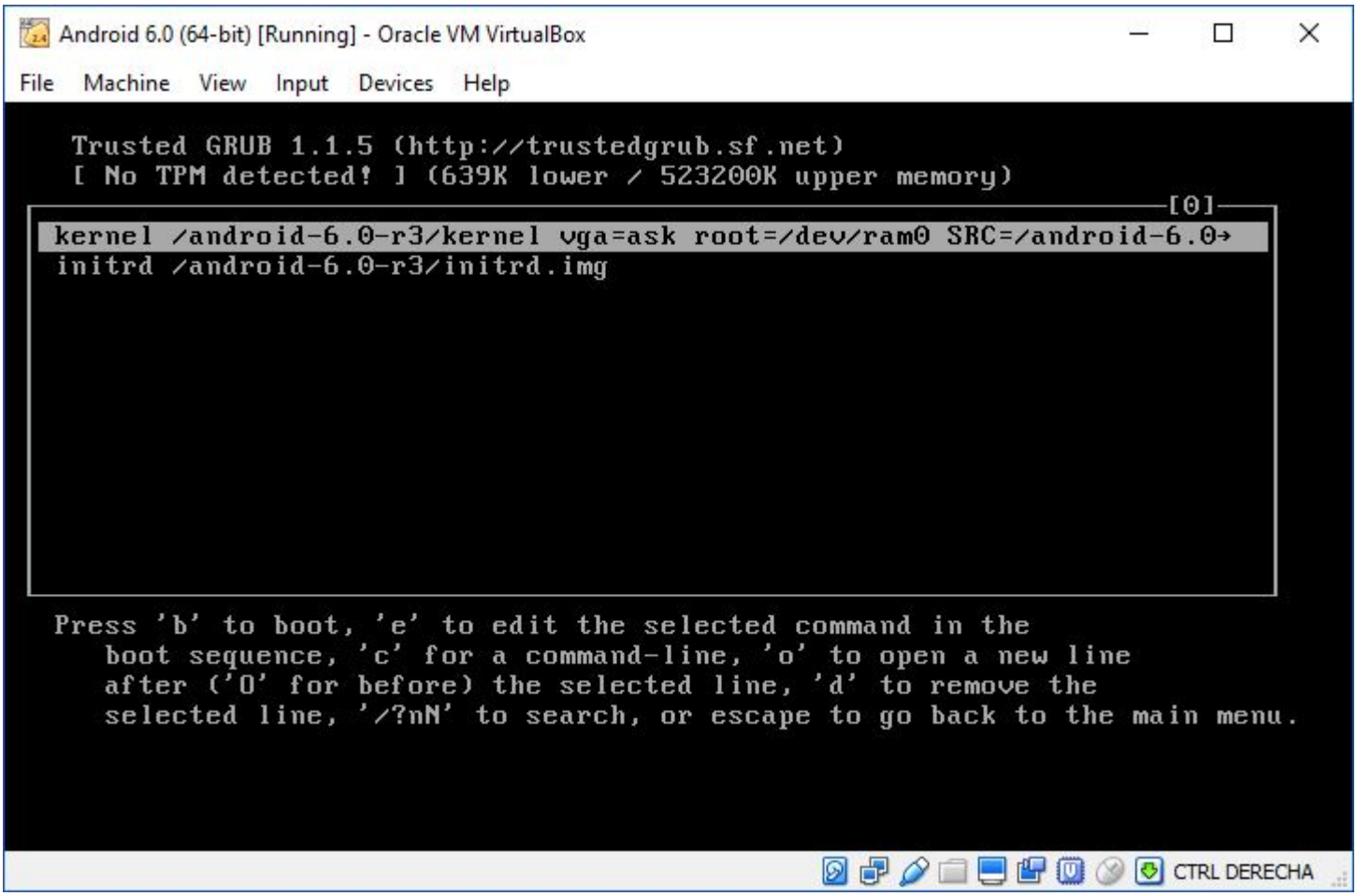
Live CDのオプションで「インストール - Androidをハードディスクにインストールする」をし、

sda1パーティションをしてandroidをインストールすると、grubがインストールされます。

マシンをしますが、ハードドライブからできるように、イメージがオプティカルドライブにないことをしてください。



grubメニューでは、 "Android-x86 6.0-r3"オプションのようにカーネルをするがありますので、eをしてください。

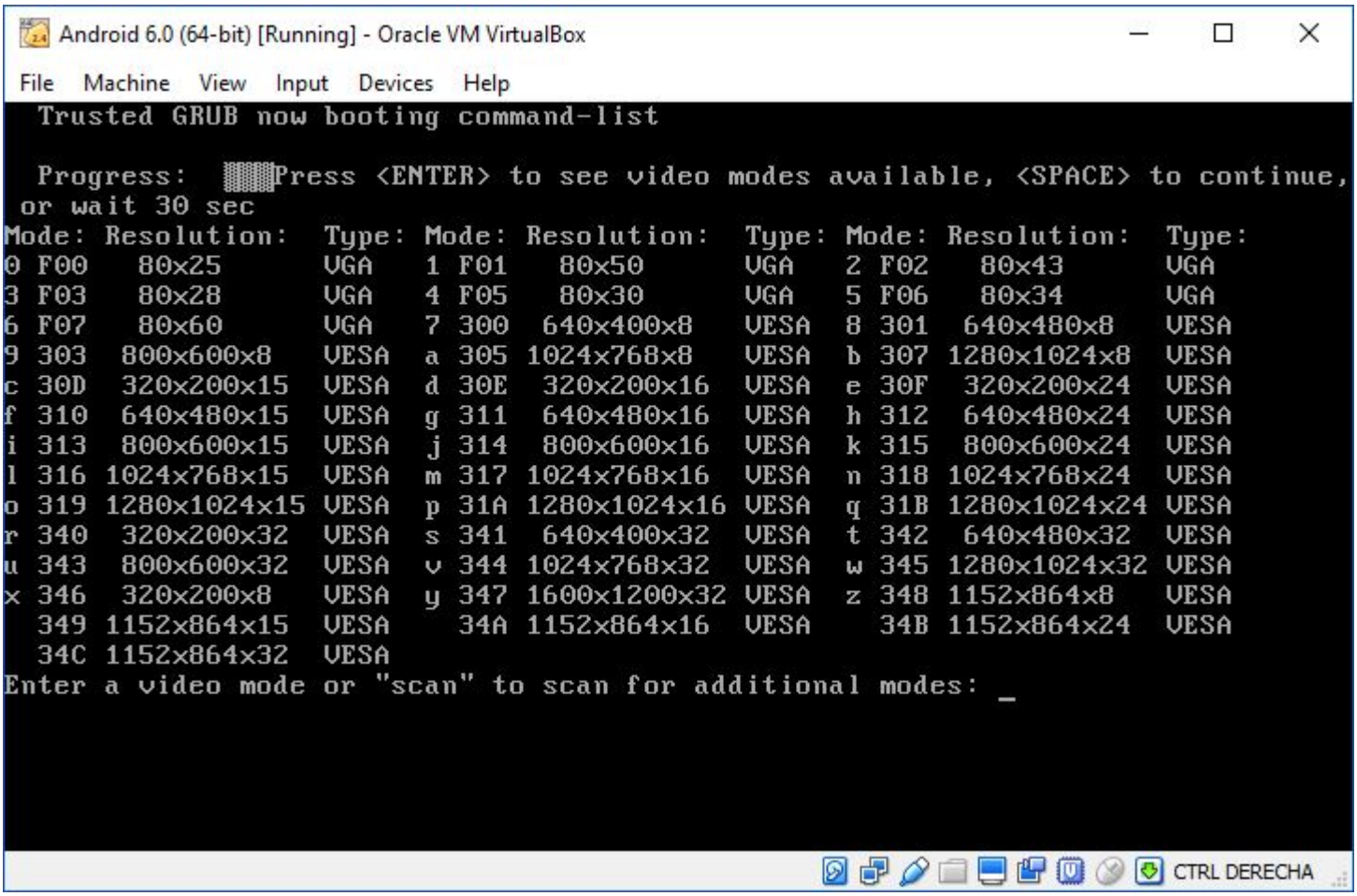


に、 "quiet"を "vga = ask"にきえ、オプション "SDCARD = sda2"をします。

の、カ―ネルのはにのようになります。

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

bをしてすると、ENTERをすとサイズをできます `vga=ask`オプション



インストールウィザードがしたら、をします。はとスペインをぶことができました。

オンラインでAndroid-x86 in VirtualBoxをむ https://riptutorial.com/ja/android/topic/9903/android-x86-in-virtualbox

18: Androidアーキテクチャのコンポーネント

き

Android Architecture Componentsは、で、テストで、なアプリケーションをするのにつしいライブラリのコレクションです。なは、ライフサイクル、ViewModel、LiveData、ルームです。

Examples

アーキテクチャコンポーネントの

プロジェクト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'
}
```

アプリケーションの

```
// 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"
```

AppCompatActivityでのライフサイクルの

このアクティビティからアクティビティをする

```
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;
    }
}
```

LiveDataトランスフォーメーションによるViewModel

```
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;
    }
}
```

UIのどこかに

```
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);
        }
}
```

の

ルームには、データベースクラス、DAOクラス、Entityクラス、およびMigrationクラスの4つのがです **DDL** メソッドのみできます 。

エンティティクラス

```
// 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;
}
```

DAOクラス

```
@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);
}
```

データベースクラス

```
// register your entities and DAOs
@Database(entities = {VideoItem.class, VideoTag.class}, version = 2)
public abstract class ContentDatabase extends RoomDatabase {
    public abstract VideoDao videoDao();
}
```

```
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() {
    }
}
```

アプリケーションクラスでするか、Daggerでする

```
ContentDatabase provideContentDatabase() {
    return Room.databaseBuilder(context, ContentDatabase.class, "data.db")
            .addMigrations(Migrations.ALL).build();
}
```

あなたのリポジトリをく

```
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);
    }
}
```

ViewModelでの

```
ContentRepository contentRepository = ...;
contentRepository.getVideoByTag(tag, limit);
```

カスタムLiveData

カスタムロジックがなは、カスタムLiveDataをすることができます。
データをするがあるのみカスタムクラスをしないでくださいTransformationsクラスを

```
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
```

```
        }
    };

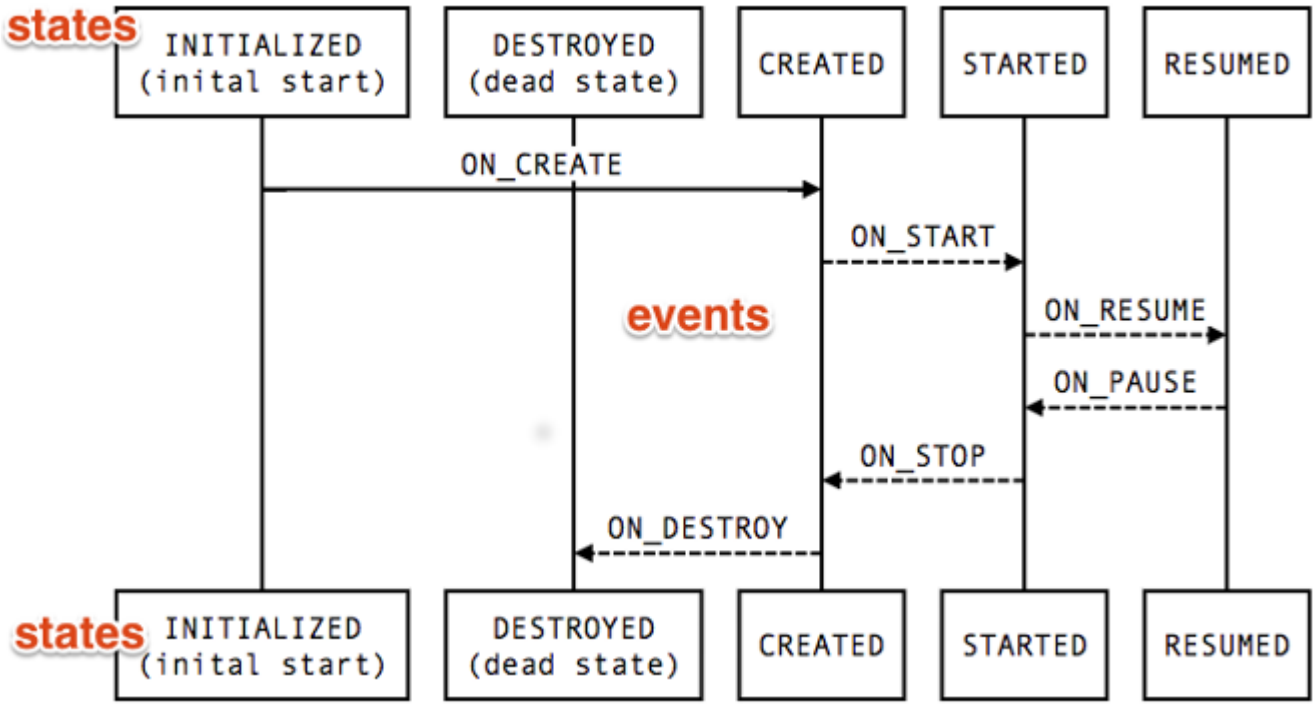
    public LocationLiveData(Context context) {
        locationManager = (LocationManager)
context.getSystemService(Context.LOCATION_SERVICE);
    }

    @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);
    }
}
```

カスタムライフサイクルコンポーネント

UIコンポーネントのライフサイクルは、のようにされました。



ライフサイクルのにされるコンポーネントをすることができます。

```
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
    }
}
```

オンラインでAndroidアーキテクチャのコンポーネントをむ
https://riptutorial.com/ja/android/topic/10872/androidアーキテクチャのコンポーネント

19: Androidアプリケーションののライブラリをする

Examples

ライブラリプロジェクトの

ライブラリをするには、 `File -> New -> New Module -> Android Library`をするがあります。これにより、なライブラリプロジェクトがされます。

それがしたら、のようにされたプロジェクトがです。

```
[project root directory]
    [library root directory]
    [gradle]
    build.gradle //project level
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    settings.gradle //this is important!
```

`settings.gradle`ファイルには、がまれているがあります。

```
include ':[library root directory]'
```

あなたの`[library root directory]`にはがまれているがあります

```
[libs]
[src]
   [main]
      [java]
         [library package]
   [test]
      [java]
         [library package]
build.gradle //"app"-level
proguard-rules.pro
```

あなたの "app"レベルの`build.gradle`ファイルには、のものがまれているがあります

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

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.2"

    defaultConfig {
        minSdkVersion 14
```

```
        targetSdkVersion 23
    }
}
```

それで、あなたのプロジェクトはうまくいくはずです

プロジェクトでのライブラリのモジュールとしての

ライブラリをするには、のにとしてライブラリをめるがあります。

```
compile project(':[library root directory]')
```

Jitpack.ioでなライブラリをする

ライブラリをするには、のをします。

1. GitHubアカウントをします。
2. ライブラリプロジェクトをむGitリポジトリをします。
3. のコードをして、ライブラリプロジェクトの`build.gradle`ファイルをします。

```
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
}
```

   のをGitHubにコミット/プッシュすることをしてください。

4. Githubののコードからリリースをします。

5. あなたのコードに`gradlew install`をしてください。
6. あなたのライブラリは、のによってになりました

```
compile 'com.github.[YourUser]:[github repository name]:[release tag]'
```

オンラインでAndroidアプリケーションののライブラリをするをむ
https://riptutorial.com/ja/android/topic/4118/androidアプリケーションののライブラリをする

20: Androidオーセンティケータ

Examples

なアカウントサービス

Androidアカウントオーセンティケータシステムをして、クライアントをリモートサーバーでることができます。 3つのがです。

- サービスは、 `android.accounts.AccountAuthenticator`によってトリガーされます。その`onBind`メソッドは、 `AbstractAccountAuthenticator`サブクラスをすがあり`AbstractAccountAuthenticator`。
- ユーザーにのをすアクティビティログインアクティビティ
- アカウントをするxmlリソースファイル

1.サービス

AndroidManifest.xmlにのをします。

```
<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" />
```

マニフェストファイルでサービスをします。

```
<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>
```

`android.accounts.AccountAuthenticator`は`intent-filter`タグにまれています。 `meta-data`タグにはxmlリソースここでは`authenticator`というがされてい`meta-data`。

サービスクラス

```
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. xml リソース

```
<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" />
```

`android:label`をりてたり、しているドロアブルをりてたりしないでください。なしでクラッシュします。

3. AbstractAccountAuthenticator クラスをします。

```
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;
    }
}
```

`AbstractAccountAuthenticator`クラスの`addAccount()`メソッドは、のの[アカウントの]からアカウントをするときにびされるため、です。 `AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE`は、したユーザーでアカウントキーをすためになAccountAuthenticatorResponseオブジェクトをむため、です。

オンラインでAndroidオーセンティケータをむ https://riptutorial.com/ja/android/topic/6759/androidオーセンティケータ

21: Androidカーネルの

Examples

RAM

、Androidは512MBのRAMをしたデバイスをサポートしています。このドキュメントは、OEMがメモリのためにAndroid 4.4をおよびするのにつものです。これらののいくつかは、のリリースにもできるほどです。

RAMデバイスフラグをにする

アプリケーションがメモリデバイスでしないのメモリをオフにするがあるかどうかをするために、ActivityManager.isLowRamDeviceというしいAPIをしています。

512MBデバイスの、このAPIは "true"をすことがされます。これは、デバイスのmakefileののシステムプロパティでにすることができます。

```
PRODUCT_PROPERTY_OVERRIDES += ro.config.low_ram=true
```

JITをにする

システムのJITメモリは、のアプリケーションのとそれらのアプリケーションのコードフットプリントにします。 JITはコードキャッシュサイズをし、にじてそののページにアクセスします。 JITのコストは、なシステムで3Mから6Mのです。

なアプリケーションは、コードキャッシュをかなりくデフォルトでは1Mであるにするがあります。して、JITキャッシュのはアプリあたり100K200Kバイトのでされます。キャッシュのサイズをさくするとメモリになりともちますが、さすぎるとJITがスラッシングモードになります。にメモリがないデバイスでは、JITをににすることをおめします。

これは、のmakefileにのをすることでできます。

```
PRODUCT_PROPERTY_OVERRIDES += dalvik.vm.jit.codecachesize=0
```

CPUガバナーをする

CPUガバナは1 Cファイルで、kernel_source / drivers / cpufreq /にありますcpufreq_smartass2.c。あなたはガバナをつけることにがありますあなたのデバイスのためののカーネルレポをてください。しかし、このファイルをカーネルににコールしてコンパイルするには、のをえるがあります

1. ガバナーファイルcpufreq_govname.cをコピーし、kernel_source / drivers / cpufreqをしてりけます。

2. カーネルをするときにKconfigこれはメニューのレイアウトのインターフェースですをき、あなたのにします。ガバナーのをすることでそれをうことができます。

```
config CPU_FREQ_GOV_GOVNAMEHERE
tristate "'gov_name_lowercase' cpufreq governor"
depends on CPU_FREQ
help
governor' - a custom governor!
```

たとえば、smartassV2の。

```
config CPU_FREQ_GOV_SMARTASS2
 tristate "'smartassV2' cpufreq governor"
 depends on CPU_FREQ
 help
 'smartassV2' - a "smart" optimized governor!
```

をするに、ガバナーがデフォルトガバナーとしてばれるもしなければなりません。

```
 config CPU_FREQ_DEFAULT_GOV_GOVNAMEHERE
 bool "gov_name_lowercase"
 select CPU_FREQ_GOV_GOVNAMEHERE
 help
 Use the CPUFreq governor 'govname' as default.
```

たとえば、smartassV2の。

```
config CPU_FREQ_DEFAULT_GOV_SMARTASS2
 bool "smartass2"
 select CPU_FREQ_GOV_SMARTASS2
 help
 Use the CPUFreq governor 'smartassV2' as default.
```

- それをくためのなをつけることができないのですか `"CPU_FREQ_GOV_CONSERVATIVE"`し、そのにコードをきます。じことが`"CPU_FREQ_DEFAULT_GOV_CONSERVATIVE"`

Kconfigがしたので、ファイルをしてじることができます。

3. まだ`/drivers/cpufreq`フォルダにあるは、Makefileをきます。 Makefileに、CPU Governorにするをします。えば

```
obj-$(CONFIG_CPU_FREQ_GOV_SMARTASS2)    += cpufreq_smartass2.o
```

ネイティブCファイルではなく、OファイルをびしてくださいこれはコンパイルされたCファイルです。ファイルをします。

4. `kernel_source/includes/linux`。 `cpufreq.h`きます。のような`cpufreq.h`されるまでにスクールします。

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_ONDEMAND)
```

```
extern struct cpufreq_governor cpufreq_gov_ondemand;
#define CPUFREQ_DEFAULT_GOVERNOR    (&amp;cpufreq_gov_ondemand)
```

のCPUガバナーもそこにされている

に、したCPUガバナをしてエントリをします。

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_SMARTASS2)
 extern struct cpufreq_governor cpufreq_gov_smartass2;
 #define CPUFREQ_DEFAULT_GOVERNOR (&amp;cpufreq_gov_smartass2)
```

ファイルをしてじます。

CPUガバナのがしました。あなたがしたすべてのステップをやったとき、あなたはメニューからあなたをぶことができるはず`menuconfig`、 `xconfig`、 `gconfig`、 `nconfig`。 メニューにチェックインすると、カーネルにみまれます。

のとほぼじです smartassV2とLulzactiveガバナーコミットをしてください

I / Oスケジューラ

にじてしいI / Oスケジューラをすることによって、カーネルをすることができます。には、ガバナーとスケジューラはじです。それらはともシステムがどのようにすべきかをします。ただし、スケジューラのは、CPUのをき、すべてデータストリームにするものです。 I / Oスケジューラは、されているI / Oアクティビティのスケジュールをします。 *noop*や*cfq*などのなスケジューラはににしています。

I / Oスケジューラは*kernel_source / block*にあります。

1. I / Oスケジューラファイルたとえば、 *sio-iosched.c* をコピーし、 *kernel_source / block*をしてください 。スケジューラーファイルをそこにりけます。

2. *Kconfig.iosched*をいて*Kconfig*にあなたのをします。えば、 *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. に、デフォルトのオプションをのようにします。

   ```
   default "sio" if DEFAULT_SIO
   ```

   ファイルをします。

4. *kernel_source / block /*で *Makefile*をき、 *SIO*にのをするだけです

```
obj-$(CONFIG_IOSCHED_SIO)    += sio-iosched.o
```

ファイルをして、です I / Oスケジュ―ラがメニュ―でポップアップするはずです。

GitHubでののコミット Simple I / Oスケジュ―ラがされました 。

オンラインでAndroidカ―ネルのをむ https://riptutorial.com/ja/android/topic/9106/androidカ―ネルの

22: Androidカスタムの

Examples

あなたのマシンをビルドにえる

あなたがかをするに、あなたのマシンをビルドするができているがあります。このためには、たくさんのライブラリとモジュールをインストールするがあります。もされるLinuxディストリビューションはUbuntuです。このでは、Ubuntuになものすべてをインストールすることにをてます。

Javaのインストール

まず、のPersonal Package ArchivePPAを`sudo apt-add-repository ppa:openjdk-r/ppa sudo apt-add-repository ppa:openjdk-r/ppa`。

に、 `sudo apt-get update`してソースを`sudo apt-get update`ます。

ののインストール

なのはすべて、のコマンドでインストールできます。

```
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
```

のためのシステムの

すべてのがインストールされたので、のコマンドをしてのシステムをしましょう。

```
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
```

に、のコマンドでキャッシュとレポをします。

```
sudo install utils/repo /usr/bin/
sudo install utils/ccache /usr/bin/
```

Resurrection Remix OSのメンテナーのであるAkhil Narang *akhilnarang* がしたスクリプトをして、このをうこともできます。これらのスクリプトはGitHubにあります 。

オンラインでAndroidカスタムROMのをむ https://riptutorial.com/ja/android/topic/9212/androidカスタムromの

23: Androidサウンドとメディア

Examples

APIのとの> 19

ここには、とのテストみコードがあります.1919のすべてのAPIでします。

```
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);
                    }
```

ビデオ

```
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();
        }
```

SoundPoolでサウンドをする

```
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;
```

```
    }
}
```

オンラインでAndroidサウンドとメディアをむ https://riptutorial.com/ja/android/topic/4730/android-サウンドとメディア

24: Androidスタジオ

Examples

UIからログをフィルタリングする

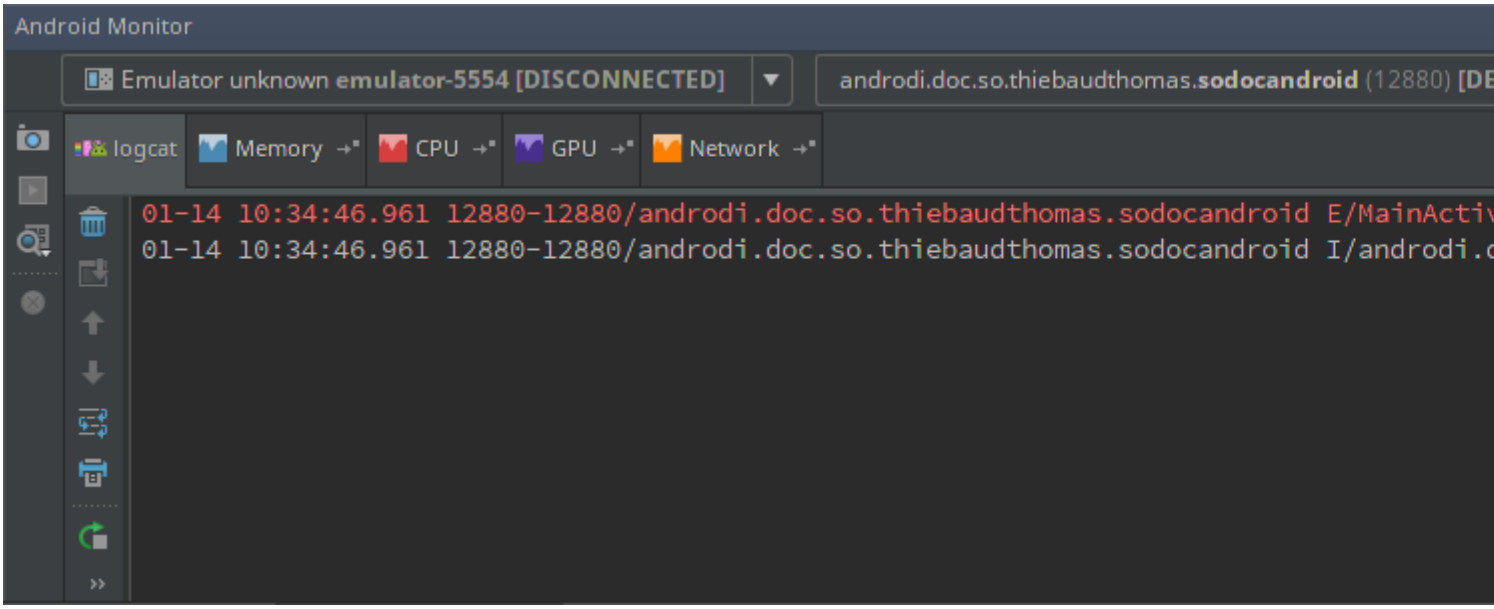
AndroidログはUIからフィルタリングできます。このコードをする

```
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");
    }
}
```

`TAG1|TAG2`とレベル`verbose`をすると

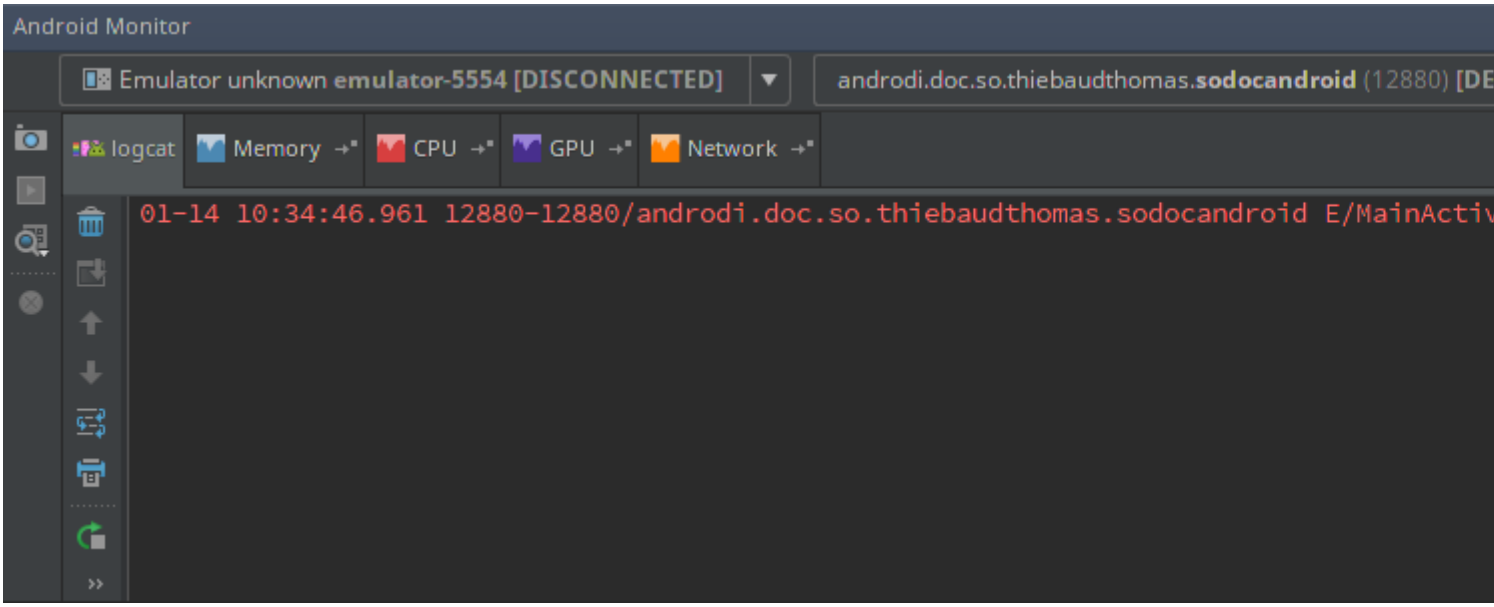
```
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log
from onCreate method with TAG1
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid
I/androdi.doc.so.thiebaudthomas.sodocandroid.MainActivity: Log from onCreate method with TAG2
```



レベルは、されたレベルのログをするようにできます。たとえば、 `verbose`レベルは`verbose, debug, info, warn, error and assert`ログをキャッチし`verbose, debug, info, warn, error and assert`。

じをして、レベルを`error`にすると、は

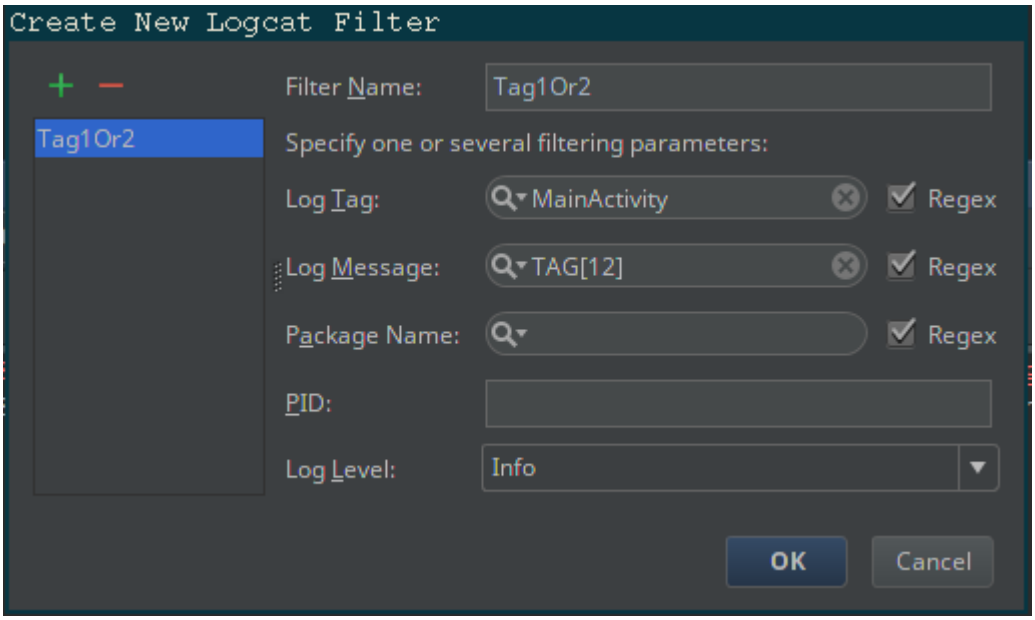
```
01-14 10:34:46.961 12880-12880/androdi.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log from onCreate method with TAG1
```



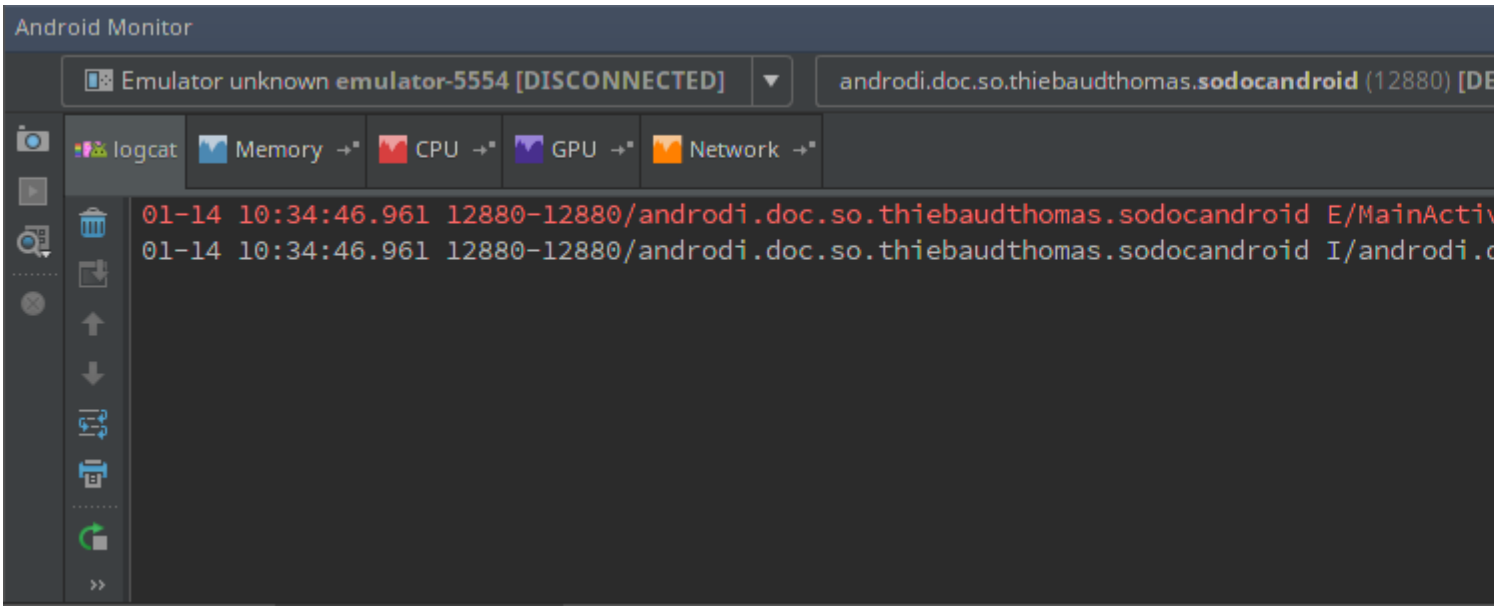
フィルタをする

カスタムフィルタをし、UIからすることができます。 `AndroidMonitor`タブで、のドロップダウン`Show only selected application`または`No filters` をクリックし、[`Edit filter configuration`をし`Edit filter configuration`。

あなたがむフィルター—をしてください

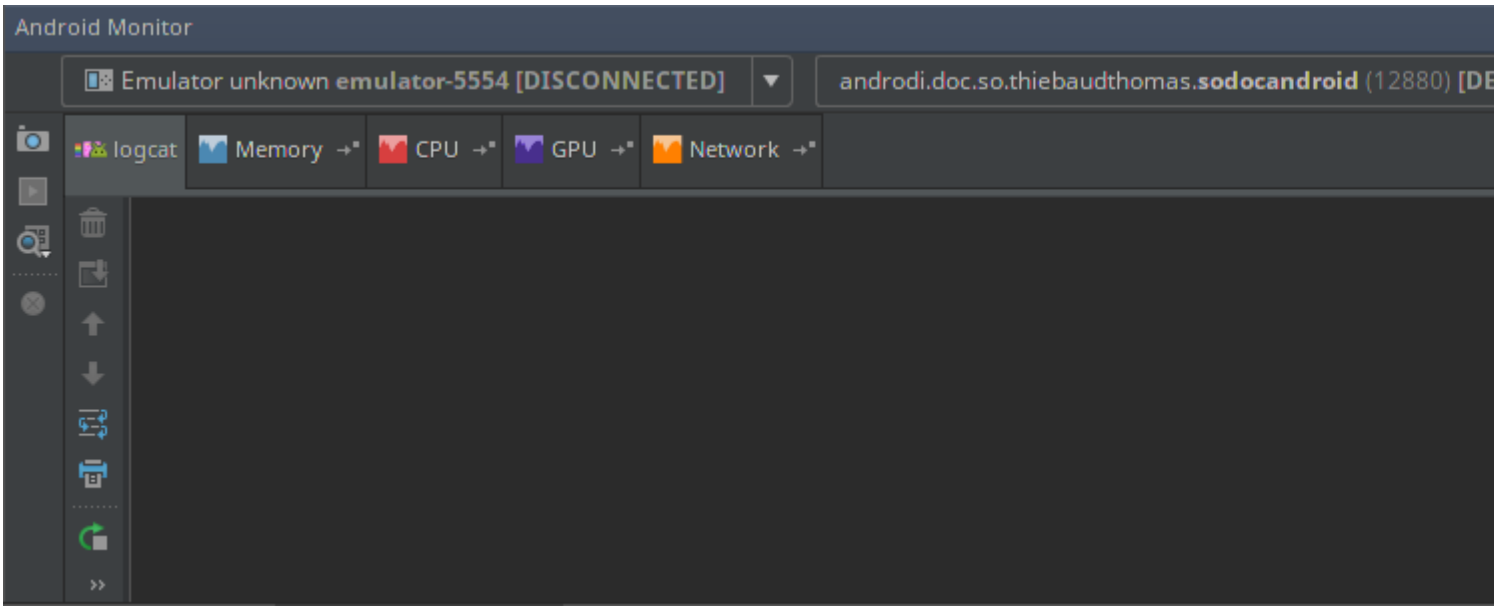


それをしてくださいじドロップダウンからできます

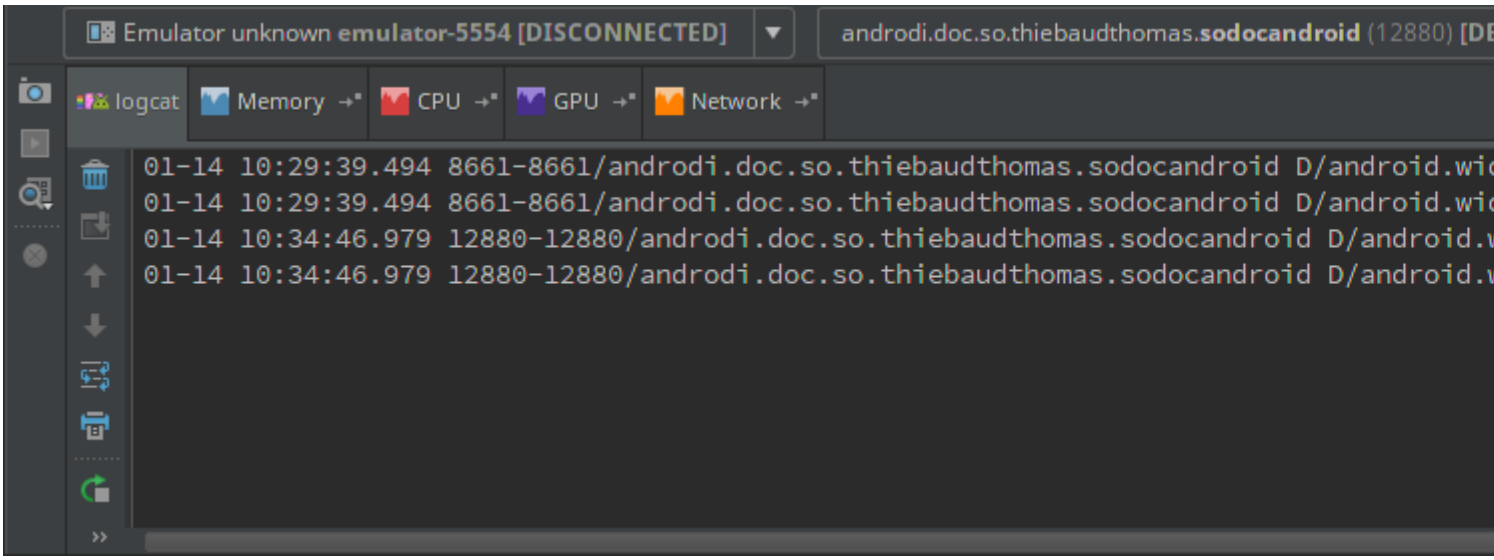


フィルターバーにをすると、アンドロイドスタジオはフィルターとのをします。

とフィルタのにがありません



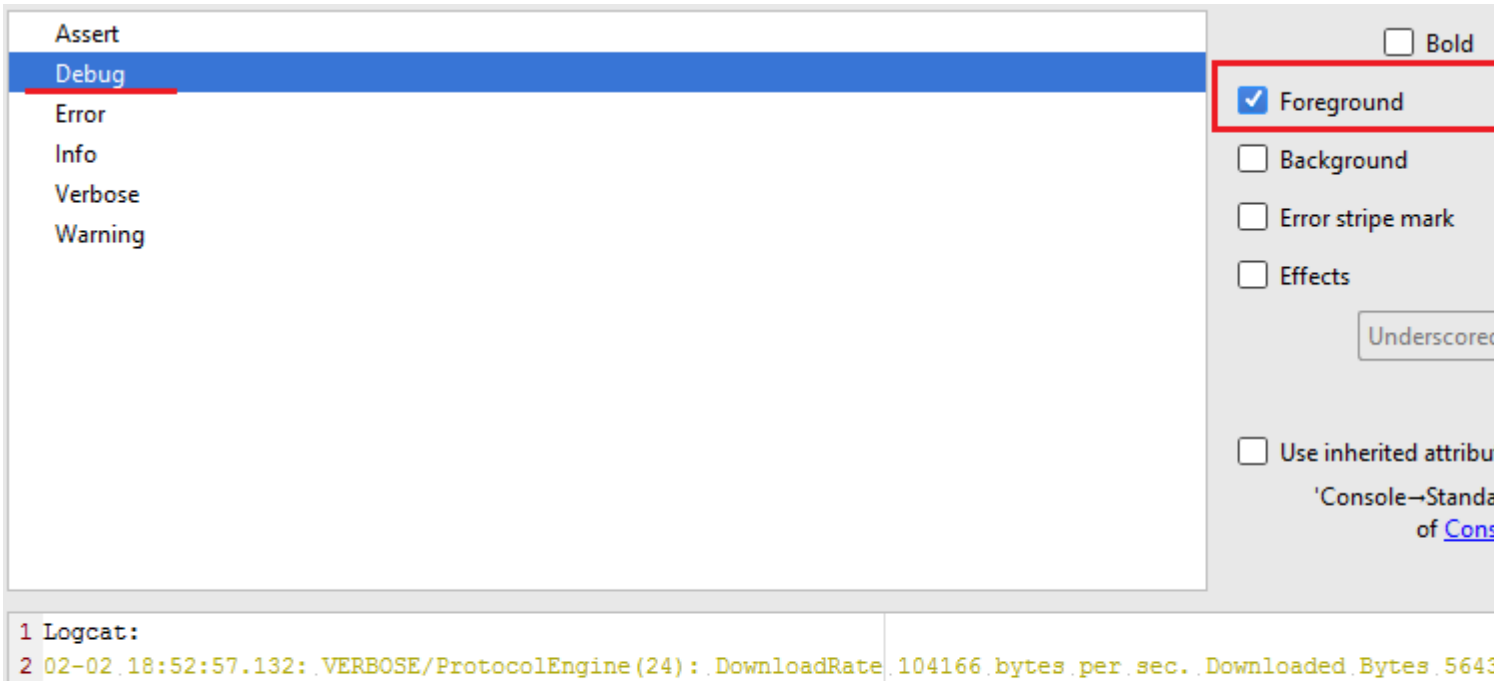
フィルタなしでは、いくつかのがあります



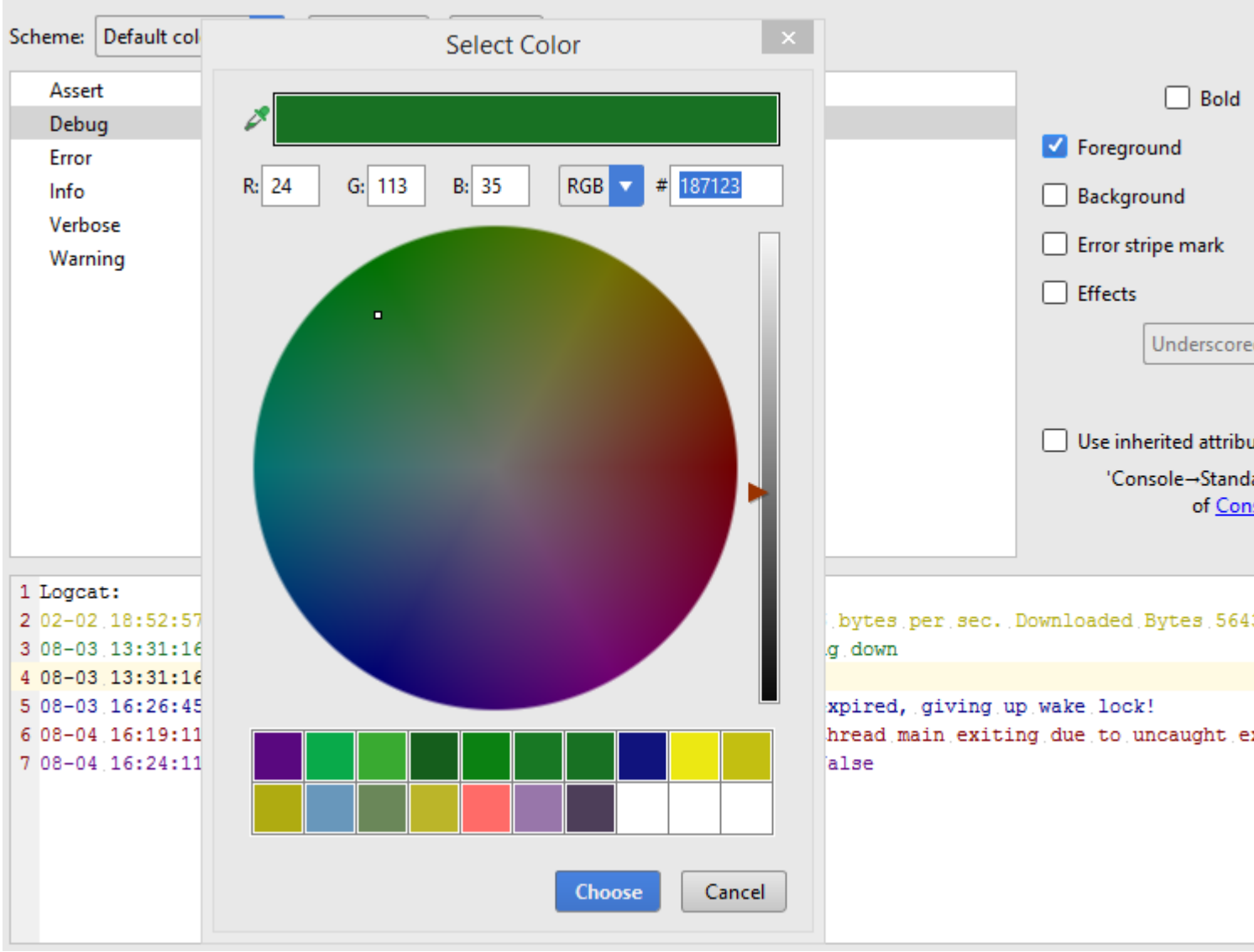
メッセージのについたlogcatメッセージのカスタム

ファイル - > - >エディタ - >とフォント - > Android Logcat

にじてをします。

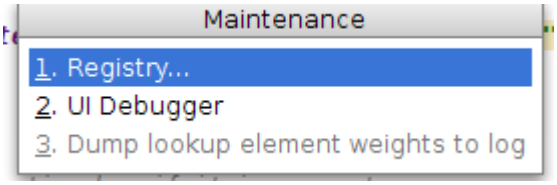


なをしてください

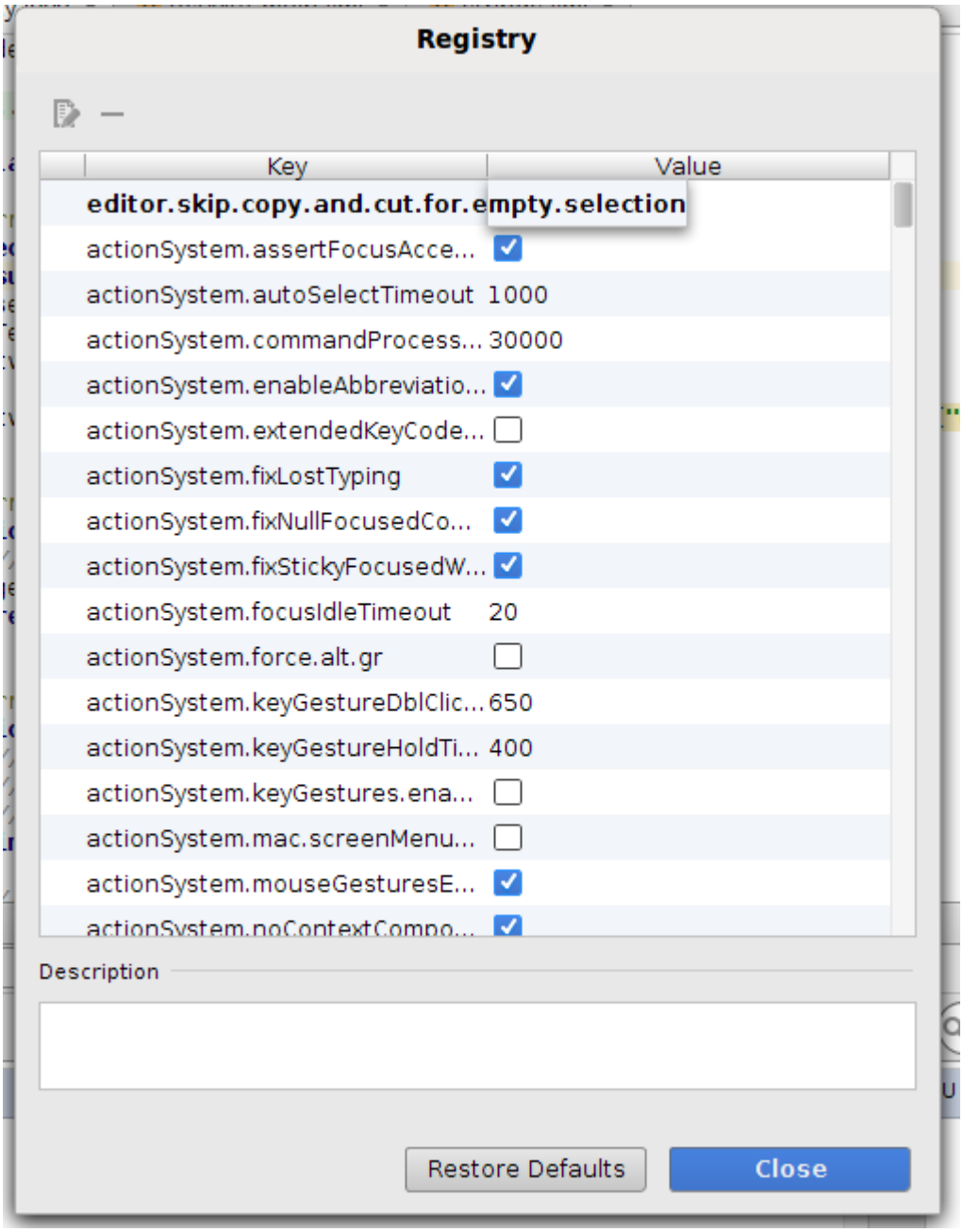


コピーをIにする

`ctrl + alt + shift + /` MacOS `cmd + alt + shift + /` は、のダイアログをするはずです



あなたがる`Registry`をクリックする



/にするキーはのとおりです。

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

`Linux Ubuntu`と`MacOS`テストされています。

Android Studioのなショートカット

は、よりな/なショートカットのいくつかです。

これらは、デフォルトのIntelliJショートカットマップにづいています。 `File -> Settings -> Keymap -> <Choose Eclipse/Visual Studio/etc from Keymaps dropdown>`キーマップ`File -> Settings -> Keymap -> <Choose Eclipse/Visual Studio/etc from Keymaps dropdown>`をして、のなIDEショートカットマップにりえることができます。

アクション	ショーートカット
フォーマットコード	CTRL + ALT + L
されていないメソッドをする	CTRL + I
logcatを	ALT + 6
ビルド	CTRL + F9
ビルドと	CTRL + F10
	CTRL + F
プロジェクトで	CTRL + SHIFT + F
と	CTRL + R
プロジェクトでのと	CTRL + SHIFT + R
オーバーライドメソッド	CTRL + O
プロジェクトをする	ALT + 1
プロジェクトをす - logcat	シフト + ESC
すべてりたたむ	Ctrl + Shift + NumPad +
デバッグポイントをする	Ctrl + Shift + F8
すべて	Ctrl + Shift + NumPad -
をく	ALT + □
ターゲットをプロジェクトビューでのファイルをく	ALT + F1 → ENTER
どこからでも	SHIFT → SHIFT ダブルシフト
コード|サラウンドバック	CTRL → ALT + T
メソッドフォームのされたコードをする	ALT + CTRL

リファクタリング

アクション	ショーートカット
リファクタリングののすべてのなリファクタアクションのメニュー/ピッカー	Mac CTRL + T - Win / Linux CTRL + ALT + T

アクション	ショートカット
リネーム	シフト + F6
メソッド	Mac CMD + ALT + M - Win / Linux CTRL + ALT + M
パラメータの	Mac CMD + ALT + P - Win / Linux CTRL + ALT + P
の	Mac CMD + ALT + V - Win / Linux CTRL + ALT + V

Androidスタジオパフォーマンスのヒントをする

オフラインをにする

1. 「ファイル」 -> 「」をクリックします。 "gradle"をし、 Offline workボックスをクリックします。
2. コンパイラ Gradleすぐにあるじダイアログに--offlineし、 Command-line Optionsテキストボックスに--offlineをします。

グレードパフォーマンスをさせる

gradle.propertiesファイルにの2のコードをします。

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

studio.vmoptionsファイルの**-Xmx**と**-Xms**をきくする

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

USERPROFILE。{FOLDER_NAME} \ studio.exe.vmoptionsおよび/または
USERPROFILE。{FOLDER_NAME} \ studio64.exe.vmoptions

マック

/ライブラリ/ Preferences / {FOLDER_NAME} /studio.vmoptions

Linux

/。{FOLDER_NAME} /studio.vmoptions/。{FOLDER_NAME} /studio64.vmoptions

Android Studioのセットアップ

システム

- Microsoft®Windows®8/7 / Vista / 200332または64ビット
- Mac®OSX®10.8.5、10.9Mavericks
- GNOMEまたはKDEデスクトップ

インストール

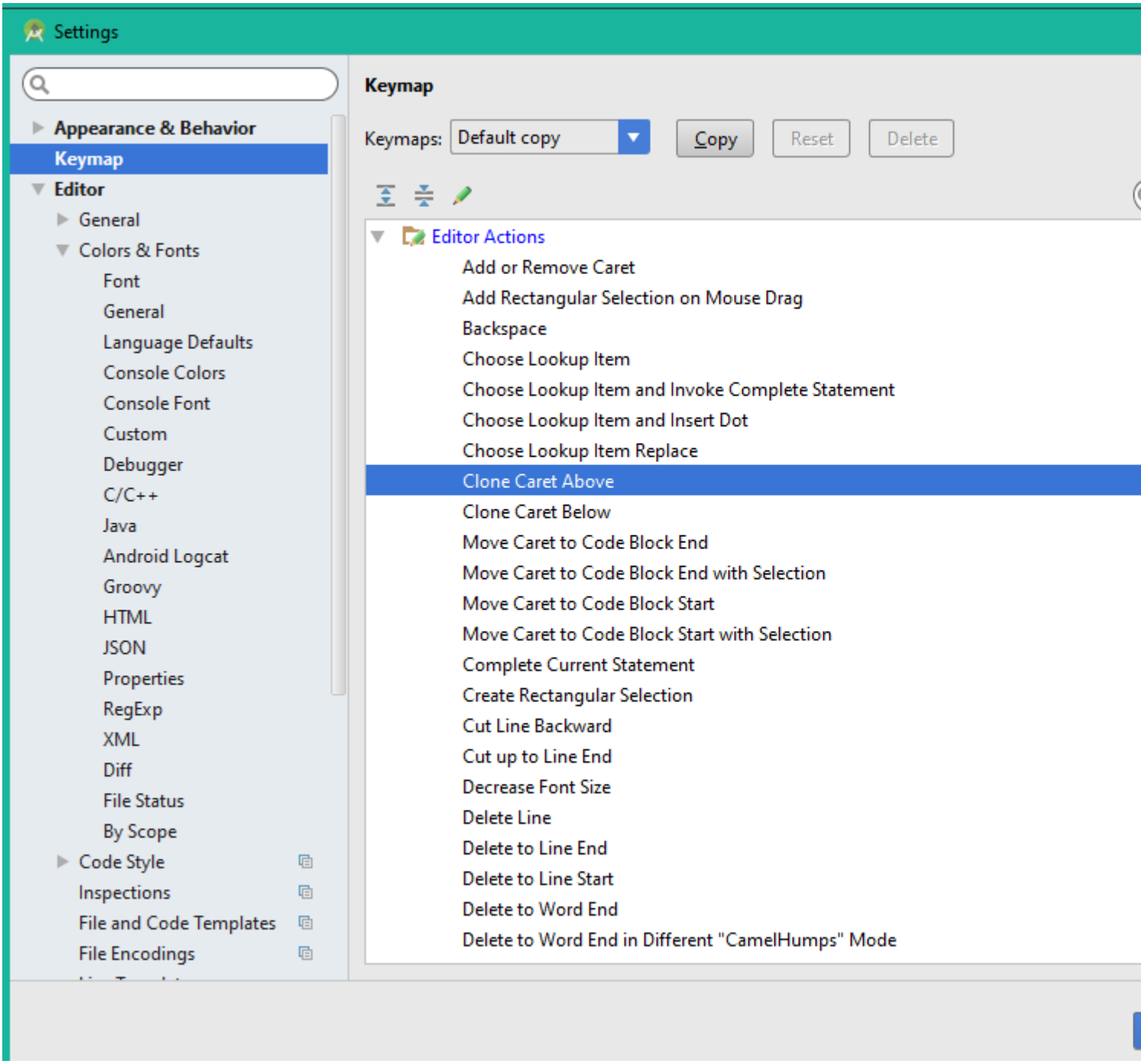
1. JDKJava Development Kitバージョン8をダウンロードしてインストールする
2. Android Studioをダウンロード
3. `Android Studio.exe`し、JDKパスをしてのSDKをダウンロードします

Linux

1. JDKJava Development Kitバージョン8をダウンロードしてインストールする
2. Android Studioをダウンロード
3. zipファイルをする
4. オープンターミナル、されたフォルダへのcd、binへの`cd android-studio/bin` `cd android-studio/bin`
5. ./studio.shをする

Android Studioでのショートカットのと

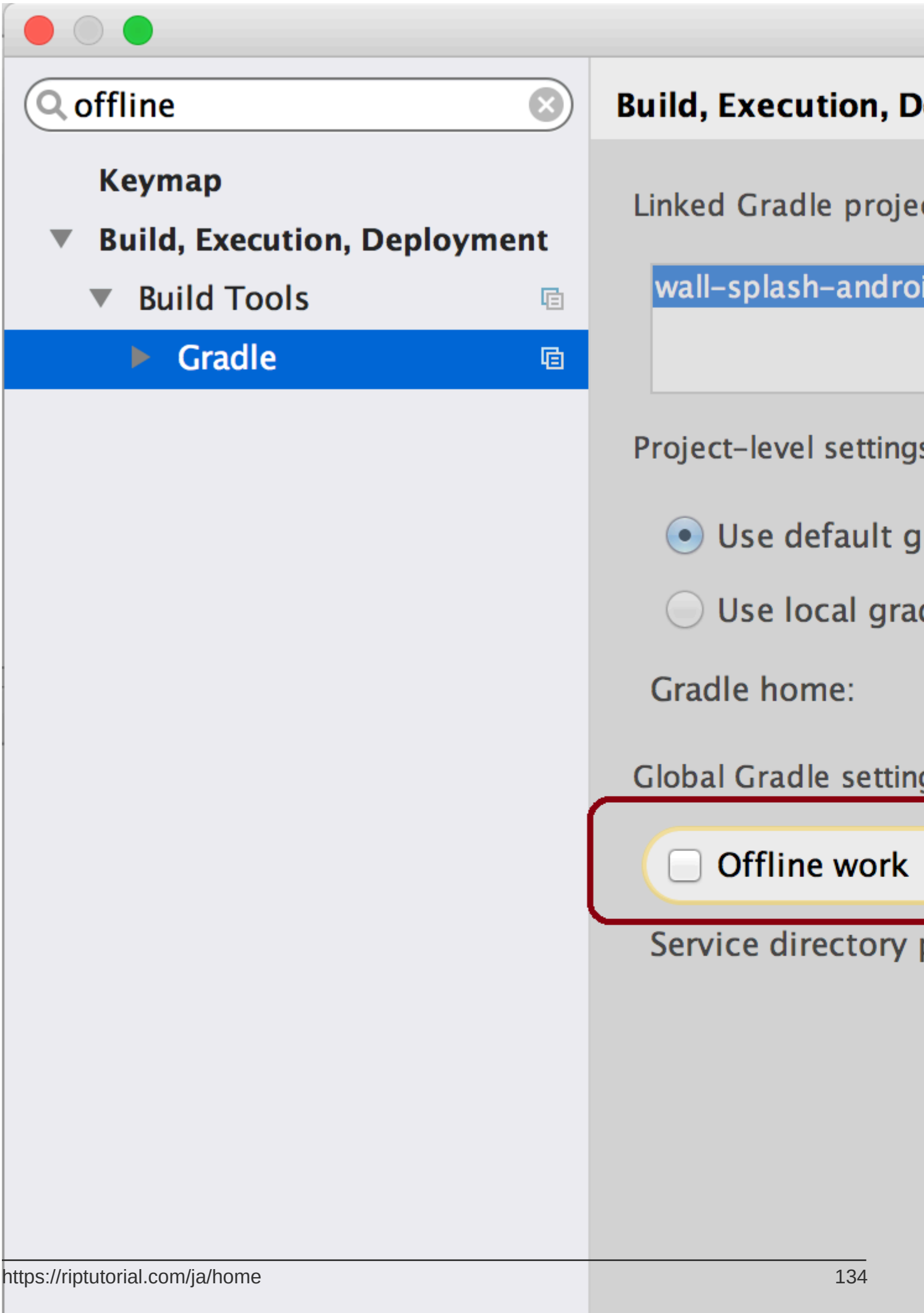
>>キーマップにすると、すべての`Editor Actions`とショートカットがされたウィンドウがポップアップします。 `Editor Actions`はショートカットがありません。それをクリックし、しいショートカットをしてください。
のをしてください



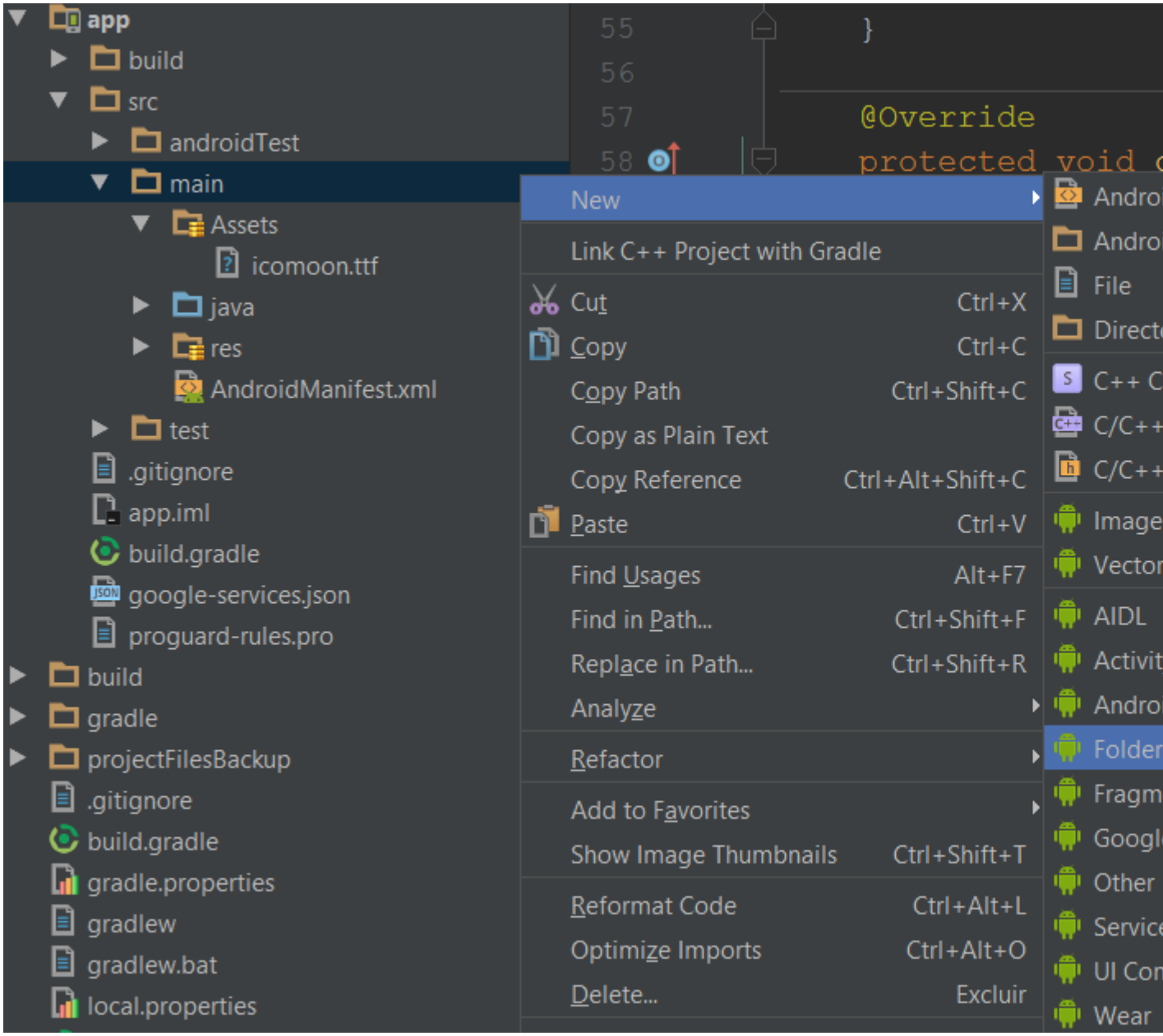
Gradleビルドプロジェクトはにかかる

Android Studio - > - > **Gradle** - > オフラインをし、Androidスタジオをします。

リファレンススクリーンショット


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

- アセットフォルダはMAINフォルダのにあり、RESフォルダとじシンボルでされます。
- このでは、フォントファイルをいています。



オンラインでAndroidスタジオをむ https://riptutorial.com/ja/android/topic/107/androidスタジオ

25: AndroidでClingライブラリをしたポートマッピング

Examples

AndroidプロジェクトへのClingサポートの

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'

}
```

NATポートのマッピング

```
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;
}
```

オンラインでAndroidでClingライブラリをしたポートマッピングをむ
https://riptutorial.com/ja/android/topic/6208/androidでclingライブラリをしたポートマッピング

26: Androidでのシェイクイベントの

Examples

Androidのでのシェイク

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

によるれの

*Seismic*はSquareのAndroidデバイスのライブラリです。それをするには、それによってされたれイベントをきめるだけです。

```
@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();
}
```

なるをするには、 `sensitivity` `SENSITIVITY_LIGHT` 、 `SENSITIVITY_MEDIUM` 、 `SENSITIVITY_HARD`またはそののなをして`sd.setSensitivity(sensitivity)`をします。のデフォルトは*11*から*15*のです。

インストール

```
compile 'com.squareup:seismic:1.0.2'
```

オンラインでAndroidでのシエイクイベントのをむ
https://riptutorial.com/ja/android/topic/4501/androidでのシエイクイベントの

27: Androidでのリソースによるローカライゼーション

Examples

```
Currency currency = Currency.getInstance("USD");
NumberFormat format = NumberFormat.getCurrencyInstance();
format.setCurrency(currency);
format.format(10.00);
```

Androidアプリにをする

しいごとになるstrings.xmlファイルをするがあります。

1. resフォルダをクリックします。
2. → リソースファイルをします。
3. なからロケールをする
4. [へ]ボタン>>をクリックします。
5. を
6. ファイルにstrings.xmlというをけます。

strings.xml

```
<resources>
    <string name="app_name">Testing Application</string>
    <string name="hello">Hello World</string>
</resources>
```

strings.xmlhi

```
<resources>
    <string name="app_name">परीक्षण आवेदन</string>
    <string name="hello">नमस्ते दुनिया</string>
</resources>
```

プログラムでをする

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

のは、 *strings.xml*からされるテキストフィールドをします。たとえば、の2つのテキストビューがあるとします。

```
<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"/>
```

その、ロケールをした、 `app_name`と`hello`をつがそれにじてされます。

「res」フォルダのにあるリソースディレクトリのタイプ

タイプのリソースをローカライズするには、アンドロイドのプロジェクトにのホームがあります。に、 `\res`ディレクトリのにくことができるさまざまなディレクトリをします。これらのディレクトリにされているリソースタイプは、のでされています。

ディレクトリ	リソースタイプ
アニメーター/	プロパティアニメーションをするXMLファイル。
アニメ/	トウイーンアニメーションをするXMLファイル。 プロパティアニメーションもこのディレクトリにできますが、プロパティアニメーションでは2つのタイプをするためにアニメータ/ディレクトリがされます。
/	のリストをするXMLファイル。カラーステートリストリソースをしてください。
ドロアブル/	"ビット`Bitmap files - Nine-Patches (re-sizable bitmaps) - State lists - Shapes - Animation drawables - Other drawables - "`サイズな`Bitmap files - Nine-Patches (re-sizable bitmaps) - State lists - Shapes - Animation drawables - Other drawables - "`ビットマップファイル.png、.9.png、.jpg、.gif、またはのリソースサブタイプにコン

ディレクトリ	リソースタイプ
	パイルされたXMLファイル。 `Bitmap files - Nine-Patches (re-sizable bitmaps) - State lists - Shapes - Animation drawables - Other drawables - "`
ミップマップ/	なるランチャーアイコンのためのなファイル。ミップマップ/フォルダをしたランチャーアイコンののについては、「プロジェクトのの」をしてください。
レイアウト/	ユーザーインターフェイスのレイアウトをするXMLファイル。レイアウトリソースをしてください。
メニュー/	オプションメニュー、コンテキストメニュー、サブメニューなど、アプリケーションメニューをするXMLファイル。メニューリソースをしてください。
/	のでするのファイル。のInputStreamでこれらのリソースをくには、リソースID R.raw.filenameをしてResources.openRawResourceをびします。
	ただし、のファイルとファイルにアクセスするがあるは、いくつかのリソースをassets /ディレクトリres / raw /のわりににすることをすることもできます。アセットファイル/はリソースIDがえられていないため、AssetManagerをしてみることができます。
/	、、、スタイル、テーマなどのなをむXMLファイル
xml /	Resources.getXMLをびしてにみることができるのXMLファイル。ななど、さまざまなXMLファイルをここにするがあります。

"res"ディレクトリのフォルダのタイプと

`res`フォルダののリソースディレクトリのにげられているは、ののディレクトリにまれているさまざまなバリエーションを、`configuration-type`ごとになる`qualifier-values`をつけてサフィックスとしてつことが`configuration-type`。

``アンドロイドプロジェクトでにられるのがなる``ディレクトリのバリエーションの

- ドロアブル/
- drawable-en /
- drawable-fr-rCA /
- drawable-en-port /
- drawable-en-notouch-12key /

- drawable-port-ldpi /
- drawable-port-notouch-12key /

すべてのなるタイプとアンドロイドリソースののののなリスト

	の
MCCとMNC	
	mcc310
	mcc310-mnc004
	mcc208-mnc00
と	
	エン
	fr
	en-rUS
	fr-rFR
	fr-rCA
レイアウトの	ldrtl
	ldltr
smallestWidth	swdp
	sw320dp
	sw600dp
	sw720dp
な	wdp
	w720dp
	w1024dp
なさ	hdp
	h720dp

	の
	h1024dp
サイズ	さい
	×ラージ
のアスペクト	いです
	くない
ラウンドスクリーン	
	notround
のき	
UIモード	
	テレビ
	アプライアンスウォッチ
ナイトモード	
	はない
スクリーンピクセル**dpi**	ldpi
	mdpi
	hdpi
	xhdpi
	xxhdpi
	xxxhdpi
	nodpi
	tvdpi
	anydpi
タッチスクリーンタイプ	らないで

	の
キーボードの	キーをした
	れ
	keyssoft
なテキスト	ノーキー
	QWERTY
	12キー
ナビゲーションキーの	
	ナビゲート
タッチナビゲーション	ノンアブ
	dpad
	トラックボール
	ホイール
プラットフォームバージョン**API**レベル	
	v3
	v4
	v7

プログラムでロケールのアンドロイドアプリケーションをする

のでは、アプリケーションのリソースをローカライズするをしています。のでは、アプリケーションではなくアプリケーションでアプリケーションのロケールをするをします。アプリケーションロケールのみをするには、のロケールutilをします。

```
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();
    }
}
```

ユーザーがするロケールをApplicationクラスからします。

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

また、アクティビティをし、このアクティビティをのすべてのアクティビティにして、アプリケーションのロケールをのように1だけできるようにするがあります。

```
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);
    }

}
```

にロケールをコンストラクターでしてください。

これで、LocalizationActivityをのようにできます。

```
public class MainActivity extends LocalizationActivity {

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

    }
```

```
}
```

> プログラムのロケ―ルをプログラムでするは、ロケ―ルのをにするためにアクティビティをするがあります。このソリュ―ションでしくするためには、アプリケ―ションのにのからロケ―ルをしてください`android:name=".LocaleApp"` inあなたManifest.xml。

リントチェッカ―が、リリ―スビルドのをすことがあります。このようなをするには、のオプションをします。

ののをにするは、デフォルトのstring.xmlにのをします

```
<string name="developer" translatable="false">Developer Name</string>
```

のをするリソ―スファイルからすべてのしているをします。のように、ファイルのツ―ルののです。

```
<?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>
```

なをにするの

http://tools.android.com/recent/non-translatablestrings

すべきではないリソ―スがたくさんあるは、それらをdonottranslate.xmlというのファイルにくことができ、lintはそのすべてをなリソ―スとなします。

4

リソ―スファイルにロケ―ルをすることもできます

```
<resources
xmlns:tools="http://schemas.android.com/tools"
    tools:locale="en" tools:ignore="MissingTranslation">
```

また、app / build.gradleからlintのチェックがけているのをにすることもできます

```
lintOptions {

        disable 'MissingTranslation'
    }
```

オンラインでAndroidでのリソ―スによるロ―カライゼ―ションをむ
https://riptutorial.com/ja/android/topic/3345/androidでのリソ―スによるロ―カライゼ―ション

28: Androidでファイルをする

Examples

ファイルをする

```
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;}
```

オンラインでAndroidでファイルをするをむ https://riptutorial.com/ja/android/topic/3927/androidでファイルをする

29: AndroidのJava

き

Androidは、すべてのJava 7とプラットフォームバージョンによってなるJava 8のサブセットをサポートしています。このページでは、できるしい、それらをするためのプロジェクトのな、およびするのあるのについてします。

Examples

RetrolambdaをってJava 8のをサブセット

Retrolambdaをすると、ラムダ、メソッド、およびJava 7,6または5でのtry-with-resourcesステートメントでJava 8コードをできます。これは、Java 8コンパイルみバイトコードをしていJavaランタイムでできるようにすることでいます。

バックポートの

- ラムダはのクラスにすることによってバックポートされます。これには、ステートレスラムダにシングルトンインスタンスをしてりしオブジェクトをりてるのをすることがまれます。メソッドはにラムダのであり、じでバックポートされます。
- ターゲットのバイトコードのバージョンがJava 7の、`Throwable.addSuppressed`へのびしをすることで、Try-with-resourcesステートメントがバックポートされます。されたがりまれるのではなくログにされるようにするには、です。
- `Objects.requireNonNull`コールはへのびしにきえられ`Object.getClass`ターゲットバイトコードのバージョンは、Java 7 JDK 9のによってされたヌルチェックる`Objects.requireNonNull`のJDKバージョンがされるのにし、`Object.getClass`。
- オプション

    1. デフォルトメソッドは、メソッドとしてコンパニオンクラスインターフェイス+ "$"にデフォルトメソッドをコピーし、インターフェイスのデフォルトメソッドをメソッドにきえ、なメソッドをそのインターフェイスをするすべてのクラスにすることによってバックポートされます。
    2. メソッドをコンパニオンクラスインターフェイス+ "$"にし、すべてのメソッドびしをしてしいメソッドのをびすことで、インターフェイスのメソッドがバックポートされます。

の

- Java 8 APIをバックポートしない

- インターフェースのバックポートのデフォルトメソッドとメソッドには、すべてのバックポートされたインターフェースと、それをするすべてのクラス、またはスタティックメソッドをにバックポートするようびされ、Retrolambdaを1するがあります。つまり、にきれいなビルドをうがあります。また、バックポートのデフォルトメソッドは、モジュールまたはのをえてしません。

- のJDK 8ビルドが`invokedynamic`びしごとにしいクラスのをしたにれる`invokedynamic`ます。Retrolambdaは、 `java.lang.invoke.LambdaMetafactory`がにするバイトコードをキャプチャするようにし`java.lang.invoke.LambdaMetafactory`。 そのため、そのメカニズムにするはRetrolambdaをするがあります。

Retrolambda gradleプラグインはにRetrolambdaであなたのアンドロイドプロジェクトをします。は、 リリースページにあります 。

1. jdk8をダウンロードしてインストールする
2. `build.gradle`をしてください

```
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
    }
}
```

の

- LambdaをつJavaファイルでLintがします。 Androidのリントは Java 8のをしていないため、かにまたはでします。をするなフォークがあります。

- Google Playサービスをすると Retrolambdaがします。バージョン5.0.77には、Retrolambdaとのないバイトコードがまれています。これは、しいバージョンのプレイサービスでするがあります。アップデートできるは、これがされるソリューションになります。このを`-noverify`するには、4.4.52などののバージョンをするか、jvm argsに`-noverify`をします。

```
retrolambda {
  jvmArgs '-noverify'
}
```

オンラインでAndroidのJavaをむ https://riptutorial.com/ja/android/topic/9223/androidのjava

30: Androidのゲーム

き

JavaをしたAndroidプラットフォームでのゲームのな

- のはをカバーしています。はありませんが、SurfaceViewをして2Dゲームのをするをしています。
- ゲームをするときは、なデータをずしてください。のすべてはわれます

Examples

CanvasとSurfaceViewをしたゲーム

ここでは、SurfaceViewをしてな2Dゲームをするについてします。

まず、がです。

```
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.
    }

}
```

このアクティビティはAndroid Manifestでするがあります。

ゲームそのもの。まず、ゲームスレッドをすることからめます。

```
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");
    }
}
```

それがなです。これで、にすることができます。

さて、にすることからめましょう

```
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;
```

こののパ―トでは、イメ―ジがになります。それは100x100でなければならないが、それはきくてもさくてもよい。のために、Rectもうことができますただし、コ―ドをしするがあります

に、ビットマップをします。

```
private Bitmap PLAYER_BMP = BitmapFactory.decodeResource(getResources(),
R.drawable.my_player_drawable);
```

レンダリングでは、このビットマップをするがあります。

```
...
c.drawBitmap(PLAYER_BMP, x, y, null);
...
```

ちげのにまだいくつかのことがあります

にブ―ルがです。

```
boolean up = false;
```

onTouchEventでは、をします

```
if(ev.getAction() == MotionEvent.ACTION_DOWN){
    up = true;
}else if(ev.getAction() == MotionEvent.ACTION_UP){
    up = false;
}
```

また、ティックでは、プレイヤ―をするためにこれがです

```
if(up){
    velY -=1;
}
else{
    velY +=1;
```

```
}
if(velY >14)velY = 14;
if(velY <-14)velY = -14;
y += velY *2;
```

これでinitでこれがになります

```
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();
```

にはのものがです。

```
public static int WIDTH, HEIGHT;
```

こので、ゲームはです。つまり、それをしてテストすることができます。

は、プレーヤーのまたはをのにさせるがあります。にじて、プレーヤーをカスタムクラスとしてすることができます。その、すべてのプレイヤーのものをそのクラスにし、そのクラスのインスタンスをして、、レンダリング、そののロジックをできます。

さて、テストのでたように、からびます。だから々はそれをするがある。

まず、Rectをするがあります。

```
private Rect screen;
```

initでは、とさをした、であるしいをします。

```
screen = new Rect(0,0,WIDTH,HEIGHT);
```

は、メソッドのでのrectをとします

```
private Rect getPlayerBound(){
    return new Rect(x, y, x + PLAYER_BMP.getWidth(), y + PLAYER_BMP.getHeight();
}
```

そしてダニ

```
if(!getPlayerBound().intersects(screen){
    gameOver = true;
}
```

gameOVerのは、ゲームのをすためにもできます。

にするゲームのの

ドキュメントにはありません

オンラインでAndroidのゲームをむ https://riptutorial.com/ja/android/topic/10011/androidのゲーム

31: Androidのバージョン

	Androidバージョン		APIレベル	Build.VERSION_CODES
エンジェルケーキアルファ	1.0	2008923	1	ベース
バッテンベルクベータ	1.1	200929	2	BASE_1_1
カップケーキ	1.5	2009430	3	ケーキ
ドーナツ	1.6	2009915	4	ドーナツ
エクレア	2.0	20091026	5	エクレア
	2.0.1	2009123	6	ECLAIR_0_1
	2.1	2010112	7	ECLAIR_MR1
フロイョ	2.2	2010520	8	フロイオ
ジンジャーブレッド	2.3	2010126	9	ジンジャーブレッド
	2.3.3	201129	10	GINGERBREAD_MR1
ハニカム	3.0	2011122	11	ハニカム
	3.1	2011510	12	HONEYCOMB_MR2
	3.2	2011715	13	HONEYCOMB_MR1
アイスクリームサンドイッチ	4.0	20111019	14	ICE_CREAM_SANDWICH
	4.0.3	201112	15	ICE_CREAM_SANDWICH_MR1

	Androidバージョン		APIレベル	Build.VERSION_CODES
		16		
ゼリービーン	4.1	201279	16	ゼリービーン
	4.2	20121113	17	JELLY_BEAN_MR1
	4.3	2013724	18	JELLY_BEAN_MR2
キットカット	4.4	20131031	19	キットカット
		2014725	20	KITKAT_WATCH
ロリポップ	5.0	20141017	21	LOLLIPOP
	5.1	201539	22	LOLLIPOP_MR1
マシュマロ	6.0	2015105	23	M
ヌガー	7.0	2016822	24	N
	7.1.1	2016125	25	N_MR1

Examples

にのAndroidバージョンをする

`Build.VERSION_CODES`は、られているSDKバージョンコードのです。

デバイスのAndroidバージョンにづいてきでコードをするには、 `TargetApi`アノテーションをしてLintエラーをし、APIレベルにのコードをするにビルドバージョンをします。

23よりいAPIレベルをサポートするプロジェクトで、API-23でされたクラスをするのをにします。

```
@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);
        //.....................
    }
}
```

オンラインでAndroidのバージョンをむ https://riptutorial.com/ja/android/topic/3264/androidのバージョン

32: Androidのもの

Examples

サーボモータの

このでは、のようなをつサーボがあるとしています。

- 0と180ののき
- 20msのパルス
- パルス0.5ms
- パルス2.5 ms

これらのがハードウェアとするかどうかをするがあります。これは、されたににすると、サーボがするがあるからです。サーボがすると、Android Thingsデバイスがするがあります。`ServoController`クラスのは、 `setup()`と`setPosition()`の2つのメソッドでされてい`setup()`。

```
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();
        }
    }
```

```
}
```

デバイスでPWMをサポートするピンは、のようにできます。

```
PeripheralManagerService service = new PeripheralManagerService();

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

サーボを80から100のでにスイングさせるには、のコードをします

```
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();
```

にサーボモータをコンピューティングデバイスにフックすることなく、のコードをすべてコンパイルしてすることができます。については、コンピューティングデバイスのピンをしてください Raspberry Pi 3のピンはここにあります 。

に、サーボをVcc、Gnd、およびにするがあります。

オンラインでAndroidのものをむ https://riptutorial.com/ja/android/topic/8938/androidのもの

33: Androidのローカライズされた/

ロケールをできる、つまりユーザーたとえば12/ 24のをフォーマットするには、 DateUtilsクラスのメソッドをすることをおめします。これらのメソッドは、ユーザーにされるにもしています。

にカスタマイズされたの、すべてのをにできるため、 SimpleDateFormatクラスをすることをおめします。

Examples

DateUtils.formatDateTimeによるカスタムローカライズされた

DateUtils.formatDateTimeでは、をできます。したフラグについて、ローカライズされたdatetimeがされます。フラグをすると、のなどをめるかどうかをできます。

```
Date date = new Date();
String localizedDate = DateUtils.formatDateTime(context, date.getTime(),
DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_WEEKDAY);
```

formatDateTimeはになをいます。

Androidの/フォーマット

をフォーマットする

```
Date date = new Date();
DateFormat df = DateFormat.getDateInstance(DateFormat.MEDIUM);
String localizedDate = df.format(date)
```

とをフォーマットします。はいで、はいである

```
Date date = new Date();
DateFormat df = DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.LONG);
String localizedDate = df.format(date)
```

にカスタマイズされた/

```
Date date = new Date();
df = new SimpleDateFormat("HH:mm", Locale.US);
String localizedDate = df.format(date)
```

よくわれるパターン

- HH0-23
- hh112

- aAM / PMマーカー
- mm0-59
-
- ddの1-31
- MM
- yyyy

オンラインでAndroidのローカライズされた/をむ
https://riptutorial.com/ja/android/topic/6057/androidのローカライズされた-

34: Androidプロジェクトへの FuseViewの

き

fusetoolsからFuse.Viewをエクスポートし、のアンドロイドプロジェクトのでします。

たちのは、 hikrサンプルアプリケーションをエクスポートして`Activity`することです。

なは@ lucamtudor / hikr-fuse-viewです

Examples

hikrアプリ、ちょうどのandroid.view.View

- ヒューズをインストールするがあります https://www.fusetools.com/downloads
- チュートリアルをしているはずです
- `fuse install android`
- ターミナルで `uno install Fuse.Views`

ステップ1

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

ステップ2 `Fuse.Views`にパッケージをする

プロジェクトルートフォルダの`hikr.unoproj`ファイルをし、 `"Packages"`に`"Fuse.Views"`をします。

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

ステップ3 アプリケーションをする`HikrApp`コンポーネントをする

3.1プロジェクトのルートフォルダに、 `HikrApp.ux`という`HikrApp.ux`しいファイルをし、 `HikrApp.ux`のをりけ`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 `HikrApp.ux`

- `<App>`タグを`<Page>`
- `<Page> ux:Class="HikrApp"`をオープニング`<Page>`し`<Page>`
- `<ClientPanel>`すると、ステータスバーまたはのナビゲーションボタンについてはもうするはありません

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しくされた`HikrApp`コンポーネント`MainView.ux`

`MainView.ux`ファイルのをのようにきえます。

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

たちのアプリはなにりましたが、は`HikrApp`というのコンポーネントにそれをしました

ステップ**4** `MainView.ux`で、 `<App>`タグを`<ExportedViews>`きえ、 `ux:Template="HikrAppView"`を`<HikrApp />`し`<HikrApp />`

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

テンプレート`HikrAppView`、Javaからのビュ―へのをするためになので、えておいてください。

してください。ヒュ―ズのドキュメントから

> `ExportedViews`は、の`fuse preview`と`uno build`ときに`App`としてします

います。 Fuse Studioからプレビュ―するときにこのエラ―がします

> エラ―インクル―ドされたのUXファイルでAppタグをつけることができませんでした。 appタグをむUXファイルをインクル―ドするのをれましたか

ステップ**5** `SplashPage.ux`の`<DockPanel>`を`<GraphicsView>`ラップし`<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>
```

ステップ**6** auseライブラリとしてのヒュ―ズプロジェクトをエクスポ―トします。

- ターミナルのルートプロジェクトフォルダ `uno clean`
- ターミナルのルートプロジェクトフォルダ `uno build -t=android -DLIBRARY`

ステップ**7** Androidプロジェクトをする

- `.../rootHikeProject/build/Android/Debug/app/build/outputs/aar/app-debug.aar`から `.../androidRootProject/app/libs`
- `flatDir { dirs 'libs' }`をrootの`build.gradle`ファイルにします

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

buildscript { ... }

...

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

...
```

- `app/build.gradle`に`compile(name: 'app-debug', ext: 'aar')`を`compile(name: 'app-debug', ext: 'aar')`

```
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'
}
```

- `AndroidManifest.xml`のアクティビティにのプロパティをします。

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

あなたの`AndroidManifest.xml`はのようになります

```
<?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>
```

ステップ**8** `Activity` `Fuse.View` `HikrAppView`をする

- あなたの`Activity`は`FuseViewsActivity`をするがあることにしてください

```
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>
```

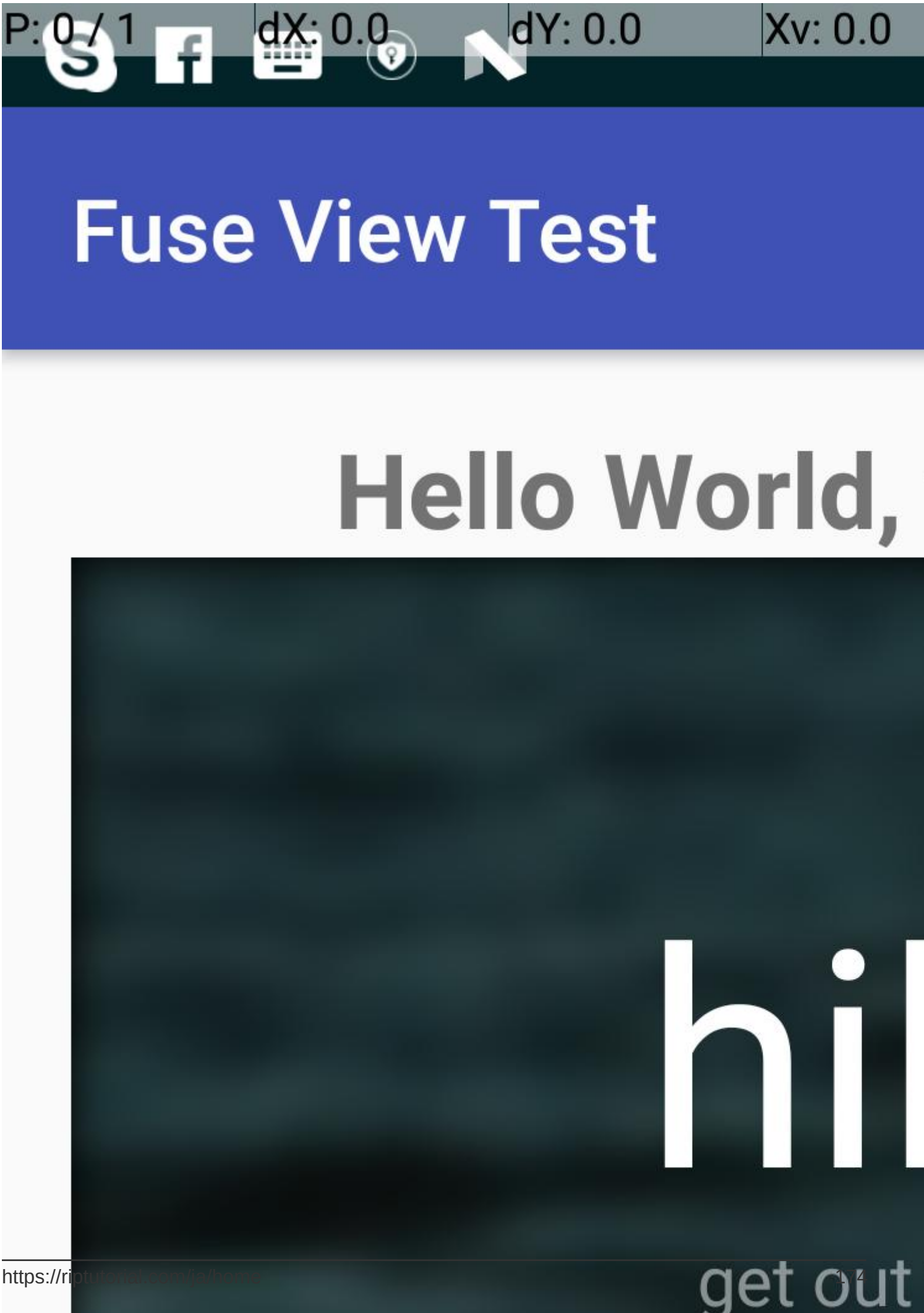
アンドロイドでるボタンをすと、アプリがクラッシュします。あなたはヒューズフォーラムでこのにうことができます。

```
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
```

そしてはこのようなものです。あなたはギブスでいクリップをつけることもできます。


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

https://riptutorial.com/ja/android/topic/10052/androidプロジェクトへのfuseviewの

35: AndroidプロジェクトのJenkins CIセットアップ

Examples

AndroidのJenkinsをするためのステップバイステップのアプローチ

これは、AndroidプロジェクトにJenkins CIをしてビルドプロセスをするためのステップバイステップガイドです。のでは、しいハードウェアをインストールしただけで、Linuxがインストールされていることをとしています。また、リモートマシンをしているもされています。

パートIマシンの

1. *ssh*であなたのUbuntuマシンにログインしてください

   ssh username@xxx.xxx.xxx

2. あなたのマシンにAndroid SDKのバージョンをダウンロードしてください

   wget https://dl.google.com/android/android-sdk_r24.4.1-linux.tgz

3. ダウンロードした*tar*ファイルをします

   sudo apt-get tarをインストールする
   tar -xvf android-sdk_r24.4.1-linux.tgz

4. はあなたのUbuntuマシンにJava 8をインストールするがあります。これはNougatのAndroidビルドのです。 Jenkinsでは、のでJDKとJRE 7をインストールするがあります。

   sudo apt-get python-software-propertiesをインストールする
   sudo add-apt-repository ppawebupd8team / java
   sudo apt-get update
   apt-get install openjdk-8-jdk

5. あなたのUbuntuマシンにJenkinsをインストールしてください

   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/debianstableバイナリ/> /
   etc/apt/sources.list.d/jenkins.list'
   sudo apt-get update
   sudo apt-get jenkinsをインストールする

6. あなたのAndroidセットアップにサポートされているのGradleバージョンをダウンロードし

てください

wget https://services.gradle.org/distributions/gradle-2.14.1-all.zip
unzip gradle-2.14.1-all.zip

7. あなたのUbuntuマシンにAndroidをしてください。まず、2でダウンロードしたAndroid SDKフォルダの*tools*フォルダにします。

cd android-sdk-linux / tools **//**な**SDK**をします
android update sdk --no-ui **// SDK**のバージョンをします
アンドロイドリストsdk -a | grep "SDKビルドツール" **//**なビルドツールを
android update sdk -a -u -t 4 **/**ビルドツールのバージョンを**prev**で**4**にリストアップされたものにします。 cmd。
javaをする

8. あなたのマシンに*Git*やそののVCSをインストールする

sudo apt-get install git

9. インターネットブラウザをしてJenkinsにログインします。アドレスバーに「 **`ipAddress:8080`** します。

10. のログインのパスワードをけるには、するファイルをのようにチェックしてくださいこのファイルにアクセスするにはsuがです。

cat / var / lib / jenkins / secrets / initialAdminPassword

パートII**Android**ジョブをするための**Jenkins**のセットアップ

1. ログインしたら、のパスにします。

Jenkins> Manage Jenkins>グローバルツール

2. このに`JAVA_HOME`にのエントリをし`JAVA_HOME`。

= JAVA_HOME
JAVA_HOME = / usr / lib / jvm / java-8-openjdk-amd64

3. *Git*にのをし、をします。

=デフォルト
/ usr / bin / git

4. に、のパスにみます。

Jenkins> Jenkinsの>

5. このので、 `ANDROID_HOME`を「グローバルプロパティ」にします。

   = ANDROID_HOME
   = / home / username / android-sdk-linux

パートIIIAndroidプロジェクトのJenkinsジョブをする

1. Jenkinsのホームで[*New Item]*をクリックします。
2. プロジェクトとをします。
3. [ ]タブで[ ]をします。に、[カスタムワークスペースをする]をします 。

   ディレクトリ/ home / user / Code / ProjectFolder

4. ソースコードで*Git*をします 。はこののために*Bitbucket*をしています

   リポジトリURL = https//ユーザpassword@bitbucket.org/project/projectname.git

5. リポジトリにのをします。

   チェックアウトにする
   サブディレクトリへのチェックアウト。 repo / home / user / Code / ProjectFolderのローカルサブディレクトリ

6. ビルドするブランチをします。

   */マスター

7. 「ビルド」タブで、「ビルドの 」ステップで「 シェルを 」をします。
8. *Execute*シェルでのコマンドをします。

   cd / home / user / Code / ProjectFolder && gradle clean assemble --no-daemon

9. プロジェクトでLintをするは、のビルドステップを*Execute*シェルにします 。

   /home/user/gradle/gradle-2.14.1/bin/gradle lint

JenkinsをしてAndroidプロジェクトをするためのシステムがついにしました。このは、QAチームとUATチームにビルドをリリースするためのあなたのをはるかににします。

PSJenkinsはあなたのUbuntuマシンののユーザなので、のコマンドをすることによって、あなた

のワークスペースにフォルダをするをえるべきです

```
chown -R jenkins .git
```

オンラインでAndroidプロジェクトのJenkins CIセットアップをむ
https://riptutorial.com/ja/android/topic/7830/androidプロジェクトのjenkins-ciセットアップ

36: AndroidのFileIO

き

Androidでのファイルのみきは、のJavaでのファイルのみきとじです。じ`java.io`パッケージをできます。ただし、きみがされているフォルダ、なアクセス、およびMTPのにするものがあります。

Androidは、 ここにされているように、のアプリケーションでファイルをするをします 。ファイルをしてするアプリケーションが1つしかないは、このはです。

Androidは、および、されたバンドル、みみデータベースなどのストレージオプションをします。によっては、なファイルをするよりもれたです。

Androidのアクティビティには、JavaのFile IOメソッドのわりにえるのメソッドはほとんどありません。たとえば、わりのため`File.delete()`をびすことができます`Context.deleteFile()`わりにするの`File.listFiles()`にびすことができる`Context.fileList()`ややないと、すべてのアプリののファイルのリストをしますコード。ただし、の`java.io`パッケージのはしていません。

Examples

フォルダの

アクティビティの`getFilesDir()`メソッドをびすことによって、フォルダをできますActivityは、Contextをするアプリケーションのクラスです。 ここを。はいはありません。あなたのアプリケーションだけがこのフォルダにアクセスできます。

アクティビティには、のコードをめることができます。

```
File myFolder = getFilesDir();
File myFile = new File(myFolder, "myData.bin");
```

バイトののをきむ

```
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()
```

このコードにのAndroidはありません。くのさなをきむは、 BufferedOutputStreamをしてデバイスのSSDのをらしてください。

オブジェクトのシリアライズ

いいJavaオブジェクトのシリアライズはAndroidでできます。のようなSerializableクラスをできます。

```
class Cirle implements Serializable {
  final int radius;
  final String name;

  Circle(int radius, int name) {
    this.radius = radius;
    this.name = name;
  }
}
```

ObjectOutputStreamにきみます。

```
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()
```

Javaオブジェクトのシリアライゼ―ションは、このチュ―トリアルの、によってはについて、あなたがをしたいのかにじて、かいかのどちらかになります。あなたがそれをうことにめたら、バ―ジョンについてまずんでください。

ストレ―ジSDカ―ドへのきみ

また、くのAndroidデバイスにされているメモリカ―ドSDカ―ドのみきがです。このにあるファイルは、USBケ―ブルをしてPCにデバイスをし、MTPプロトコルをにした、ユ―ザ―がのプログラムからアクセスすることができます。

SDカ―ドのをつけることはいくらかがあります。 Environmentクラスには、SDカ―ドにする「ディレクトリ」をするメソッドと、SDカ―ドがすべてし、きみなのもまれます。 このには、しいがつかるようにするためのながまれています。

ストレ―ジにアクセスするには、あなたのがです。Androidマニフェスト

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

いバ―ジョンのAndroidの、これらのをマニフェストにれるだけでですインスト―ルにユ―ザ―がするがあります。 Android 6.0からAndroidはのアクセスにユ―ザ―にをめるので、このしいアプロ―チをサポ―トするがあります。それのは、マニフェストになくアクセスがされます。

Android 6.0では、まずをし、されていないはそのをリクエストするがあります。コードはこのSOののにあります。

"Invisible MTP files"のをする。

MTPプロトコルをしてUSBケーブルでデスクトップにエクスポートするためのファイルをすると、しくされたファイルが、されたデスクトップPCでされているファイルエクスプローラにすぐにされないことがあります。しいファイルをするには、 MediaScannerConnectionをびすがあります

```
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)));
```

このMediaScannerConnectionコールコードは、ファイルでのみし、ディレクトリではしません。このはこのAndroidのバグレポートでされています。これはのバージョンやのデバイスでされるがあります。

きなファイルをう

さなファイルはでされ、なのコードのわりにみきできます。しかし、ファイルがきくなったりがいは、AndroidでAsyncTaskをして、バックグラウンドでファイルをするがあります。

```
    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
        }
    }
}
```

その

```
new FileOperation().execute("Some parameters");
```

このSOのには、AsyncTaskをしてびすのながまれています。また、IOExceptionやそののエラーをするにするエラーにするもしてください。

オンラインでAndroidのFileIOをむ https://riptutorial.com/ja/android/topic/8689/androidのfileio

37: AndroidFacebook SDK

- **newInstance** Facebookヘルパークラスのインスタンスをする。
- **loginUser** ユーザーにログインします。
- **signOut** ユーザーをログアウトします。
- **getCallbackManager** Facebookのコールバックをする。
- **getLoginCallback** ログインのコールバックをする。
- **getKeyHash** Facebook Key Hashをする。

パラメーター

パラメータ	
タグ	ロギングにされる
FacebookSignInHelper	facebookヘルパーへのな
CallbackManager	facebookのコールバック
アクティビティ	
PERMISSION_LOGIN	Facebookにログインするためになすべてのアクセスをむ。
loginCallback	Facebookのログインのコールバック

Examples

AndroidにFacebookログインをする

のを`build.gradle`してください

```
  // Facebook login
    compile 'com.facebook.android:facebook-android-sdk:4.21.1'
```

のヘルパークラスをあなたのユーティリティパッケージにしてください

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

あなたのでのコードをしてください

```
FacebookSignInHelper facebookSignInHelper =
FacebookSignInHelper.newInstance(LoginActivity.this, fireBaseAuthHelper);
facebookSignInHelper.loginUser();
```

`OnActivityResult`にのコードをしてください

```
 facebookSignInHelper.getCallbackManager().onActivityResult(requestCode, resultCode, data);
```

Facebookプロフィールのデータにアクセスするをする

ユーザーのFacebookプロファイルのをするは、じものにするアクセスをするがあります。

```
loginButton = (LoginButton)findViewById(R.id.login_button);

loginButton.setReadPermissions(Arrays.asList("email", "user_about_me"));
```

リスト、、などのをしけることができます。 なをんでのリストをしてください。

パブリックプロファイルにアクセスするためのなアクセス、、、などをするはありません。

Facebookログインののカスタムボタンをする

にFacebookログイン/サインアップをすると、ボタンはのようになります。



ほとんどの、アプリのデザインとしません。それをカスタマイズするはのとおりです。

```
<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>
```

の`com.facebook.login.widget.LoginButton`にラップして、を`FrameLayout`てください。

に、じ`FrameLayout`カスタムボタンをします。はいくつかのサンプルをしました。あなたはいつでもFacebookのボタンのためのあなたのなをることができ、ボタンのとしてそれをします。

にのカスタムボタンのクリックをfacecbookボタンのクリックにするだけです

```
//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();
```

```
    }
}
```

すばらしいですこれでボタンはのようになります



Facebookログイン/サインアップのにするのガイド

1. をするがあります。

2. *AndroidManifest.xml*ファイルにFacebookアクティビティをします

```
<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. レイアウトXMLファイルにログインボタンをします

```
<com.facebook.login.widget.LoginButton
    android:id="@+id/login_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

4. あなたはFacebookのボタンをっています。ユーザーがクリックすると、Facebookのログインダイアログがアプリののにされます。ここでユーザーはをし、*[*ログイン *]*ボタンをすことができます。がしい、ダイアログはするをし、ボタンをむのアクティビティにコールバックがされます。のコードは、そのコールバックをけるをしています。

```
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.
    }
});
```

Facebookからのログアウト

Facebook SDK 4.0、これはたちがログアウトするです

```
com.facebook.login.LoginManager.getInstance().logOut();
```

4.0よりのバージョンでは、にアクセストークンをクリアすることでログアウトしなくなりました。

```
Session session = Session.getActiveSession();
session.closeAndClearTokenInformation();
```

オンラインでAndroidFacebook SDKをむ https://riptutorial.com/ja/android/topic/3919/android-facebook-sdk

38: AndroidGoogle Maps API v2

パラメーター

パラメータ	
グーグルマップ	`GoogleMap`は、 `onMapReady()`イベントでけられるオブジェクトです
MarkerOptions	`MarkerOptions`は`Marker`ビルダークラスであり、マーカーをマップにするためにされます。

1. Google PlayサービスSDKがインストールされています。
2. Googleコンソールアカウント
3. GoogleコンソールでしたGoogle Maps APIキー。

Examples

デフォルトのGoogleマップアクティビティ

このアクティビティコードは、SupportMapFragmentをしてGoogleマップをみむためのなをします。

GoogleマップV2 APIには、くしいでをみむことができます。

アクティビティでは、 **OnMapReadyCallBack**インターフェイスをするがあります。このインターフェイスには、 **SupportMapFragment**をするたびにされる**onMapReady**メソッドオーバーライドがしています。 **getMapAsyncOnMapReadyCallback** ;コールはにしました。

マップではマーカー 、 ポリゴン 、 ポリライン をしてインタラクティブなをユーザーにします。

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));
    }
}
```

のコ―ドは、レイアウトをさせることにしてください。これは、Container Layoutのにれにされたsupportmapfragmentをち、IDが`R.id.map`でされています。レイアウトファイルをにします。

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>
```

カスタムGoogleマップスタイル

マップスタイル

Googleマップには、のコ―ドをして、さまざまなスタイルのセットがされます。

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

なるマップスタイルはのとおりです。

ノ―マル

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

なロ―ドマップ。、の、などのなのがされています。とラベルもされます。



ハイブリッド

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

をしたデータ。とラベルもされます。




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

データ。とラベルはされません。




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

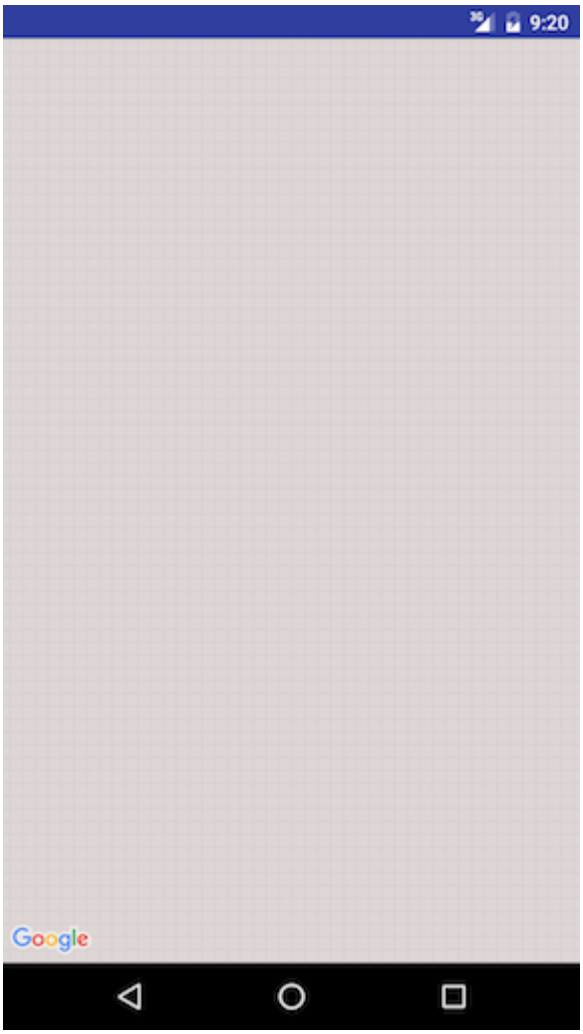
データ。このマップには、、とラベル、およびがまれています。のとラベルもされます。



し

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

タイルはありません。マップは、タイルがロードされていないのグリッドとしてレンダリングされます。



そののスタイルオプション

マップ

いズームレベルでは、のがにされます。これらはマップとばれ、「」および「サテライト」マップタイプのにのみされます。

マップをまたはにするはのとおりです。

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

マップにカスタムスタイルをできます。

onMapReadyメソッドでのコードスニペットをします

```
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);
    }
```

*res*フォルダのに*raw*というフォルダをし、スタイルjsonファイルをします。サンプルstyle.jsonファイル

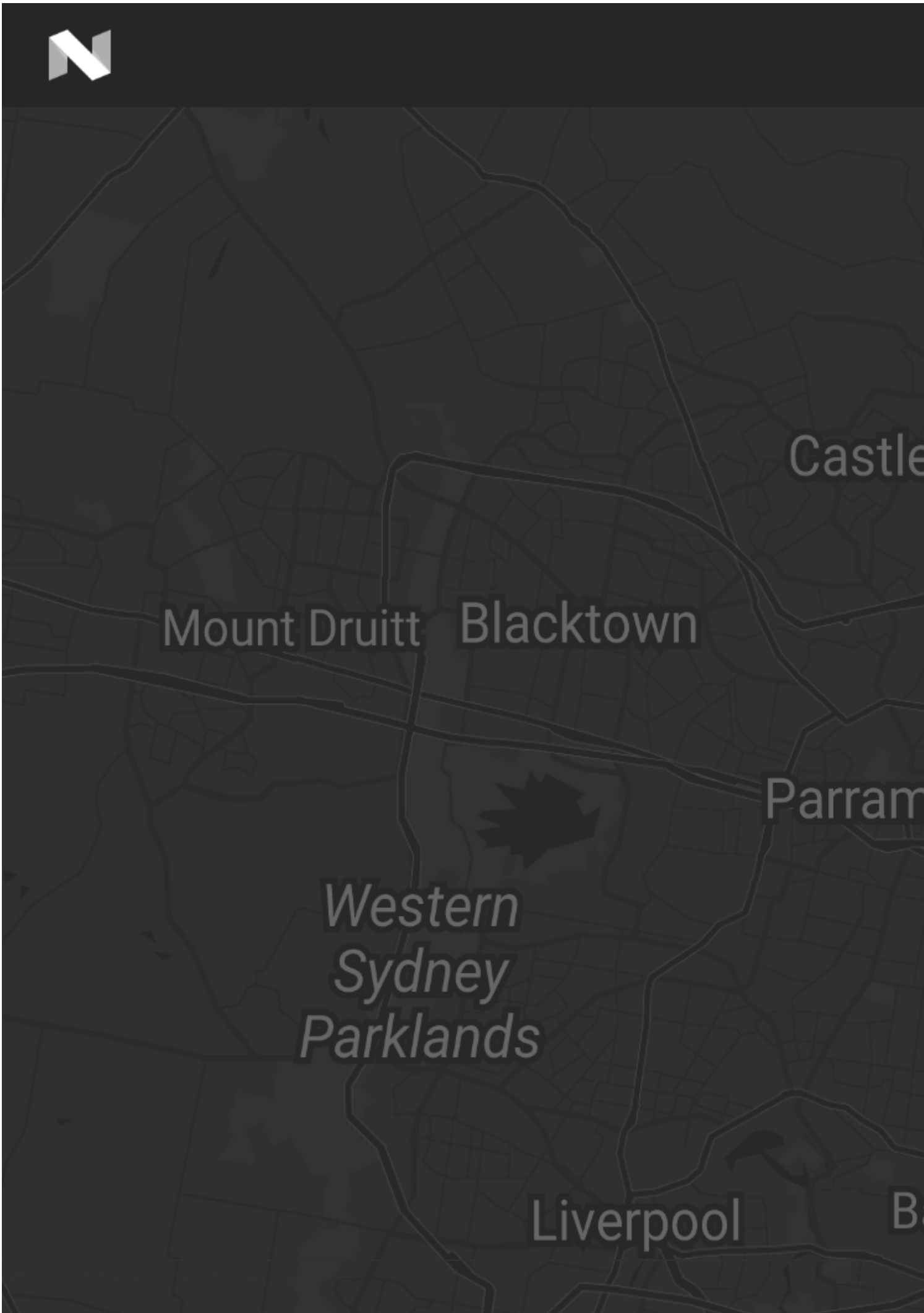
```
  [
{
  "featureType": "all",
  "elementType": "geometry",
  "stylers": [
    {
      "color": "#242f3e"
    }
  ]
},
{
  "featureType": "all",
  "elementType": "labels.text.stroke",
  "stylers": [
    {
      "lightness": -80
    }
  ]
},
{
  "featureType": "administrative",
  "elementType": "labels.text.fill",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

スタイルjsonファイルをするには、このリンクをクリックしてください


Castle
Mount Druitt
Blacktown
Parram
Western
Sydney
Parklands
Liverpool

するには、このようにします。

`MyLocation`ホルダークラス

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

`MyLocation`オブジェクトのリストを`MyLocation`、それぞれにマーカーをするは`MyLocation`です。

```
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));
 }
}
```

このでは、`mMap`はアクティビティのクラスメンバで、`onMapReady()`オーバーライドでけったマップにりてています。

MapViewのレイアウトにGoogleマップをめむ

されたMapViewクラスをすると、GoogleマップをAndroidビューとしてうことはです。そのはMapFragmentとによくています。

レイアウトでは、MapViewをのようにします。

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

あなたのアクティビティは、するためにOnMapReadyCallbackインターフェイスをするがあります

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

Googleマップにをする

のにマ―カ―をし、カメラをのにするなアクティビティクラスをにします。

ここではいくつかにんでいます

- のの
- Locationがえられると、 `setMyLocationEnabled()`びし、GoogleApiClientをビルドしてします
- GoogleApiClientがされると、ロケ―ションのをリクエストします

```
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
        }
    }

}
```

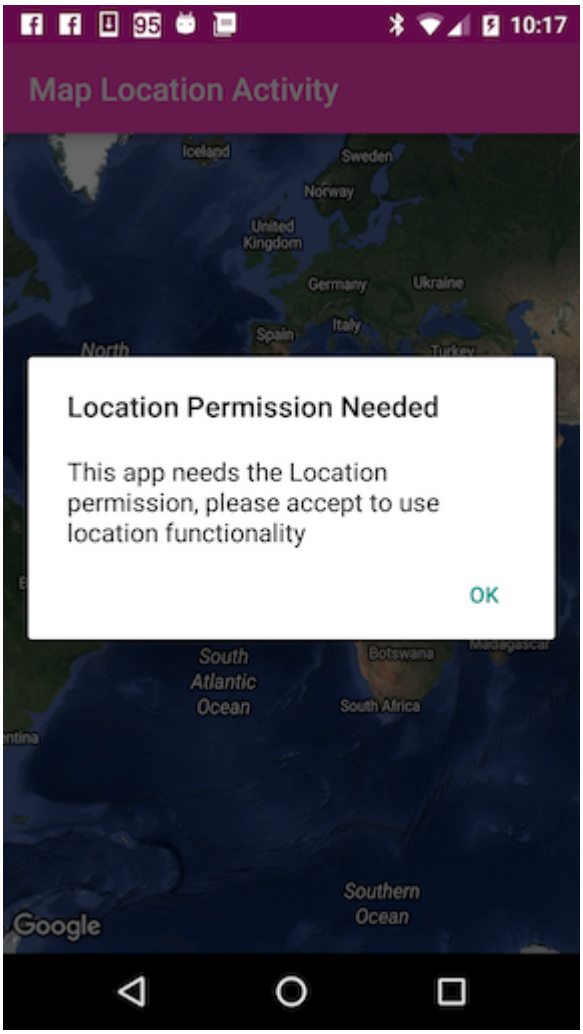
activity_main.xml

```
<?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">

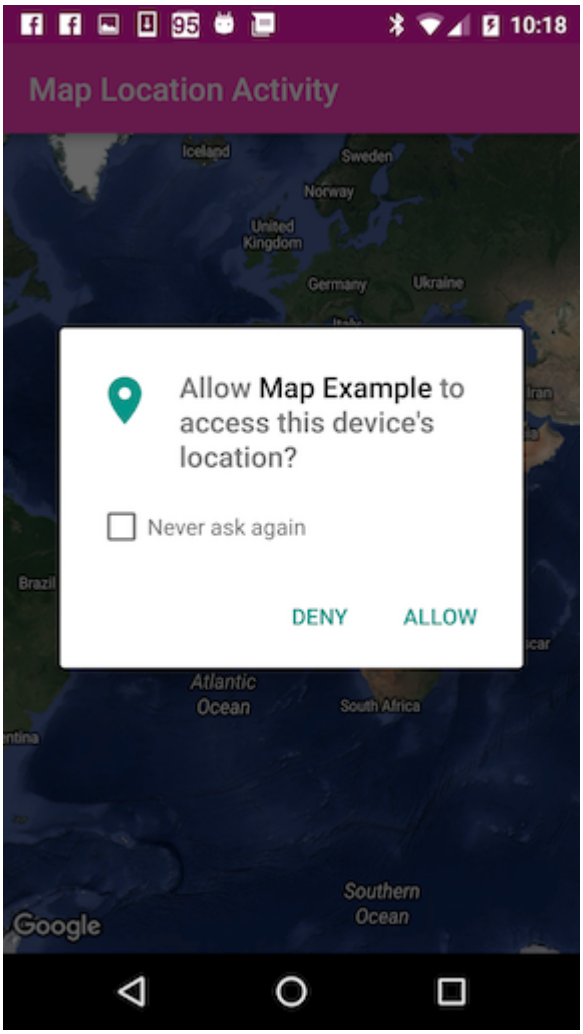
    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:map="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/map"
        tools:context="com.example.app.MapLocationActivity"
        android:name="com.google.android.gms.maps.SupportMapFragment"/>

</LinearLayout>
```

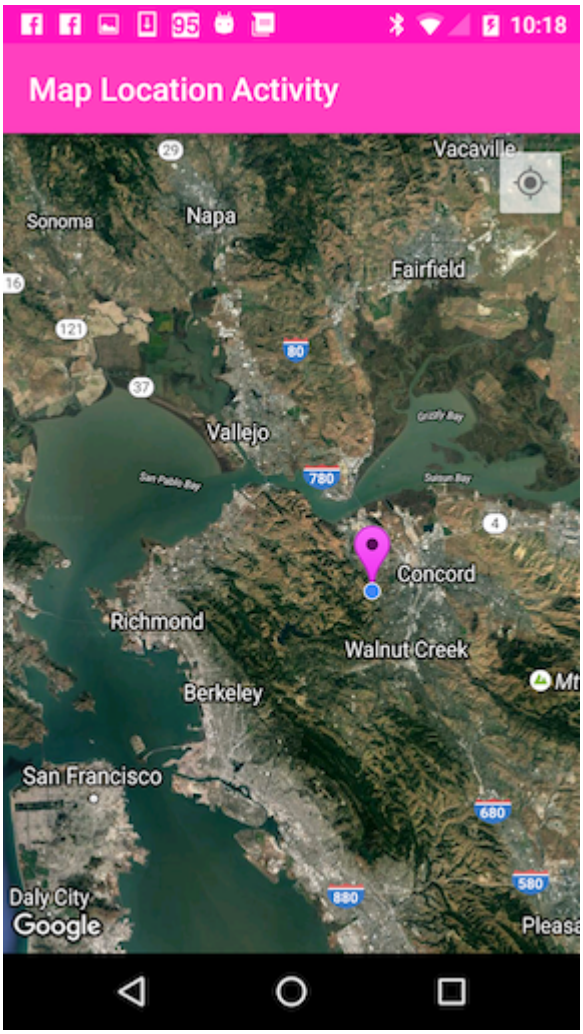
AlertDialogをしてMarshmallowとNougatになをするこのは、にをしていた、またはをえてででりした



`ActivityCompat.requestPermissions()`びすことで、MarshmallowとNougatにするLocationをユーザーにめる



のがえられたら、カメラをのにしてマーカーをします。



キーストアファイルの**SH1**フィンガープリントの

のGoogle Maps APIキーをするには、デバッグ/リリースキーストアのSH1フィンガープリントをAPIコンソールにするがあります。

ここでする**JDK**の**keytool**プログラムをして、キーストアをすることができます。

もう1つのは、デバッグ/リリースでされたアプリケーションでこのスニペットをし、ハッシュをログにすることによってプログラムでをすることです。

```
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());
}
```

がクリックされたときにGoogleマップをしないライトモード

Googleマップがライトモードでされると、をクリックするとGoogleマップアプリケーションがきます。このをにするには、`MapView`で`setClickable(false)`をびすがあります。

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

UISettings

`UISettings`をすると、Googleマップのをできます。

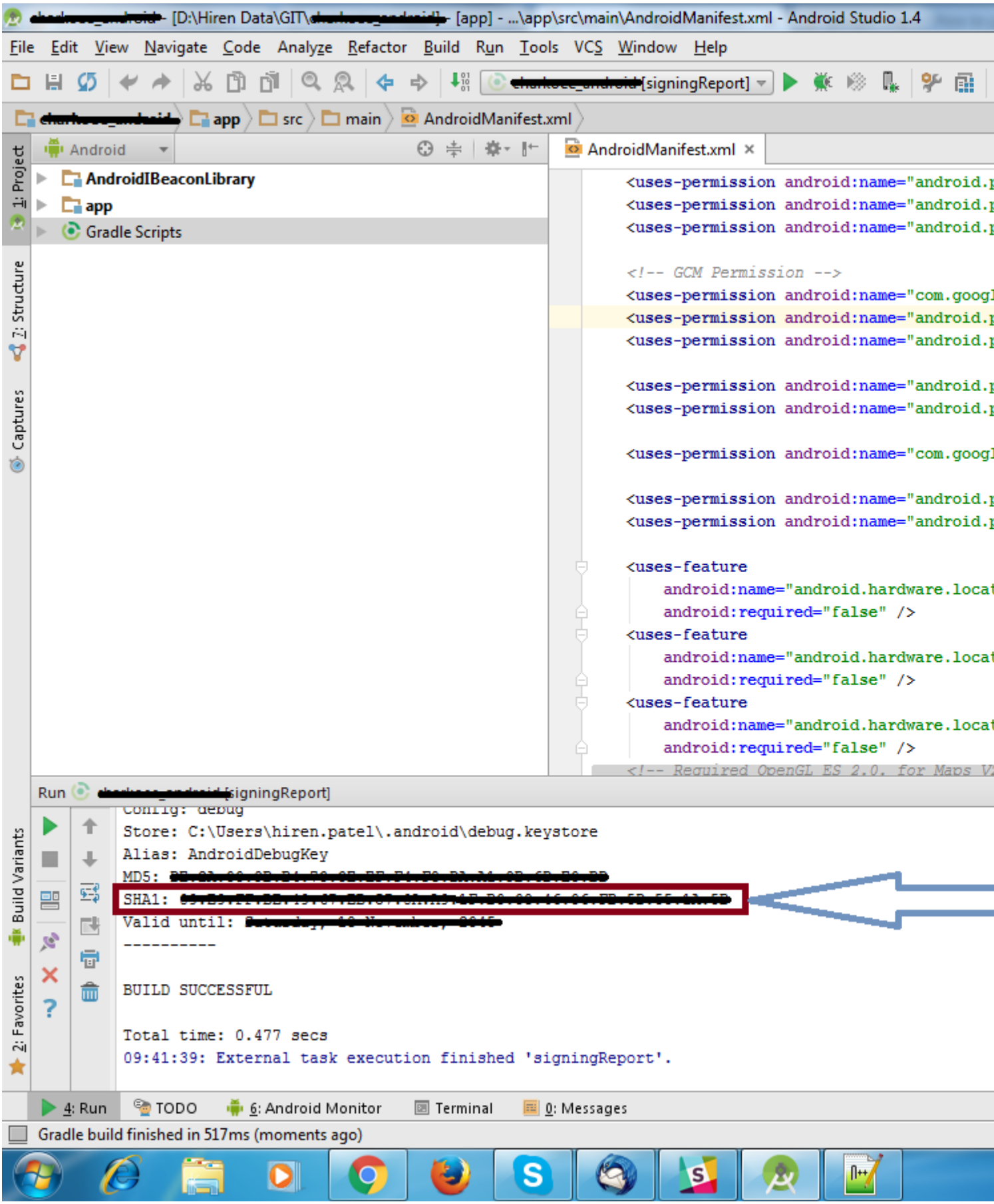
いくつかのなのをにします。

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



デバッグ**SHA1**フィンガープリントをする

1. Android Studioをく
2. プロジェクトをく
3. Gradleをクリックしてくださいのパネルから、 **Gradle Bar**がされます
4. 「」をクリックします **Gradle Bar**からRefreshをクリックすると、プロジェクトのGradleスクリプトのリストがされます
5. あなたのプロジェクトあなたのプロジェクトフォームリスト rootをクリックしてください
6. タスクをクリックします。
7. アンドロイドをクリック
8. signingReportをダブルクリックします バーに **SHA1**と**MD5**がされます



InfoWindowクリックリスナー

に、マーカーのInfoWindowクリックイベントごとになるアクションをするのをします。

マーカーIDがキーであるHashMapをし、はInfoWindowがクリックされたときにるべきするアクションです。

その、 `OnInfoWindowClickListener`をして`OnInfoWindowClickListener`をクリックしたユーザーのイベントをし、HashMapをして、するアクションをします。

このなでは、マーカーのInfoWindowがクリックされたことについて、のアクティビティをきます。

HashMapをActivityまたはFragmentのインスタンスとしてします。

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

に、マーカーをするたびに、マーカーIDとInfoWindowがクリックされたときにするアクションでHashMapにをします。

たとえば、2つのマーカーをし、それぞれにしてするアクションをします。

```
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");
```

InfoWindowクリックリスナーで、HashMapからアクションをし、マーカーのアクションについてするアクティビティをきます。

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

コードがフラグメントにあるは、MainActivity.thisをgetActivityにきえます。

オフセットの

マップポイントの*x*と*y*のをにじてすると、Googleマップのオフセットのをすることができます。デフォルトでは、マップビュ―のにされます。あなたはそれをしたいメソッドのをびしてください `changeOffsetCenter(location.getLatitude(),location.getLongitude());`ように、あなたの`onLocationChanged`でするがい`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

        }
```

オンラインでAndroidGoogle Maps API v2をむ https://riptutorial.com/ja/android/topic/170/android-google-maps-api-v2

39: AndroidGradle

き

GradleはJVMベースのビルドシステムで、はコンパイルおよびアプリケーションのプロセスをするためにできるのスクリプトをできます。これはなプラグインベースのシステムで、ビルドプロセスのさまざまなをすることができます。 `.jar`コンパイルと、のダウンロードと、`AndroidManifest`へのフィールドの、またはのSDKバージョンのなどがまれます。

- `apply plugin` されるはずのプラグインは、 `'com.android.application'`または `'com.android.library'`です。
- `android` あなたのアプリのな
    - `compileSdkVersion` コンパイルSDKのバージョン
    - `buildToolsVersion` ビルドツールのバージョン
    - `defaultConfig` フレーバーとビルドタイプによってきされるデフォルト
        - `applicationId` あなたがするアプリケーションIDは、えばPlayStoreのではあなたのパッケージとほとんどじです
        - `minSdkVersion` のSDKバージョン
        - `targetSdkVersion` あなたがコンパイルしたSDKのバージョンですにしいものにするがあります
        - `versionCode` にきくなるがあるバージョン
        - `versionName` ユーザーがアプリのページにできるバージョン
    - `buildTypes` のをるTODO
- `dependencies` あなたのアプリのMavenやローカルの
    - のを`compile`する
    - `testCompile` ユニットまたはテストの

もしてください

- のgradleのホームページ
- どのようにgradleのビルドをする
- gradleのためのアンドロイドプラグイン
- Android Gradle DSL

AndroidGradle - ドキュメント

あなたがAndroidでgradleのについてよりくのトピックとをつけることができるのタグがあります。

http://www.riptinar.com/topic/2092

Examples

なbuild.gradleファイル

これは、モジュールのデフォルトの`build.gradle`ファイルのです。

```
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ドメイン

のファイルのブロックは、`DSL`ドメインとばれます。

プラグイン

の、 `apply plugin: 'com.android.application'`は、 GradleのAndroidプラグインをビルドにし、`android {}`ブロックをしてAndroidのビルドオプションをします。

Androidアプリケーションの

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

Androidライブラリの

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

のサンプルの**DSL**をする

2の、 `android {...}`ブロックは、あなたのプロジェクトにするをむAndroid `DSL`です。

たとえば、アプリのコンパイルにGradleがするAndroid APIレベルをする`compileSdkVersion`をできます。
サブブロック`defaultConfig`は、マニフェストのデフォルトがされます。それらをProduct Flavorsで `override`することができ`override`。

のでをつけることができます

- アプリモジュールのDSL
- ビルドの
- の
-

`dependencies`ブロックは`android`ブロックのでされています`{...}` これはAndroidプラグインでされていませんが、のGradleです。
`dependencies`ブロックは、アプリにめるライブラリはAndroidライブラリですが、Javaライブラリもですをします。 Gradleはこれらのをにダウンロードしますローカルコピーができない。のライブラリをするは、の`compile`をするだけです。

ここにあるの1つをてみましょう

```
compile 'com.android.support:design:25.3.1'
```

このはに

のプロジェクトにAndroidサポートデザインライブラリへのをしてください。

Gradleはライブラリがダウンロードされていることをし、アプリでできるようにします。コードもアプリにまれます。

Mavenにれている、このは*GroupId* 、コロン、 *ArtifactId* 、のコロン、にインクルードするのバージョンであり、バージョンをにできます。

プラス+をしてアーティファクトバージョンをすることはですが、することをおめします。ライブラリがあなたのなしにをってされると、アプリケーションにクラッシュするがあります。

なるのをすることができます

- ローカルバイナリ
- モジュールの
- リモート

aarのフラットにするがあります。

しくは、 このトピックをしてください。

***appcompat-v7*の -v7にする**

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

これは、このライブラリ `appcompat` がAndroid APIレベル7とがあることをします。

junitにするjunit4.12

これはユニットテストのテストです。

なるビルドにのの

のビルドにしてのみをするように`testCompile`か、の`compile`わりに`debugCompile` 、 `testCompile`または`releaseCompile`をして、 ビルドのまたはのフレーバー たとえば、デバッグ、テストまたはリリース 。

これは、リリースビルドからテストおよびデバッグのをるためにちます。これにより、リリース`APK`をなりスリムにち、デバッグをしてアプリケーションにするをできなくなります。

---

signingConfig

`signingConfig`すると、 `keystore`をむようにGradleをし、これらのをしてされたAPKがされ、Playストアリリースのがっていることをできます。

ここでは、 のトピックをつけることができます。

Gradleファイルにのをすることはされません。をするには、 `signingConfigs`をします。
さまざまなでできます。

- ファイルにする
- それらをのにします 。

については、このトピックをしてください キーストアのパスワードをせずにAPKにしてください 。

Gradle for Androidのについては、 の**Gradle**のトピックをごください 。

のフレーバーの

のフレーバーは、のように`android { ... }`ブロックの`build.gradle`ファイルでされています。

```
...
android {
    ...
    productFlavors {
        free {
            applicationId "com.example.app.free"
            versionName "1.0-free"
        }
        paid {
            applicationId "com.example.app.paid"
            versionName "1.0-paid"
        }
    }
}
```

こうすることで、 `free`と`paid` 2のがさ`paid`。それぞれはののとをつことができます。たとえば、のしいフレーバーには、の`main`フレーバーとはの`applicationId`と`versionName`があり`versionName` デフォルトでできるので、ここにはされません。

フレーバーのの

のビルドにしてするとに、のフレーバーにをすることができます。

このでは、 `free`と`paid`という2つのフレーバーをにしているとしますここではフレーバーのについてしくします 。
`free`フレーバーにはAdMobのを、 `paid`ものにはPicassoのライブラリをすることができます。

```
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'
}
...
```

フレーバーのリソースをする

のののためにリソースをすることができます。

このでは、 `free`と`paid`とばれる2つのフレーバーをにしているものとします。フレーバーのリソースをするために、 `main/res`フォルダーのにのリソースフォルダーをします。このフォルダーはのようにリソースをできます。このでは、フレーバーごとに`status`します。

*/ 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>
```

のフレーバの`status`は、 `main`フレーバの`status`のをきします。

ビルドフィールドのと

BuildConfigField

Gradleをすると、 `buildConfigField`でをできます。これらのは、に`BuildConfig`クラスのフィールドとしてアクセスできます。これは、 `defaultConfig`ブロックのすべてのフィールドをし、にじて々

のビルドフレーバのためにそれらをオーバーライドすることで、 フレーバをするためにできます。

このでは、ビルドのをし、テストではなくプロダクションのビルドにフラグをけます。

```
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'
        }
    }
}
```

にされた<package_name> genフォルダの`BuildConfig`.javaには、のについてのフィールドがあります。

```
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";
}
```

されたフィールドは、された`BuildConfig`クラスにアクセスすることで、にアプリでできるようになりました

```
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のproductFlavorsはリソースをします。どのようなタイプのリソース string 、 dimen 、 colorなど dimen 。これは、なファイルにリソースをすることとています。たとえば、 strings.xml ファイルのをします。は、あなたのproductFlavor / buildVariantについて、gradleでされたものをできることです。にアクセスするには、リソースファイルからresにアクセスしているとじコードをします。

```
getResources().getString(R.string.app_name)
```

なのは、このでされたリソースは、ファイルにされているのリソースをできないということです。らはしいリソースをすることしかできません。

のライブラリGoogle Maps Android APIなどでは、マニフェストにmeta-dataタグとしてされているAPIキーがです。デバッグおよびプロダクションビルドになるキーがなは、Gradleでされたマニフェストプレースホルダをします。

あなたのAndroidManifest.xmlファイル

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="${MAPS_API_KEY}"/>
```

に、 build.gradleファイルにフィールドをします。

```
android {
    defaultConfig {
        ...
        // Your development key
        manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
    }

    productFlavors {
        prod {
            // Your production key
            manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
        }
    }
}
```

Androidのビルドシステムは、いくつかのフィールドをにし、 BuildConfig.javaます。これらのフィールドはのとおりです。

フィールド	
DEBUG	アプリケーションがデバッグモードであるかリリースモードであるかをす Boolean

フィールド	
APPLICATION_ID	アプリケーションのIDをむ`String` `com.example.app`
BUILD_TYPE	アプリケーションのビルドタイプをむ`String` は、 `debug`または`release`いずれか
FLAVOR	ビルドののフレーバをむ`String`
VERSION_CODE	バージョンビルドをむ`int` これはじです`versionCode`に`build.gradle`または`versionCode`に`AndroidManifest.xml`
VERSION_NAME	バージョンビルドをむ`String` これはじです`versionName`に`build.gradle`または`versionName`に`AndroidManifest.xml`

にえて、フレーバーののディメンションをしたは、ディメンションにのがされます。たとえば、`color`と`size`フレーバーの2つのがある、のもあります。

フィールド	
FLAVOR_color	「」フレーバのをむ`String`
FLAVOR_size	「サイズ」フレーバのをむ`String`

を "dependencies.gradle"ファイルでする

マルチモジュールプロジェクトをう、をくのビルドファイル、にAndroidサポートライブラリやFirebaseライブラリなどのなライブラリにするのではなく、のにさせるとです。

されるの1つは、モジュールごとに1つの`build.gradle`をむGradleビルドファイルと、プロジェクトルートのビルドファイルと、のためののビルドファイルをすることです。

```
root
  +- gradleScript/
  |     dependencies.gradle
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
```

そして、すべてのは`gradleScript/dependencies.gradle`ます

```
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}",
    ];
}
```

これはトップレベルのファイル`build.gradle`でそのファイルからできます。

```
// Load dependencies
apply from: 'gradleScript/dependencies.gradle'
```

`module1/build.gradle`ように

```
// Module build file
dependencies {
    // ...
    compile supportDependencies.appCompat
    compile supportDependencies.design
    compile firebaseDependencies.crash
}
```

のアプロ―チ

ライブラリのバ―ジョンをするためのあまりえめなアプロ―チは、バ―ジョンをとしてし、どこでもすることでできます。

のルート`build.gradle`します。

```
ext.v = [
    supportVersion:'24.1.1',
]
```

じライブラリをするすべてのモジュ―ルで、なライブラリをします

```
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}"
```

フレーバーのリソースのディレクトリ

さまざまなのアプリケーションビルドになるリソースをめることができます。フレーバーのリソースをするには、 `src`ディレクトリにフレーバーののののディレクトリをし、とじでリソースをします。

たとえば、のある`Development`がいて、のランチャーアイコンをしたいは、ディレクトリ `src/development/res/drawable-mdpi ic_launcher.png src/development/res/drawable-mdpi`をし、そのディレクトリにのアイコンをむ`ic_launcher.png`ファイルをします。

ディレクトリはのようになります。

```
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'
```

もちろん、このはdrawable-hdpi、drawable-xhdpi などのアイコンもします。

Android Studioプロジェクトにbuild.gradleファイルが2つあるのはなぜですか

`<PROJECT_ROOT>\app\build.gradle`は、 アプリケーションモジュールにのものです 。

`<PROJECT_ROOT>\build.gradle`は、すべてのサブプロジェクト/モジュールにのオプションをできる「トップレベルビルドファイル」です。

プロジェクトでのモジュールをしている、ローカルライブラリとしての`build.gradle`ファイルがあります。 `<PROJECT_ROOT>\module\build.gradle`

レベルのファイルでは、のプロパティをビルドスクリプトブロックまたはいくつかのなプロパティとしてできます。

```
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"
}
```

`app\build.gradle` 、モジュールのプロパティのみをします。

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


android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion
}

dependencies {
    //.....
}
```

gradleからシェルスクリプトをする

シェルスクリプトはビルドをにあなたがえることのできるものにするためのになです。

として、protobufファイルをコンパイルし、のjavaファイルをソースディレクトリにしてさらにコンパイルするためのなスクリプトをします。

```
def compilePb() {
    exec {
        // NOTICE: gradle will fail if there's an error in the protoc file...
        executable "../pbScript.sh"
    }
}

project.afterEvaluate {
    compilePb()
}
```

このの「pbScript.sh」シェルスクリプトは、プロジェクトのルートフォルダにあります

```
#!/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
```

Gradleエラーのデバッグ

はGradleのです- ゼロのとはですかどのようにしますかなのためにそれをてください。

アプリケーションをしていて、にそうであるようにえるいくつかのGradleエラーがしたとしましょう。

```
: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
```

StackOverflowであなたのをすると、あなたのプロジェクトをきれいにしたりビルドしたり、MultiDexをにしたりしているとうもいます。

よりくのをるはありますが、Gradleのは、そのメッセージののののエラーをします `module:someTask FAILED`とされたの`:module:someOtherTask`。 したがって、あなたがあなたのエラーにするをしたは、あなたのをして、エラーのをしてください。

したがって、「ゼロの」がられます。そのは、あなたがしようとしているもののいです。ここでは、もにこるものがいくつかあります。

- `1`はなるなエラーコードであり、はGradleにあるがい
- `2`は、するやプロジェクトのミスにしているようです。
- `3`はあまりにもくのやメモリのをんでいるようです。

のなプロジェクトのクリーンアップとリビルドをしたはのとおりです。

- `1` - されたエラーにしてください。に、これはコンパイルのエラーです。つまり、プロジェクトのコードのがです。これには、AndroidプロジェクトのXMLとJavaのがまれます。
- `2` `3` - ここでは、 multidexをにするがいくつかあります。をするがありますが、もなです。あなたがなぜそれをしているのかできないはリンクを、おそらくそれはありません。なソリューションでは、ライブラリののののたとえば、マップやサインインなどのライブラリを1つだけするがあるなど、すべてのGoogle Playサービスをらすがあります。

ビルドのとのフレーバになるアプリケーションIDをする

applicationIdSuffixをして、`buildType`または`productFlavor`なるアプリケーションIDまたはパッケージをできます。

`buildType`の`applicationId`のの

```
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"
    }
}
```

の`applicationIds`はのようになります。

- のためのcom.package.android `release`
- com.package.android。 **dev** `development`
- com.package.android。 `testing`ための**qa**

これは`productFlavors`でもにうことができます

```
productFlavors {
    free {
        applicationIdSuffix ".free"
    }
    paid {
        applicationIdSuffix ".paid"
    }
}
```

の`applicationIds`はのようになります。

- com.package.android。 フリーフレーバーを`free`
- com.package.android。 われたのために`paid`

キーストアのパスワードをせずに**APK**にする

のプロパティをして、 `build.gradle`ファイルのapkにするをできます。

- `storeFile` キーストアファイル
- `storePassword` キーストアのパスワード
- `keyAlias` キーエイリアス
- `keyPassword` キーエイリアスのパスワード

くの、 `build.gradle`ファイルでこのようなをけるがあります。

Akeystore.propertiesファイルをしてリリースをする

keystore.propertiesなどのプロパティファイルからをみむように、アプリケーションのbuild.gradleをすることができます。

このようなをするとです

- は、 build.gradleファイルとはのものです
- キーストアファイルにパスワードをするためにプロセスにするはありません
- バージョンからkeystore.propertiesファイルをにすることができkeystore.properties

まず、のようなのプロジェクトのルートにkeystore.propertiesというファイルをしますのできえます。

```
storeFile=keystore.jks
storePassword=storePassword
keyAlias=keyAlias
keyPassword=keyPassword
```

さて、あなたのアプリのbuild.gradleファイルで、のようにsigningConfigsブロックをしてsigningConfigs

```
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']
            }
        }
    }
}
```

これはにありますが、バージョンから**keystore.properties**ストアファイルと**keystore.properties**ファイルのをすることをれないでください。

すべきことのカップル

- keystore.propertiesファイルでされたstoreFileパスは、アプリのbuild.gradleファイルからのパスであるがあります。このでは、キーストアファイルがアプリケーションのbuild.gradle

ファイルとじディレクトリにあることをとしています。
- このでは、プロジェクトのルートに`keystore.properties`ファイルがあり`keystore.properties`。のにくは、`rootProject.file('keystore.properties')`のをプロジェクトのルートをにしてのにしてください。

Bをする

じことがプロパティファイルなしでもでき、パスワードをつけるのがしくなります

```
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
    }
}
```

`"ps"`はグローバルにすることもできますが、Android Studioのシエルにのみすることで、よりなアプローチがです。
Linuxでは、これはAndroid Studioの`Desktop Entry`することでうことができます

```
Exec=sh -c "export ps=myPassword123 ; /path/to/studio.sh"
```

しくは、 このトピックをしてください。

"version.properties"ファイルでビルドをバージョンする

Gradleをすると、パッケージをビルドするたびににインクリメントすることができます。これをうには、のの`build.gradle`とじディレクトリに`version.properties`ファイルをします。

```
VERSION_MAJOR=0
VERSION_MINOR=1
VERSION_BUILD=1
```

メジャーとマイナーのをにする。に`build.gradle android`セクションにのコードをします

```
// 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)
  }
}
```

このは、Javaで、な{メジャー}。{マイナー}。{ビルド}の`BuildConfig.VERSION_NAME`と、ビルドのの`BuildConfig.VERSION_CODE`としてアクセスできます。

apkをしてバージョンをする

これは、アプリケーションファイル.apkをするためのコードです。 `newName`なるをりてることによって、をできます

```
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 + "]");
        }
    }
}
```

さなAPKファイルサイズのをにする

すべてのイメージをでするは、APKファイルのサイズをさくするためにAPT Cruncherをにします。

```
android {
```

```
    aaptOptions {
        cruncherEnabled = false
    }
}
```

gradleをしてProguardをにする

アプリケーションのProguardをにするには、モジュールレベルのグラデーションファイルでにするがあります。 `minifyEnabled`のを`true`にするがあり`true`。

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

のコードは、デフォルトのAndroid SDKにまれるProguardのを、あなたのモジュールの "proguard-rules.pro"ファイルとみわせて、あなたのapkにします。

GradleとAndroidStudioのなNDKプラグインのサポートをにする

なGradleプラグインをにしてし、AndroidStudioのNDKサポートをします。のをたしていることをしてください。

- Gradle 2.10こののの
- Android NDK r10
- ビルドツールv19.0.0をしたAndroid SDK

MyApp / build.gradleファイルをする

えばbuild.gradleのdependencies.classpathをします。

```
classpath 'com.android.tools.build:gradle:2.1.2'
```

に

```
classpath 'com.android.tools.build:gradle-experimental:0.7.2'
```

v0.7.2はではだったので、バージョンをでチェックし、それにわせてラインをしてください

build.gradleファイルはのようになります。

```
buildscript {
    repositories {
        jcenter()
    }
```

```
    dependencies {
        classpath 'com.android.tools.build:gradle-experimental:0.7.2'
    }
}

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

MyApp / app / build.gradleファイルをする

build.gradleファイルをののようにします。おいのバージョンがなるがあります。

```
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'])
}
```

するに、Gradleファイルにエラーがないことをしてします。

プラグインがかどうかをテストする

まず、Android NDKモジューールをダウンロードしてください。 AndroidStudioでしいアプリをし、ActivityMainファイルにのをします

```
public class MainActivity implements Activity {
    onCreate() {
        // Pregenerated code. Not important here
    }
    static {
        System.loadLibrary("myLib");
    }
    public static native String getString();
}
```

`getString()`は、するJNIがつからないというでされます。いがされるまで、びしのにマウスをします。をクリックして、 `create function JNI_...`を`create function JNI_...`。これにより、しいJNIびしでmyApp / app / src / main / jniディレクトリにmyLib.cファイルがされます。これはのようになります。

```
#include <jni.h>

JNIEXPORT jstring JNICALL
Java_com_example_mydomain_myapp_MainActivity_getString(JNIEnv *env, jobject instance)
{
    // TODO

    return (*env)->NewStringUTF(env, returnValue);
}
```

されないは、プラグインがしくされていないか、NDKがダウンロードされていません

すべての**gradle**プロジェクトタスクをする

```
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
```

に "unaligned" apkをする

にされたapkファイルの`unaligned`おそらくそうではないがないは、 `build.gradle`ファイルにのコードをすることができます

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

ここから

ビルドバリアントをする

らかので、ビルドバリアントをすることができます。たとえば、のをし、ユニット/テストなどのデバッグでのみします。

プロジェクトから**mockRelease**をしましょう。 **build.gradle**ファイルをいてのようにします。

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

ツリーをる

タスクのをします。あなたのモジュールがされているにじて、それがいずれであってもよい `./gradlew dependencies`やモジュールのアプリのをするために`./gradlew :app:dependencies`

build.gradleファイルのの

```
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'
}
```

のグラフがされます。

```
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 (*)

. . .
```

ここでは、プロジェクトが`com.android.support:design`バージョン23.2.1をんでいることがわかります。それ`com.android.support:support-v4`がバージョン23.2.1になります。しかし、`com.google.android.gms:play-services`は、じ`support-v4` -v4にしていますが、いバージョン21.0.0では、gradleによってされたです。

`(*)`は、それらのがすでににリストされているため、gradleがサブツリーをスキップするときにされます。

versionnumber / buildconfigurationsにgradle.propertiesをする

のをすることができます

- したgradleインクルードファイル"dependencies.gradle"ファイルによるの
- スタンドアロンのプロパティファイル"version.properties"ファイルでのビルドのバージョン

root `gradle.properties`ファイルを`gradle.properties`

プロジェクトの

```
root
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
  +- gradle.properties
```

gradle.propertiesのすべてのサブモジュールのグローバル

```
# used for manifest
# todo increment for every release
appVersionCode=19
appVersionName=0.5.2.160726

# android tools settings
appCompileSdkVersion=23
appBuildToolsVersion=23.0.2
```

サブモジュールでの

```
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 {
    ...
}
```

F-Droidアプリストアにあなたのアプリをしたいは、f-droidロボットがのバージョンナンバーをんでバージョンのを/することができないため、gradleファイルのマジックナンバーをするがあります。

をする

によってはたとえば、Google APIキーをするなど、キーストアのフィンガープリントをつけるがあります。 Gradleには、キーストアのフィンガープリントをむすべてのをするなタスクがありま

す。

```
./gradlew signingReport
```

これはサンプルです。

```
: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
----------
```

ビルドタイプの

`android {}`ブロックのモジュールレベルの`build.gradle`ファイルでビルドタイプをおよびできます。

```
    android {
        ...
        defaultConfig {...}

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

            debug {
                applicationIdSuffix ".debug"
```

```
            }
        }
    }
```

オンラインでAndroidGradleをむ https://riptutorial.com/ja/android/topic/95/androidgradle

40: API-23 +におけるランタイムパーミッション

き

Android MarshmallowはRuntime Permissionモデルをしました。は、 となの 2つのカテゴリにされます。ここでなパーミッションはにユーザによってえられます。

sdk 23からAndroidは、なグループとしてされているで、Android 6.0をしているデバイスのにするをとします。なグループは、ユーザのプライバシーおよび/またはセキュリティをするとみなされるグループです。

なグループのリストはのとおりです。

なグループ

グループ
カレンダー
カメラ
おいわせ
ロケーション
マイクロフォン

センサ
SMS
ストレージ

これらのグループからのアクセスには、Android 6.0のデバイスのアクセスのがです。のsdkは23です。

のアクセス

に、のアクセスのをします。これらはユーザのプライバシーやセキュリティにとってではないとえられているため、sdk 23のランタイムはありません。

ACCESS_LOCATION_EXTRA_COMMANDS
ACCESS_NETWORK_STATE
ACCESS_NOTIFICATION_POLICY
ACCESS_WIFI_STATE
ブルートゥース
BLUETOOTH_ADMIN
BROADCAST_STICKY
CHANGE_NETWORK_STATE
CHANGE_WIFI_MULTICAST_STATE
CHANGE_WIFI_STATE

DISABLE_KEYGUARD
EXPAND_STATUS_BAR
GET_PACKAGE_SIZE
INSTALL_SHORTCUT
ィンターネット
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
ましをする
SET_TIME_ZONE
SET_WALLPAPER
SET_WALLPAPER_HINTS
TRANSMIT_IR
UNINSTALL_SHORTCUT
USE_FINGERPRINT

WAKE_LOCK
WRITE_SYNC_SETTINGS

Examples

Android 6.0のの

このは、Android 6でにをチェックするをしています。

```
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;
        }
    }
}
```

ブロードキャスト、URIでのアクセスの

インテントをされたブロードキャストレシーバにするときは、パーミッションチェックをうことができます。するパーミッションは、そのタグのにされているものとクロスチェックされます。らは、するにブロードキャストをできるをします。

パーミッションをつブロードキャストリクエストをするには、`Context.sendBroadcast(Intent intent, String permission)`コールでそのパーミッションをとしてしますが、のアプリはブロードキャストをするためにそのパーミッションをたなければならないことにしてください。はにのにインストールするがあります。

メソッドのシグネチャはのとおりです。

```
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");
```

ブロードキャストがsendBroadcastをしてされたされたアクセスをめるがあることをマニフェストですることができます。

```
<!--  Your special permission -->
```

```
<permission android:name="org.quadcore.mypermission"
   android:label="my_permission"
   android:protectionLevel="dangerous"></permission>
```

また、このブロードキャストをするはずのアプリケーションのマニフェストでパーミッションをします

```
<!--  I use the permission ! -->
<uses-permission android:name="org.quadcore.mypermission"/>
<!-- along with the receiver -->
<receiver android:name="Bcastreceiver" android:exported="true" />
```

とのにがなは、のチェックがされ、けられたターゲットにされるがあります。パーミッションをするAppをにインストールするがあります。

については、 こちらをごください 。

じアクセスグループからののアクセス

マニフェストには、2つのグループから4つのなアクセスがあります。

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

アクセスがなアクティビティ。アプリがバックグラウンドのときにをりしてアプリがクラッシュするがあるため、がなすべてのアクティビティでをすることがです。

```
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();
    }
```

グループからこれらのの1つだけをするがあり、このグループののすべてのは、ユーザーがをりさないりされます。

```
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;
}
```

これは、ユーザーがをするかどうかのをします。このでは、がされていない、アプリケーションはされます。

```
@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);
    }
}
```

https://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en

PermissionUtilの

PermissionUtilは、コンテキストでパーミッションをめるでなです。されたすべてのアクセス `onAllGranted()` 、がされた `onAnyDenied()` 、または `onRational()` がなに、がこるべきかをにできます。

AppCompatActivityまたはFragmentのどこにいても、ユーザーのをめる

```
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);
```

これを`onRequestPermissionsResult`します

```
if(mRequestObject!=null){
    mRequestObject.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

されたアクセスをAndroidManifest.xmlにもします

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

すべてのアクセスのコードをクラスにめ、このクラスのアクティビティをして、よりクリーンでなコードをする

```
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);
    }
}
```

アクティビティの

アクティビティは、でしたクラスをのようにするがあります。

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

オンラインでAPI-23 +におけるランタイムパーミッションをむ
https://riptutorial.com/ja/android/topic/1525/api-23-plusにおけるランタイムパーミッション

41: AsyncTask

パラメーター

パラメータ	
Params	にタスクにされるパラメータのタイプ。
	バックグラウンドにされたのタイプ
	バックグラウンドのの。

Examples

な

Androidのアクティビティとサービスでは 、ほとんどのコールバックはメインスレッドでされます 。これは、UIをすることがになりますが、メインスレッドで、プロセッサにまたはI / O-いタスクをすると、あなたのUIがし、しなくなることがありますドキュメント 、にがこるかを。

これらのいタスクをバックグラウンドスレッドにくことでこれをできます。

これをうための1つのは、バックグラウンドスレッドのいをにするためのフレームワークをするAsyncTaskをすることです。バックグラウンドスレッドがをする、およびにUIスレッドタスクをします。

`AsyncTask`するときにオーバーライドできるメソッド

- `onPreExecute()` タスクがされるに**UI**スレッドでびされます。
- `doInBackground()` `onPreExecute()`がしたにバックグラウンドスレッドでびされます。
- `onProgressUpdate()` `publishProgress(Progress...)`びしに**UI**スレッドでびされ `publishProgress(Progress...)` 。
- `onPostExecute()` バックグラウンドがしたに**UI**スレッドでびされます。

```
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.");
        }
    }

}
```

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

またはに

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

`AsyncTask`をする`AsyncTask`は、 `< >`に3つのをすことができます。
`<Params, Progress, Result>` パラメーターセクションを としてされています。

のでは`<File, Void, String>`をしています

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

`Void`は、をとしてマークするにされます。

プリミティブ `int` 、 `float`などをパラメータとしてすことはできません。そのようなは、 ラッパークラスをすがあります 。たとえば、 `int`わりに`Integer` 、 `Float`わりに`float`です。

AsyncTaskおよび**Activity**のライフサイクル

AsyncTasksはアクティビティインスタンスのライフサイクルにいません。アクティビティでAsyncTaskをし、デバイスをさせると、アクティビティがされ、しいインスタンスがされます。しかし、AsyncTaskはぬことはありません。それはするまできけるでしょう。

ソリューションAsyncTaskLoader

つのサブクラスローダーは AsyncTaskLoaderです。このクラスは、AsyncTaskとじをしますが、はるかにれています。アクティビティのをよりにでき、フラグメントやアクティビティのライフサイクルでします。いいことは、AsyncTaskLoaderは、AsyncTaskがされているどのようなでもできることです。 Activity / Fragmentがするためにいつでもデータをメモリにロードするがある、AsyncTaskLoaderはよりいをすることができます。

AsyncTaskのキャンセル

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

これはの、あなたのタスクをめません。キャンセルされたフラグをします。これは、`isCancelled()`りをチェックすることによってチェックできますコードがであるとします。

```
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;
            }
        }
    }
}
```

、AsyncTaskがキャンセルされた`doInBackground(Params... params)`まだ、そのされている`onPostExecute(Result result)`

にびされません `doInBackground(Params... params)` ります。 AsyncTaskはわりに`onCancelled(Result result)`をびして、にタスクがキャンセルされたことをします。

の

には、 `AsyncTask`によってわれるのをするがあります。このは、、などですことができます。これをうには、2つのをするがあります。まず、するが`onProgressUpdate`そのパラメータののののパラメータとじである`AsyncTask`。

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

に、々はをするが`publishProgress`ずしも`doInBackground`、およびそれがすべてで、のでは、すべてのをいます。

```
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;
 }
```

AndroidでAsyncTaskをしてをダウンロードする

このチュートリアルでは、AndroidでAsyncTaskをしてイメージをダウンロードするについてします。のでは、ダウンロードにプログレスバーをしながらイメージをダウンロードしています。

Android AsyncTaskについて

タスクをすると、Handsをスレッドにしなくてもマルチスレッドをできます。 AsyncTaskをすると、UIスレッドをかつにできます。バックグラウンドをし、UIスレッドでをすことができます。UIにするかをしている、えばリストにするデータをダウンロードするは、AsyncTaskをしてください。

- AsyncTasksは、のにはにするのがです。
- タスクは、Params、ProgressおよびResult、および`onPreExecute()` 、 `doInBackground()` 、 `onProgressUpdate()`および`onPostExecute()`とばれる3つのジェネリックタイプによってされます。
- `onPreExecute()`では、バックグラウンドをするにするがあるコードをすることができます。
- doInBackgroundにはバックグラウンドでするのあるコードがあります。ここでは

`doInBackground()` publishProgressメソッドでをイベントスレッドにし、バックグラウンドがしたことをしてをすことができます。

- `onProgressUpdate()`メソッドは、 `doInBackground()`メソッドをしてされている`publishProgress()`メソッドからのをけり、このメソッドはこのをしてイベントスレッド
- `onPostExecute()`メソッドは、 `doInBackground()`メソッドによってされたをします。
- されるジェネリックタイプは、
  - Params、にタスクにされるパラメ―タのタイプ
  - バックグラウンドにされたのタイプ。
  - 、バックグラウンドののタイプ。
- タスクがタイプをしていないは、Voidタイプとしてマ―クすることができます。
- のタスクは、 `cancel(boolean)`メソッドをびす`cancel(boolean)`によって`cancel(boolean)`ことができます。

Android AsyncTaskをしてをダウンロ―ドする

.xmlレイアウト

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

<Button
    android:id="@+id/downloadButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Click Here to Download" />

<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:contentDescription="Your image will appear here" />

</LinearLayout>
```

.javaクラス

```
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;
        }
    }
}
```

のところ、のためのコメントフィ―ルドがないまたははそれをつけられていないか、またははそれのためのをていないので、ここにいくつかのコメントがあります

これは、AsyncTaskでができるかをすいです。

しかし、このでは、

- メモリリ―クの
- タスクがするにがした、アプリケ―ションがクラッシュする

は、をしてください。

- WeakReferenceとしてアクティビティをしてメモリリークをする
- http://stackoverflow.com/documentation/android/117/asynctask/5377/possible-problems-with-inner-async-tasks
- AsyncTaskでアクティビティをらさないようにする

WeakReferenceとしてアクティビティをしてメモリリークをする

AsyncTaskは、それをびすアクティビティへのをするのがです。

AsyncTaskがActivityのクラスである、AsyncTaskとそのメンバー/メソッドをできます。

ただし、AsyncTaskがActivityのクラスでないは、ActivityをAsyncTaskにすがあります。これをうと、するのあるの1つは、AsyncTaskがそのバックグラウンドスレッドでをするまで、AsyncTaskがアクティビティのをすることです。 AsyncTaskのバックグラウンドスレッドのがするにアクティビティがまたはすると、AsyncTaskはアクティビティへのをしているため、ガベージコレクションできません。

その、メモリリークがします。

これをぐには、アクティビティへのではなくAsyncTaskでWeakReferenceをします。

WeakReferenceをするAsyncTaskのをにします。

```
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();
```

```
        }
    }
}
```

アクティビティから**AsyncTask**をびす

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

フラグメントから**AsyncTask**をびす

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

にされたとき、 `AsyncTasks`は1つのバックグラウンドスレッドでしてされまし た。 `DONUT`からめて、これはスレッドのプ―ルにされ、のタスクをにさせることができまし た。 `HONEYCOMB`すると、タスクは1つのスレッドでされ、によってするなアプリケ―ションエラ―をします。

あなたがにしたいは、びすことができます`executeOnExecutor(java.util.concurrent.Executor, Object[])`と`THREAD_POOL_EXECUTOR`。

SERIAL_EXECUTOR - >に1つのタスクをしてするExecutor。

THREAD_POOL_EXECUTOR - >タスクをしてするためにできるExecutor。

サンプル

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

AsyncTaskタスクのと

AsyncTaskはクラスであり、 `Thread`クラスをしません。 メソッド`doInBackground(Params... params)`があり、タスクをするためにオ―バ―ライドされています。このメソッドは`AsyncTask.call()`からびされます。

Executorは`java.util.concurrent`パッケ―ジのです。

さらに、AsyncTaskには2つの`Executor`がまれています

`THREAD_POOL_EXECUTOR`

ワ―カ―スレッドをしてタスクをしてします。

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

SERIAL_EXECUTOR

それは、タスクをに、すなわち1つずつする。

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

どちらの`Executor` sがであり、それゆえっだけ`THREAD_POOL_EXECUTOR`と1 `SerialExecutor`オブジェクトがしますが、あなたはいくつかすることができ`AsyncTask`オブジェクトを。

したがって、デフォルトのExecutor `SerialExecutor` でのバックグラウンドタスクをしようとすると、これらのタスクはキュ―にれられ、されます。

`THREAD_POOL_EXECUTOR`でのバックグラウンドタスクをしようとすると、それらはにされます。

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

Performボタンをクリックすると、タスクがされ、がされます。

スレッドプール1でされたタスク

タスクはするのに1000msかかる。

t = 36sでは、タスク2,3,4はしてされているためキューにれられ、もされます。

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

コメント`Task Executed in thread pool` 1で`Task executed Serially` 2で`Task executed Serially`コメントをします。

Performボタンをクリックすると、タスクがされ、がされます。

にタスクをしているため、のタスクのがしたにすべてのタスクがされます。したがって、タスク1

のがすると、タスク2のみがバックグラウンドでをします。に。

```
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
```

オンラインでAsyncTaskをむ https://riptutorial.com/ja/android/topic/117/asynctask

42: AudioManager

Examples

なオーディオフォーカスの

```
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);
        }
    }
}
```

オーディオフォーカスのリクエスト

```
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.
        }
    }
}
```

オンラインでAudioManagerをむ https://riptutorial.com/ja/android/topic/6798/audiomanager

43: AutoCompleteTextView

なテキストフィールドにするときにユーザーにをするは、 `AutoCompleteTextView`をできます。ユーザーがににをします。のリストは、ユーザがボックスのをきえるものをできるドロップダウンメニューにされる。

Examples

シンプルでハードコーディングされた**AutoCompleteTextView**

デザインレイアウト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" />
```

`setContentView()` またはそのフラグメントまたはカスタムビューにするのにコードのビューをつける

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

ハードコードされたデータをアダプタでする

```
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);
```

ヒントこのようなハードコーディングされたリストのわりに、あるの`Loader`でデータをするのがましいです。

CustomAdapter、ClickListener、および**Filter**をしたオートコンプリート

メインレイアウト**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>
```

レイアウト**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;
    }
}
```

モデル クラス**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;
    }
}
```

アダプタクラス**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();
                }
            }
        }
    };
}
```

オンラインでAutoCompleteTextViewをむ
https://riptutorial.com/ja/android/topic/5300/autocompletetextview

44: BluetoothとBluetooth LE API

Bluetooth Classicは、Android 2.0APIレベル5でごいただけます。 Bluetooth LEは、Android 4.3 APIレベル18からごいただけます。

Examples

アクセス

アプリケーションでBluetoothをするには、このをマニフェストファイルにします。

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

デバイスのをしたり、Bluetoothをするがあるは、のもするがあります。

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

Android APIレベル23をターゲティングするには、のアクセスがです。

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

*をにするのについては、 のトピックをしてください。

Bluetoothがになっているかどうかをする

```
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
```

```
        }
    }
}
```

デバイスをにする

```
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
        }
    }
}
```

くの**Bluetooth**デバイスをす

に`BluetoothAdapter`し`BluetoothAdapter`。

```
BluetoothAdapter mBluetoothAdapter;
```

に、 `ACTION_FOUND`の`BroadcastReceiver`をします。

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

```
};
```

`BroadcastReceiver`する

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

に、 `startDiscovery`をびして、くのBluetoothデバイスのをします

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

をすることをれないでください`BroadcastReceiver`に`onDestroy`

```
unregisterReceiver(mReceiver);
```

Bluetoothデバイスにする

BluetoothDeviceをしたら、それとできます。ソケット\ストリームをってわれるこのの

これらは、Bluetoothののためのなです。

1ソケットをする

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

2ソケットにする

```
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ソケット**/**スト リームの

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

ストリーム - データチャネルとしてされますデバイスからのデータ

ストリーム - データチャネルとしてデバイスにデータを

3のステップがすると、にされたストリームをして、のデバイスでデータをできます。

*1*データソケットストリームからのみみ

```
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*データのストリームへのきみ

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

- もちろん、、みきはのスレッドでうがあります。
- ソケットとストリームオブジェクトがです

くの**Bluetooth**エネルギーデバイスをつける

しかし、デバイスのスキャンはAPI 21でされています。デバイスのは、スキャンするサービスUUID にされている16ビットUUIDまたはのものをすることからめるがあります。 。このでは、APIをBLEデバイスをするためのしたをします。

1. Bluetoothデバイスモデルをする

```
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. Bluetoothスキャンインターフェイスをする

```
public interface ScanningAdapter {

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

3. スキャンファクトリクラスをする

```
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. APIのファクトリをする

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. びしてデバイスのリストをする

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

wait few seconds...

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

オンラインでBluetoothとBluetooth LE APIをむ
https://riptutorial.com/ja/android/topic/2462/bluetoothとbluetooth-le-api

45: Bluetoothエネルギー

き

このドキュメントは、 のドキュメントをしたものであり、Android 5.0API 21でされたの Bluetooth LE APIにをてています。とののと、スキャンとのをするについてします。

Examples

BLEデバイスの

Bluetooth APIをするには、のがです。

```
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
```

Android 6.0 **API**レベル**23** のをターゲティングしていて、スキャン/をするは、のがです。

```
android.permission.ACCESS_FINE_LOCATION
```

または

```
android.permission.ACCESS_COARSE_LOCATION
```

Android 6.0APIレベル**23**のデバイスでも、サービスをにするがあります。

スキャン/をするには、BluetoothAdapterオブジェクトがです。

```
BluetoothManager bluetoothManager = (BluetoothManager)
context.getSystemService(Context.BLUETOOTH_SERVICE);
bluetoothAdapter = bluetoothManager.getAdapter();
```

BluetoothLeScannerクラスの`startScan (ScanCallback callback)`メソッドは、スキャンをするもなです。 をけるには、 `ScanCallback`オブジェクトがです。

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

GATTサーバへの

のBluetoothDeviceオブジェクトをしたら、そのオブジェクトにパラメータとして`connectGatt()`メソッドをしてできます。このメソッドは、BLEデバイスににするかどうかをすブールとイベントとクライアントがまれるBluetoothGattCallbackはされます

```
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        device.connectGatt(context, false, bluetoothGattCallback,
BluetoothDevice.TRANSPORT_AUTO);
    } else {
        device.connectGatt(context, false, bluetoothGattCallback);
    }
```

BluetoothGattCallbackの`onConnectionStateChange`をオーバーライドしてをします。

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

へのきみとみり

Gatt Serverにすると、サーバーのをきんだりんだりすることで、Gatt Serverとやりとりすることになります。これをうには、まずこのサーバーでなサービスとサービスでできるをするがあります。

```
 @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
                    }
```

```
                }
            }
        }
```

なきみはのようになります。

```
characteristic.setValue(newValue);
characteristic.setWriteType(BluetoothGattCharacteristic.WRITE_TYPE_DEFAULT);
gatt.writeCharacteristic(characteristic);
```

きみがすると、 `BluetoothGattCallback onCharacteristicWrite`メソッドがびされます。

```
    @Override
    public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
        super.onCharacteristicWrite(gatt, characteristic, status);
        Log.d(TAG, "Characteristic " + characteristic.getUuid() + " written);
    }
```

なきみはのようになります。

```
gatt.readCharacteristic(characteristic);
```

きみがすると、 `BluetoothGattCallback onCharacteristicRead`メソッドがびされます。

```
@Override
public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
  super.onCharacteristicRead(gatt, characteristic, status);
  byte[] value = characteristic.getValue();
  }
```

ガットサ―バ―からのの

のがされたときに、Gatt Serverにするようリクエストできます。

```
gatt.setCharacteristicNotification(characteristic, true);
BluetoothGattDescriptor descriptor = characteristic.getDescriptor(
    UUID.fromString("00002902-0000-1000-8000-00805f9b34fb");
descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
mBluetoothGatt.writeDescriptor(descriptor);
```

サ―バ―からのすべてのは、BluetoothGattCallbackの`onCharacteristicChanged`メソッドでされます。

```
    @Override
    public void onCharacteristicChanged(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic) {
        super.onCharacteristicChanged(gatt, characteristic);
        byte[] newValue = characteristic.getValue();
}
```

BLEデバイスの

Bluetooth LE Advertisingをすると、まずをせずに、くのすべてのデバイスにデ―タパッケ―ジをブロ―ドキャストできます。 31バイトのデ―タにはしいがあることにしてください。あなたのデバイスをすることは、のユ―ザ―があなたにできるようにするためのです。

すべてのデバイスがBluetooth LE Advertisingをサポ―トしているわけではないため、に、デバイスがサポ―トするためになすべてのをたしていることをするがあります。その、`BluetoothLeAdvertiser`オブジェクトをすることができ、をすることができます

```
        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);
        }
```

ガットサ―バ―の

デバイスをペリフェラルとしてさせるには、まず`BluetoothGattServer`をいて、なくとも1つの`BluetoothGattService`と1つの`BluetoothGattCharacteristic`をするがあり`BluetoothGattCharacteristic`。

```
BluetoothGattServer server=bluetoothManager.openGattServer(context,
bluetoothGattServerCallback);
```

```
BluetoothGattService service = new BluetoothGattService(SERVICE_UUID,
BluetoothGattService.SERVICE_TYPE_PRIMARY);
```

これは、フルきみ、みり、のパーミッションをつBluetoothGattCharacteristicのです。にじて、このをするをかくすることができます。

```
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`は、 `BluetoothGattServer`するすべてのイベントをします。

```
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);
                }
```

またはへのみ/みのをけると、そのがにするためにをするがあります。

```
@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);
}
```

オンラインでBluetoothエネルギーをむ https://riptutorial.com/ja/android/topic/10020/bluetoothエネルギー

46: BroadcastReceiver

き

BroadcastReceiverレシーバは、システムイベントやアプリケーションイベントにできるAndroidコンポーネントです。このイベントがすると、Androidランタイムは、されたすべてのイベントのにします。

えば、がしたこと、がっていること、またはがりまれたことをらせるである。

アプリケーションは、ブロードキャストをすることもできます。たとえば、のデータがデバイスにダウンロードされ、できるようになっていることをのアプリケーションにらせることができます。

Examples

の

ブロードキャストは、システムまたはアプリケーションのイベントにできるAndroidコンポーネントです。

レシーバは、 `AndroidManifest.xml`ファイルをしてすることも、 `Context.registerReceiver()`メソッドをしてにすることもできます。

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

ここでは、Androidがブートプロセスをすると、システムによってされる`ACTION_BOOT_COMPLETED`をりげました。

のようにマニフェストファイルにレシーバをすることができます

```
<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>
```

デバイスがされると、 `onReceive()`メソッドがびされ、にをうことができますえば、サーービスの、アラームのなど。

BroadcastReceiverの

BroadcastReceiversは、Android OS、のアプリ、またはじアプリでされたブロードキャストインテントをするためにされます。

インテントはインテントフィルターでされますが 、これにはストリングアクションがです。インテントにをすることができます。

に、BroadcastReceiversは、のインテントフィルターでインテントをするようにします。プログラムですることができます

```
mContext.registerReceiver(new BroadcastReceiver() {
    @Override
   public void onReceive(Context context, Intent intent) {
      //Your implementation goes here.
   }
}, new IntentFilter("Some Action"));
```

または`AndroidManifest.xml`ファイルにします。

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="Some Action"/>
    </intent-filter>
</receiver>
```

インテントをけるには、アクションをAndroid OS、のアプリまたはAPI、またはのアプリケーションで`sendBroadcast`をしてしたものにします。

```
mContext.sendBroadcast(new Intent("Some Action"));
```

さらに、インテントは、でることができる、プリミティブ、び*Parcelables*、などのをむことができる`onReceive`。

LocalBroadcastManagerの

*LocalBroadcastManager*は、ブロードキャストするためにされるインテントをまないリスナーにそれらをさらすことなく、アプリケーションで。

*LocalBroadcastManager*をすると、 `context.sendBroadcast()`するよりでです。これは、のアプリケーションによってされたブロードキャストをするがないため、セキュリティのがじるがあります。

ローカルブロードキャストののなをにします。

```
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);
```

Bluetoothブロードキャスト

マニフェストファイルにをする

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

あなたのフラグメントまたはアクティビティ

- メソッドをする

```
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;
    }
};
```

ブロードキャストをする

- onResumeでこのメソッドをびす

```
private void registerBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    manager.registerReceiver(mBluetoothStatusChangedReceiver, new
IntentFilter(Constants.BROADCAST_BLUETOOTH_STATE));
}
```

ブロードキャストのをする

- onPauseでこのメソッドをびします。

```
private void unregisterBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    // Beacon□□□
    manager.unregisterReceiver(mBluetoothStatusChangedReceiver);
}
```

ブロードキャストのプログラムによると

`BroadcastReceiver`をまたはにするには、 `PackageManager`へのをするがあり`ComponentName`/にするのクラスをむ`ComponentName`オブジェクトがです。

```
ComponentName componentName = new ComponentName(context, MyBroadcastReceiver.class);
PackageManager packageManager = context.getPackageManager();
```

ここで、 `BroadcastReceiver`をにするために、のメソッドをびすことができます

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP);
```

わりに、 `COMPONENT_ENABLED_STATE_DISABLED`をしてをにすることもできます。

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
    PackageManager.DONT_KILL_APP);
```

BOOT_COMPLETEDイベントをするBroadcastReceiver

のは、 `BOOT_COMPLETED`イベントをけることができる`BroadcastReceiver`をするをしています。このようにして、すぐにデバイスをしたときに`Service`をしたり、 `Activity`をすることができます。

また、 `BOOT_COMPLETED`イベントをすると、デバイスのがオフのときにアラームがされるため、アラームをできます。

`BOOT_COMPLETED`アクションをけるには、なくとも1はアプリケーションをするがあります。

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
            }
        }
    }
}
```

LocalBroadcastManagerの

BroadcastReceiverは、に、のアクションをするためにOSをしてインテントをリレーするメカニズムです。な

> 「は、システムイベントやアプリケーションイベントにできるAndroidコンポーネントです。

LocalBroadcastManagerは、アプリケーションプロセスでブロードキャストをするです。このメカニズムにはくのがあります

1. データはアプリケーションプロセスにるため、データがすることはありません。
2. ののは、OSでにするため、LocalBroadcastはよりくされます。

LocalBroastManagerのなはのとおりです。

SenderActivity

```
  Intent intent = new Intent("anEvent");
  intent.putExtra("key", "This is an event");
  LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
```

ReceiverActivity

1. をする

```
LocalBroadcastManager.getInstance(this).registerReceiver(aLBReceiver,
               new IntentFilter("anEvent"));
```

2. がびされたときにアクションをするなオブジェクト

```
private BroadcastReceiver aLBReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        // perform action here.
    }
};
```

3. ビュ―がもうされなくなったときにをします。

```
@Override
protected void onPause() {
    // Unregister since the activity is about to be closed.
    LocalBroadcastManager.getInstance(this).unregisterReceiver(aLBReceiver);
    super.onDestroy();
}
```

カスタムブロ―ドキャストをして2つのアクティビティをする

アクティビティ―Aがアクティビティ―Bでこっているイベントをできるように、2つのアクティビティ―をできます。

アクティビティ**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
    }
};
```

B

```
final String eventName = "your.package.goes.here.EVENT";

private void launchEvent() {
    Intent eventIntent = new Intent(eventName);
    this.sendBroadcast(eventIntent);
}
```

もちろん、アクティビティでされるインテントにエクストラをするブロードキャストにをすることもできます。をできるだけシンプルにつためにされていません。

スティッキーブロードキャスト

sendStickyBroadcastメソッドインテントをしている、するインテントはスティッキーです。つまり、ブロードキャストがしたにするインテントがります。がすようにStickyBroadcastは、ブロードキャストがした、ブロードキャストからデータをみるメカニズムです。これは、 `Activity's onCreate()`に、 `Activity's onCreate()`インテントのキーのをしたいにできます。

```
Intent intent = new Intent("com.org.action");
intent.putExtra("anIntegerKey", 0);
sendStickyBroadcast(intent);
```

きブロードキャストの

きブロードキャストは、ブロードキャストリスナーのをするがあるにされます。

このでは、 `firstReceiver`はに`secondReceiver`よりに`firstReceiver`をします。

```
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);
```

さらに、は、きをすることができる。

```
@Override
public void onReceive(final Context context, final Intent intent) {
    abortBroadcast();
}
```

この、のいすべてのはブロードキャストメッセージをしません。

Android

Android 3.1、すべてのアプリケーションはインストールにになります。では、アプリケーションはらかのでされません。ただし、アクティビティのまたはアクティビティ、サービス、またはブロードキャストにするなインテントはされません。

APKをインストールするシステムアプリをするときは、しくインストールされたAPPがしていないにするまでブロードキャストをしないことをしてください。

アプリをにするなは、このアプリになブロードキャストをすることです。ほとんどのアプリは`INSTALL_REFERRER`をしているので、これをフックポイントとしてできます

インストールされているアプリのマニフェストをスキャンし、なブロードキャストをにします

```
Intent intent = new Intent();
intent.addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES);
intent.setComponent(new ComponentName(packageName, fullClassName));
sendBroadcast(intent);
```

オンラインでBroadcastReceiverをむ
https://riptutorial.com/ja/android/topic/1460/broadcastreceiver

47: CardView

き

がく、とがいFrameLayout。

CardViewはLollipopのプロパティをとしてし、のプラットフォームではカスタムのエミュレートされたのにります。

Lollipopよりのプラットフォームでは、いコーナークリッピングのなのため、CardViewはをめてするをりらない。わりに、このようなをけるためにパディングをしますこのをするにはsetPreventCornerOverlapbooleanを。

パラメーター

パラメータ	
cardBackgroundColor	CardViewのです。
cardCornerRadius	CardViewのコーナー。
cardElevation	CardViewの。
cardMaxElevation	CardViewの。
cardPreventCornerOverlap	Cardコンテンツとみのあるのをぐため、v20のCardViewにパディングをします。
cardUseCompatPadding	のバージョンとじをつように、API v21 +にパディングをします。 "true"や "false"などのブールでもいません。
contentPadding	カードのとCardViewのとののパディング
contentPaddingBottom	カードのとCardViewのとののパディング。
contentPaddingLeft	カードのとCardViewのとののパディング。
contentPaddingRight	CardViewの。
cardElevation	カードのとCardViewのとののパディング。
contentPaddingTop	カードのとCardViewのとののパディング。

`CardView`は、LollipopAPI 21でのとなをします。しかし、Lollipop `CardView`がプログラムなシャドウ

ィンプリメ`CardView`ちるに、

`ImageView` `CardView`みをびたに`CardView`うとすると、それがLollipopAPI 21よりもしくえることがあります。このをするには、びすがあります`setPreventCornerOverlap(false)`あなたに`CardView`、または`app:cardPreventCornerOverlap="false"`レイアウトします。

`CardView`をするに、 `build.gradle`ファイルにサポートライブラリのをするがあります。

```
dependencies{
    compile 'com.android.support:cardview-v7:25.2.0'
}
```

いくつかのバージョンがここにあります

https://developer.android.com/reference/android/support/v7/widget/CardView.html
https://developer.android.com/training/material/lists-cards.html

Examples

CardView

`CardView`はAndroidサポートライブラリのメンバーであり、カードのレイアウトをします。

プロジェクトに`CardView`をするには、 `build.gradle`にのをします。

```
compile 'com.android.support:cardview-v7:25.1.1'
```

いくつかのバージョンがここにあります

あなたのレイアウトでは、のカードをすることができます。

```
<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>
```

このにのレイアウトをすることができ、それらはカードにまれます。

また、CardViewにはのUIがりまれ、 コードからできます 。

```
<?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>
```

CardViewのカスタマイズ

CardViewは、カードがプラットフォームでしたをつように、デフォルトのとのをします。

これらのデフォルトは、xmlファイルののをしてカスタマイズできます。

1. `card_view:cardElevation`は、CardViewのをします。
2. `card_view:cardBackgroundColor`は、CardViewののをカスタマイズするためにされますのをできます。
3. `card_view:cardCornerRadius`は、CardViewの4つのエッジをカーブさせるためにされます
4. `card_view:contentPadding`は、カードとカードのとののパディングをします。

card_viewは、のレイアウトビューでされたです。 xmlnscard_view = " http://schemas.android.com/apk/res-auto "

ここにをします。

```
<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>
```

また、をしてプログラムでうこともできます。

```
card.setCardBackgroundColor(....);
card.setCardElevation(...);
card.setRadius(....);
card.setContentPadding();
```

のプロパティについては、 javadocをしてください。

リップルアニメーションをする

CardViewでリップルアニメーションをにするには、のをします。

```
<android.support.v7.widget.CardView
  ...
  android:clickable="true"
  android:foreground="?android:attr/selectableItemBackground">
  ...
</android.support.v7.widget.CardView>
```

CardViewでをとしてするLollipopののデバイスの

CardViewでImage / Colorをバックグラウンドとしてしているは、にわずかないパディングデフォルトのカードのがのがすることがあります。これは、カードビューのデフォルトのいのためにします。 Pre-lollipopデバイスのマージンをけるはのとおりです。

`card_view:cardPreventCornerOverlap="false"`card_view:cardPreventCornerOverlap="false"をするがあります。 1。 XMLでは、のスニペットをします。

```
<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"
```

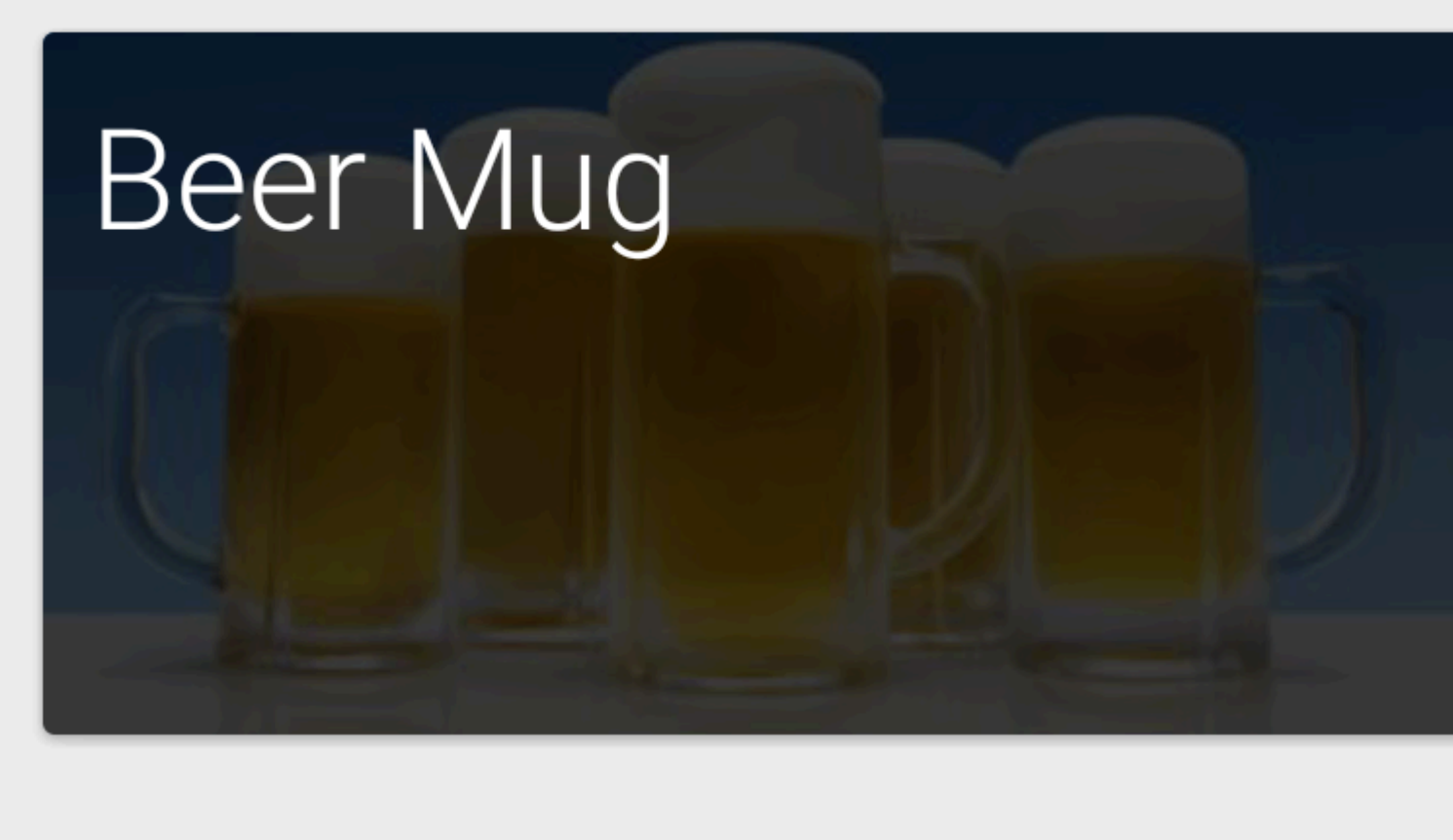
```
            android:layout_height="match_parent"
            android:adjustViewBounds="true"
            android:scaleType="centerCrop"
            android:src="@drawable/bg_image" />

</android.support.v7.widget.CardView>
```

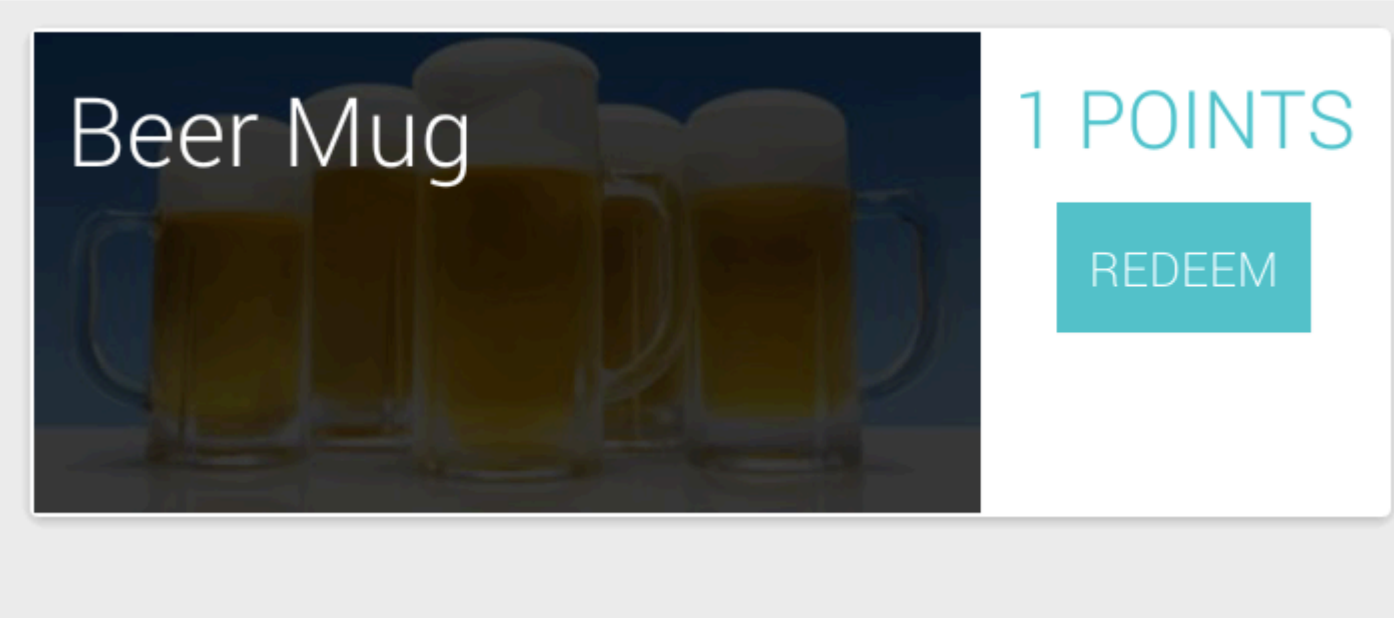
2. このようなJavaでは、 `cardView.setPreventCornerOverlap(false)` 。

そうすることで、カードのにあるなパディングがりかれます。このにするいくつかのなをします。

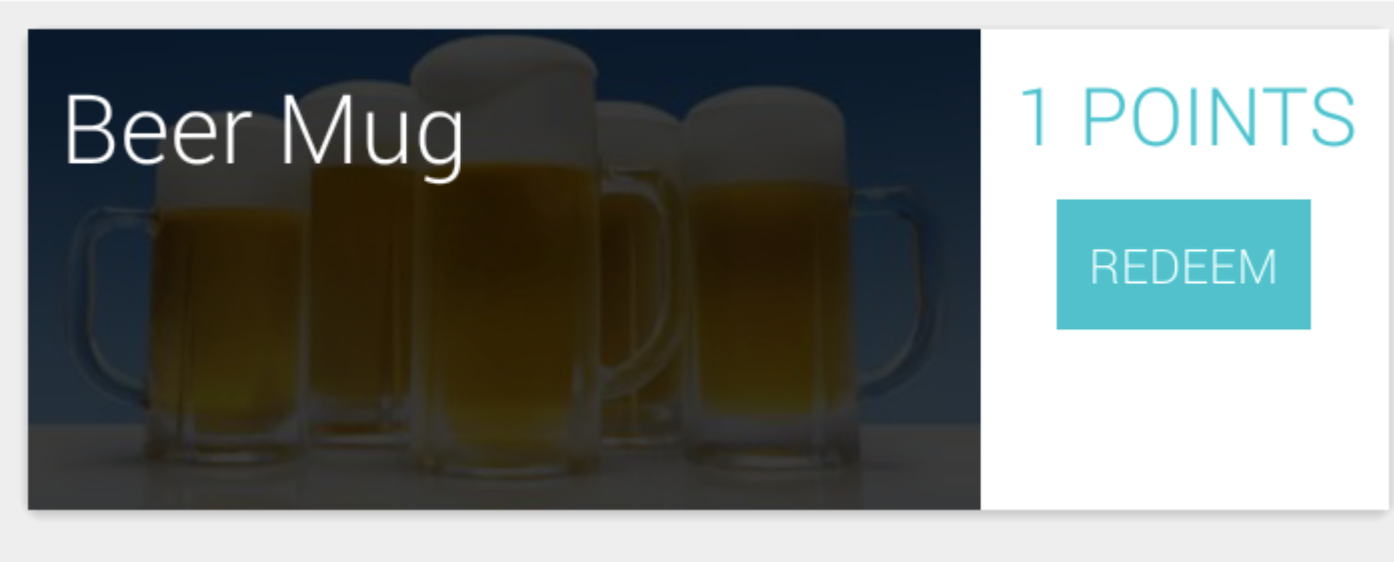
API 21のをつ**1**のカード にかい



2をたない API19のをつカード のりにパディングがあることにしてください



3 `cardView.setPreventCornerOverlap(false)` つ**API 19**のをつカード はされまし た



これについては、 こちらのドキュメントをしてください
オリジナルのSOFはこちら

TransitionDrawableでCardViewのをアニメートする

```
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);
}
```

オンラインでCardViewをむ https://riptutorial.com/ja/android/topic/726/cardview

48: CleverTap

き

CleverTapがするとエンゲージメントSDKのクイックハック - Android

https://clevertap.comからCleverTapをしてください 。

Examples

SDKのインスタンスをしてイベントをする

```
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
}
```

デバッグレベルの

カスタムアプリケーションクラスで、 `onCreate()` メソッドをオーバーライドし、のをします。

```
CleverTapAPI.setDebugLevel(1);
```

オンラインでCleverTapをむ https://riptutorial.com/ja/android/topic/9337/clevertap

49: ConstraintLayout

き

`ConstraintLayout`は`ViewGroup`で、ウィジェットをにし、サイズをできます。 Android 2.3APIレベル9にしています。

フラットなビューをして、きくなレイアウトをすることができます。 `RelativeLayout`とていますが、すべてのビューはビューとレイアウトのにってレイアウトされていますが、 `RelativeLayout`よりもで、Android Studioのレイアウトエディタでいやすくなっています。

- **ConstraintLayout**
  - public void addViewViewの、intインデックス、ViewGroup.LayoutParams params
  - public ConstraintLayout.LayoutParams generateLayoutParamsAttributeSet attrs
  - public void onViewAddedビューを
  - public void onViewRemovedビューを
  - public void removeViewビューの
  - public void requestLayout
  - されたブールcheckLayoutParamsViewGroup.LayoutParams params
  - ConstraintLayout.LayoutParams generateDefaultLayoutParamsをします。
  - ViewGroup.LayoutParams generateLayoutParamsをしましたViewGroup.LayoutParams params
  - されたvoid onLayoutboolean changed、int left、int top、int right、int bottom
  - されたvoid onMeasureint widthMeasureSpec、int heightMeasureSpec
- **ConstraintLayout.LayoutParams**
  - public void resolveLayoutDirectionint layoutDirection
  - public void validate
  - されたvoid setBaseAttributesTypedArray a、int widthAttr、int heightAttr

パラメーター

パラメータ	
	レイアウトにする`View`
	レイアウトの`View`のインデックス
パラメータ	`View`の`LayoutParams`
attrs	`AttributeSet`を`LayoutParams`
る	またはされた`View`
かわった	この`View`がサイズまたはをしたかどうかをします。
	`View`にする。
	`View`にするの
	`View`にする
	の、`View`にする
widthMeasureSpec	`View`によってされるスペース
heightMeasureSpec	`View`によってされるのスペース
layoutDirection	-
a	-
widthAttr	-
heightAttr	-

Google IO 2016では、GoogleがConstraintLayoutというのしいAndroidレイアウトをしました。のところ、このレイアウトはベータであるため、してください。

レイアウトの

https://codelabs.developers.google.com/codelabs/constraint-layout/index.html

Examples

プロジェクトにConstraintLayoutをする

ConstraintLayoutをするには、Androidスタジオバージョン2.2がで、バージョン32のAndroidサポートリポジトリがです。

1. Constraint Layoutライブラリを`build.gradle`ファイルのとしてします。

```
dependencies {
   compile 'com.android.support.constraint:constraint-layout:1.0.2'
}
```

2. プロジェクトをする

プロジェクトにしいレイアウトをするには

1. モジュールのレイアウトディレクトリをクリックし、[`New > XML > Layout XML.`] `New > XML > Layout XML.`] `New > XML > Layout XML.`]をクリックし`New > XML > Layout XML.`
2. レイアウトのをし、ルートタグに`"android.support.constraint.ConstraintLayout"`とします。
3. **Finish**をクリックします。

それのは、レイアウトファイルをしてください

```
<?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>
```

チェーン

`ConstraintLayout` alpha 9、 チェーンがです。 チェーンのビューのセットである`ConstraintLayout`と**B**に、すなわち、それらののようにされ、Bはのとにされています。

```
<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>
```

このでは、2つのビュー―がにされ、がにセンタリングされています。チェーンのバイアスをすることによって、これらのビュー―のをすることができます。チェーンののにのコードをします。

```
app:layout_constraintVertical_bias="0.2"
```

チェーンでは、のはのビュー―で、のチェーンではのビュー―です。のはチェーンのるいをします。

チェーンはであり、にされます。 ここにのAndroidがあります。

オンラインでConstraintLayoutをむ https://riptutorial.com/ja/android/topic/5076/constraintlayout

50: ConstraintSet

き

このクラスをすると、 `ConstraintLayout` でするのをプログラムでできます。これにより、をしてし、の`ConstraintLayout` することができます。

Examples

プログラムによる**ContraintLayout**による**ConstraintSet**

```
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
        }
    }
}
```

オンラインでConstraintSetをむ https://riptutorial.com/ja/android/topic/9334/constraintset

51: ContentProvider

コンテンツプロバイダは、されたデータセットへのアクセスをします。それらはデータをカプセルし、データセキュリティをするためのメカニズムをします。コンテンツプロバイダは、あるプロセスのデータをのプロセスでされているコードとするインタフェースです。

コンテンツプロバイダのデータにアクセスするには、アプリケーションの`Context`で`ContentResolver`オブジェクトをして、プロバイダとしてクライアントとしてします。`ContentResolver`オブジェクトは、`ContentProvider`をするクラスのインスタンスであるプロバイダオブジェクトとします。プロバイダオブジェクトは、クライアントからのデータをけり、されたをし、をします。

のアプリケーションとデータをしないは、のプロバイダをするはありません。ただし、のアプリケーションでカスタムをするには、のプロバイダがです。なデータやファイルをアプリケーションからのアプリケーションにコピーしてりけるは、のプロバイダもです。

Androidには、オーディオ、ビデオ、、のなどのデータをするコンテンツプロバイダがまれています。いくつかは`android.provider`パッケージのリファレンスドキュメントにされています。いくつかのがあり、これらのプロバイダにはどのAndroidアプリケーションからでもアクセスできます。

Examples

なコンテンツプロバイダクラスの

1クラスをする

コントラクトクラスは、アプリケーションがコンテンツURI、、インテントアクション、およびコンテンツプロバイダのそののをするのにつをします。クラスはプロバイダににはまれません。プロバイダのはそれらをし、のができるようにするがあります。

プロバイダには、1つのがあり、Androidのとしてします。のプロバイダとのをけるために、のコンテンツをしてください。このはAndroidパッケージにもてはまるため、プロバイダを、プロバイダをむパッケージのとしてすることができます。たとえば、Androidパッケージが`com.example.appname`、プロバイダに`com.example.appname.provider`というをえるがあります。

```
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";
}
```

コンテンツURIは、プロバイダのデータをするURIです。コンテンツURIには、プロバイダそののと、テーブルまたはファイルパスをすがまれます。オプションのidは、テーブルの々のをします。 ContentProviderのすべてのデータアクセスメソッドは、としてコンテンツURIをちます。これにより、アクセスするテーブル、、またはファイルをすることができます。クラスでこれらをし

ます。

```
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ヘルパークラスをする

ヘルパークラスは、データベースのとバージョンをします。

```
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.
    }
}
```

3ContentProviderクラスをするクラスをする

```
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は、とパスをにマップします。 `match()`メソッドは、URIのユニークなをします`match()`ユニークであれば、のをできます。 switchステートメントは、テーブルをするか、のレコードをするかをします。たちのUriMatcherは、URIがTableのContent URIであれば100をし、URIがそのテーブルののをすには101をします。 `#`ワイルドカードをしてのと`*`にさせ、のにさせることができます。

```
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;
}
```

`addURI()`びしはです UriMatcherはにされたものからにされたものまでにされます。 `#`や`*`ようなワイルドカ―ドはりなので、あなたがあなたのURIをしくしたことをするがあります。えば

```
uriMatcher.addURI(CONTENT_AUTHORITY, "/example", 1);
uriMatcher.addURI(CONTENT_AUTHORITY, "/*",       2);
```

マッチャ―は`/*`マッチにるにに`/example`をしますので、なです。これらのメソッドびしがし、`uriMatcher.match("/example")`びすと、UriMatcherは`/*`パスにってったをすとマッチをすのをめます

これらのをオ―バ―ライドするがあります

onCreate プロバイダをします。 Androidシステムは、プロバイダをしたにこのメソッドをびします。 ContentResolverオブジェクトがプロバイダにアクセスしようとするまで、プロバイダはされません。

```
@Override
public boolean onCreate() {
    dbhelper = new DatabaseHelper(getContext());
    return true;
}
```

getType コンテンツURIにするMIMEタイプをします。

```
@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 プロバイダからデ―タをします。をして、する、すと、およびのソ―トをします。デ―タをCursorオブジェクトとしてします。

```
@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;
}
```

プロバイダにしいをします。をしてテーブルをし、するをします。しくされたのコンテンツURIをします。

```
@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 プロバイダからをします。をして、するテーブルとをします。されたのをします。

```
@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 プロバイダののをします。をして、するテーブルとをし、しいのをします。されたのをします。

```
@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マニフェストファイルの

```
<provider
        android:authorities="com.example.myApp"
        android:name=".DatabaseProvider"/>
```

オンラインでContentProviderをむ https://riptutorial.com/ja/android/topic/3075/contentprovider

52: ContentValuesクラスをしたSQLiteデータベースへのアクセス

Examples

SQLiteデータベースののと

まず、のようにSQLiteデータベースをくがあります。

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

データベースをいた、 `ContentValues`クラスをしてにをまたはできます。のでは、のが`str_edtfname`で、のが`str_edtfname`と`str_edtlname`ます。 `table_name`は、するテーブルのできえるがあります。

データの

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

データの

```
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});
```

オンラインでContentValuesクラスをしたSQLiteデータベースへのアクセスをむ
https://riptutorial.com/ja/android/topic/10154/contentvaluesクラスをしたsqliteデータベースへのアクセス

53: DayNightテーマAppCompat v23.2 / API 14+

Examples

アプリにDayNightテーマをする

DayNightテーマは、とデバイスののののについて、カラースキームをりえるクールなをアプリにします。

`styles.xml`をしてください

```
<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>
```

のテーマりえをするためにできるテーマはのとおりです。

- `"Theme.AppCompat.DayNight"`
- `"Theme.AppCompat.DayNight.NoActionBar"`
- `"Theme.AppCompat.DayNight.DarkActionBar"`

`colorPrimary`、 `colorPrimaryDark`、 `colorAccent`に、りえたいのをすることもできます `textColorPrimary`や`textColorSecondary`。 アプリのカスタムをこの`style`することもできます。

テーマをにりえるには、デフォルトの`colors.xml`を`res/values`ディレクトリに、の`colors.xml`を `res/values-night`ディレクトリにし、/のをにするがあります。

テーマをりえるには、Javaコードから`AppCompatDelegate.setDefaultNightMode(int)`メソッドをびします。 これにより、1つのアクティビティやフラグメントだけでなく、アプリケーションのがされます。たとえば、のようになります。

```
AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
```

あなたはあなたのにっての3つのいずれかをすことができます

- `AppCompatDelegate.MODE_NIGHT_NO` これは、アプリケーションのデフォルトテーマをし、 `res/values`ディレクトリにされているをし`res/values`。 このテーマにはるいをすることをおめします。
- `AppCompatDelegate.MODE_NIGHT_YES` これはあなたのアプリののテーマをし、 `res/values-night` ディレクトリにされているをとります。このテーマにはいをすることをおめします。
- `AppCompatDelegate.MODE_NIGHT_AUTO` これは、と`values-night`ディレクトリの`values`と`values-`

`night`したについて、アプリケーションのをにりえます。

`getDefaultNightMode()`メソッドをして、のモードのステータスをすることもできます。えば

```
int modeType = AppCompatDelegate.getDefaultNightMode();
```

ただし、アプリをしてくと、テーマスイッチがされないことにしてください。これをうと、テーマはの`AppCompatDelegate.MODE_NIGHT_AUTO`にります。テーマスイッチをさせたいは、にをし、アプリがされたにアプリがかれるたびにされたをロードするようにしてください。

オンラインでDayNightテーマAppCompat v23.2 / API 14+をむ
https://riptutorial.com/ja/android/topic/7650/daynightテーマ-appcompat-v23-2---api-14plus-

54: Drawables

Examples

ない

drawableはのにすることができます。これは、アプリケーションのさまざまなテーマをサポートし、なリソースファイルのをらすのにです。

SDK 21+でのフレームワークAPIの

```
Drawable d = context.getDrawable(R.drawable.ic_launcher);
d.setTint(Color.WHITE);
```

SDK 4+でandroid.support.v4ライブラリをする

```
//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
```

それはないしてください`int color`、しかし、あなたは「」クラスでされたこれらのにされるものではなく、のリソースをされていません。あなたがしたいXMLでされたをっているときは、まずはそのをするがあります。

*`Color.WHITE`*をのできえることができます

いAPIをとする

```
getResources().getColor(R.color.your_color);
```

またはよりしいターゲット

```
ContextCompat.getColor(context, R.color.your_color);
```

みをびたコーナーでる

なフォルダに**custom_rectangle.xml**でなファイルをします。

```
<?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" />

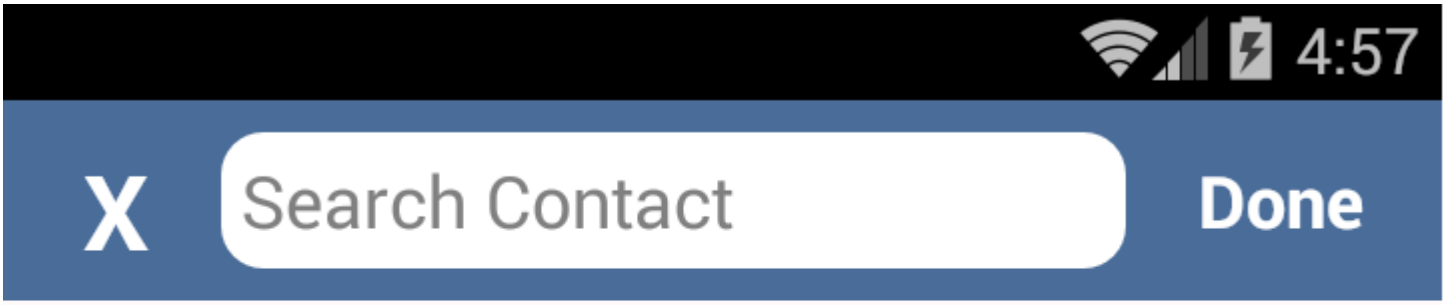
    <stroke
        android:width="1dp"
        android:color="@android:color/white" />

</shape>
```

すぐビュ―に のを

```
mView.setBackGround(R.drawlable.custom_rectangle);
```

リファレンススクリーンショット



のビュ―このは`TextView` の、 **drawble**フォルダにdrawble **round_view.xml**をします。

```
<?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>
```

DrawableをViewにりてます

```
    <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" />
```

はオレンジののようにえるはずです



カスタム

Drawableでクラスをし、これらのメソッドをオーバーライドする

```
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;
    }
}
```

レイアウトにImageViewをする

```
<ImageView
   android:layout_width="80dp"
   android:id="@+id/imageView"
   android:layout_height="80dp" />
```

カスタムDrawableをImageViewにする

```
 IconDrawable iconDrawable=new
IconDrawable(ContextCompat.getDrawable(this,android.R.drawable.ic_media_play),ContextCompat.getColor(th

 imageView.setImageDrawable(iconDrawable);
```

スクリーンショット



オンラインでDrawablesをむ https://riptutorial.com/ja/android/topic/4841/drawables

55: EditText

Examples

EditTextsの

EditTextは、Androidアプリのテキストウィジェットです。ユーザーがアプリにテキストをするがある、これがなです。

EditText

EditTextのをカスタマイズするためにできるくのなプロパティがあります。これらのうちのいくつかをにします。さらにしいフィールドのについては、のテキストフィールドガイドをしてください。

EditTextは、すべてのデフォルトをむレイアウトに、のXMLでされます。

```
<EditText
    android:id="@+id/et_simple"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
</EditText>
```

*EditText*はに*TextView*のいであり、じプロパティをすべてすることにしてください。

の

EditTextにされたテキストのをするはのとおりです。

```
EditText simpleEditText = (EditText) findViewById(R.id.et_simple);
String strValue = simpleEditText.getText().toString();
```

さらなるエントリのカスタマイズ

エントリを**1**のテキストにしたいかもしれませんをける

```
<EditText
  android:singleLine="true"
  android:lines="1"
/>
```

digitsをすると、フィールドにできるをすることができます。

```
<EditText
  android:inputType="number"
  android:digits="01"
/>
```

これにより、されたは「0」と「1」にされます。たちはのをのようにしたいかもしれません

```
<EditText
  android:maxLength="5"
/>
```

これらのプロパティをして、テキストフィ―ルドのされるをできます。

の

android:textColorHighlightプロパティをして、EditTextでしたテキストのハイライトのをできます

```
<EditText
    android:textColorHighlight="#7cff88"
/>
```

プレ―スホルダヒントの

EditTextコントロ―ルのヒントをして、ユ―ザ―にのをめるプロンプトをすることができます。

```
<EditText
    ...
    android:hint="@string/my_hint">
</EditText>
```

ヒント

ボトムライ ンのをする

AppCompatライブラリをしているとして、スタイルcolorControlNormal、colorControlActivated、およびcolorControlHighlightをオ―バ―ライドできます。

```
<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>
```

DialogFragmentにこれらのスタイルがされていないは、onCreateViewメソッドにされたLayoutInflaterをするとのバグがあります。

このはにAppCompat v23ライブラリでされています。アップグレ―ドについては、このガイドをしてください。のなは、onCreateViewメソッドにされたインフレ―タのわりにActivityのレイアウトインフレ―タをすることです。

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = getActivity().getLayoutInflater().inflate(R.layout.dialog_fragment,
container);
  }
```

EditTextのリッスン

EditTextへのをリッスンすると、がしたときにアクションをするについては、なイベントリスナーのcliffnotesをしてください。

ラベル・フィードバックの

に、EditTextは、ユーザがをしたでヒントメッセージをすでした。さらに、エラーメッセージはができるがありました。

`TextInputLayout`をすると、ヒントやエラーメッセージをするためにフローティングラベルをできます。 はこちらをごください 。

InputTypeのカスタマイズ

テキストフィールドは、、、パスワード、メールアドレスなどのなるタイプをつことができます。タイプは、フィールドでどのようなのがされるかをし、キーボードがにされるのレイアウトをするようにします。

デフォルトでは、 `EditText`コントロールのテキストコンテンツはプレーンテキストとしてされます。 `inputType`をすることで、やパスワードなどのさまざまなののをにすることができます。

```
<EditText
    ...
    android:inputType="phone">
</EditText>
```

もなタイプはのとおりです。

タイプ	
textUri	URIとしてされるテキスト
textEmailAddress	メールアドレスとしてされるテキスト
textPersonName	のであるテキスト
textPassword	されるべきパスワードであるテキスト
	のみのフィールド
	をする
	をする
	をする
textMultiLine	フィールドにのテキストをする

`android:inputType`では、しいをすべてにするか、やスペルなどのをするかなど、のキーボードをすることもできます。
キーボードのをするなタイプのをにします。

タイプ	
textCapSentences	しいごとにのをにするのテキストキーボード
textCapWords	すべてのをにするのテキストキーボード。タイトルやにしています
textAutoCorrect	よくりのないをするのテキストキーボード

にじての`inputType`をすることができます '|'でります。

```
<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" />
```

なすべてのタイプのリストがここにされます 。

`inputype`

`inputype`の`EditText`ウィジェットアンドロイド4.4.3および2.3.3でテスト

```
<EditText android:id="@+id/et_test" android:inputType="?????"/>
```

textLongMessage =キーボードアルファベット/デフォルト。 Enterボタン/へはい。ケース。はい。します。、と。すべて

textFilter =キーボードアルファベット/デフォルト。 Enterボタン/へはい。ケース。 いいえ 。します。、と。すべて

textCapWords =キーボードアルファベット/デフォルト。 Enterボタン/へはい。 ケースキャメルケース 。はい。します。、と。すべて

textCapSentences =キーボードアルファベット/デフォルト。 Enterボタン/へはい。 ケース 。はい。します。、と。すべて

=キーボード。 Enterボタン/へいいえ。ケース - 。 いいえ 。します。 chars

textMultiLine =キーボードアルファベット/デフォルト。 **Enter**ボタンの 。はい。ケース。はい。します。、と。すべて

number = キーボード。 Enterボタン/へいいえ。ケース - 。いいえ。 します。も

textEmailAddress =キーボードアルファベット/デフォルト。 Enterボタン/へいいえ 。ケース。いいえ 。します。 chars @と。すべて

タイプなし**=**キーボードアルファベット/デフォルト。 **Enter**ボタンの 。はい。ケース。はい。します。、と。すべて

textPassword =キーボードアルファベット/デフォルト。 Enterボタン/へいいえ。ケース。 いいえ 。します。、と。すべて

text =キーボードキーボードアルファベット/デフォルト。 Enterボタン/へはい。ケース。はい。します。、と。すべて

textShortMessage =キーボードアルファベット/デフォルト。 **Enter**ボタン 。はい。ケース。はい。します。、と。すべて

textUri =キーボードアルファベット/デフォルト。 Enterボタン/へいいえ。ケース。 いいえ 。します。 chars **/**と。すべて

textCapCharacters =キーボードアルファベット/デフォルト。 Enterボタン/へはい。 はい。します。、と。すべて

phone = キーボード。 Enterボタン/へいいえ。ケース - 。 いいえ 。します。***。 - /+WPN、+**

textPersonName =キーボードアルファベット/デフォルト。 Enterボタン/へはい。ケース。はい。します。、と。すべて

`Auto-capitalization`は、デフォルトのをします。

2 `Numeric keyboard` 、すべてのは1234567890です。

3 `Correction/Suggestion`は、デフォルトをします。

SoftKeyboardをす

Softkeyboardをにすることは、EditTextをするのなです。ソフトキーボードはデフォルトでるボタンをすだけでじることができるので、ほとんどのはInputMethodManagerをしてhideSoftInputFromWindowをびすキーボードをにして、ビューをむウィンドウのトークンをします。をうコード

```
public void hideSoftKeyboard()
{
        InputMethodManager inputMethodManager = (InputMethodManager)
getSystemService(Activity.INPUT_METHOD_SERVICE);
        inputMethodManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(), 0);
```

```
}
```

コードはですが、するのきなは、イベントがしたときにhideをびすがあることです。 EditTextのをするとSoftkeyboardがになるがあるはどうすればいいですかのコードは、あなたのonCreateメソッドでだけびされるがあるきれいなをしています。

```
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);
            }
        }
}
```

カスタムテキストとそのアクションのアイコンまたはボタンをクリックし、リスナーをクリックします。

このは、にアイコンでテキストをするのにちます。

> これではsetCompoundDrawablesWithIntrinsicBoundsをしています。アイコンのをしたいは、setIconでsetCompoundDrawablesWithIntrinsicBoundsをしてアイコンのをできます。

```
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);
    }
}
```

タッチをしたいは、EXTRA_TOUCH_AREAのデフォルトを50にすることができます。

ボタンとクリックリスナをにするために、あなたのアクティビティやフラグメントからこのようにびすことができます。

```
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.

            }
        });
```

オンラインでEditTextをむ https://riptutorial.com/ja/android/topic/5843/edittext

56: ExoPlayer

Examples

ExoPlayerをプロジェクトにする

Via jCenter

プロジェクトのbuild.gradleファイルにのようにします。

```
compile 'com.google.android.exoplayer:exoplayer:rX.X.X'
```

ここで、rX.XXはおみのバージョンです。バージョンについては、プロジェクトのリリースをしてください。については、 Bintrayのプロジェクトをしてください 。

ExoPlayerの

ExoPlayerをインスタンスする

```
exoPlayer = ExoPlayer.Factory.newInstance(RENDERER_COUNT, minBufferMs, minRebufferMs);
```

オーディオのみをするには、のをします。

```
RENDERER_COUNT = 1 //since you want to render simple audio
minBufferMs = 1000
minRebufferMs = 5000
```

のバッファは、にじてできます。

これでデータソースをするがあります。 mp3をストリーミングするは、DefaultUriDataSourceをできます。コンテキストとUserAgentをすがあります。なままローカルファイルをし、nullをuserAgentとしてす

```
DataSource dataSource = new DefaultUriDataSource(context, null);
```

に、sampleSourceをします。

```
ExtractorSampleSource sampleSource = new ExtractorSampleSource(
                uri, dataSource, new Mp3Extractor(), RENDERER_COUNT, requestedBufferSize);
```

uriがファイルをしている、mp3をしたいは、Extractorとしてなデフォルトの Mp3Extractorをすることができます。 requestedBufferSizeは、にじてすることができます。たとえば、5000をします。

これで、のようにサンプルソースをしてオーディオトラックレンダラーをできます。

```
MediaCodecAudioTrackRenderer audioRenderer = new MediaCodecAudioTrackRenderer(sampleSource);
```

に、exoPlayerインスタンスでprepareをびします。

```
exoPlayer.prepare(audioRenderer);
```

をするには

```
exoPlayer.setPlayWhenReady(true);
```

のTrackRendererをしてビデオとオーディオをするな

```
// 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!
```

オンラインでExoPlayerをむ https://riptutorial.com/ja/android/topic/6248/exoplayer

57: FileProvider

Examples

ファイルの

このでは、のアプリとファイルをするをします。このではpdfファイルをしますが、このコードはのすべてのでもします。

ロードマップ

したいファイルがかれているディレクトリをする

ファイルをするために、アプリケーションのなファイルをにするFileProviderクラスをします。FileProviderはあらかじめされたディレクトリのファイルのみをできるので、これらをしましょう。

1. パスをむしいXMLファイルをします。えば、 *res / xml / filepaths.xml*
2. パスをする

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <files-path name="pdf_folder" path="documents/"/>
</paths>
```

FileProviderをし、ファイルパスにリンクする

これはマニフェストでわれます。

```
<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>
```

ファイルのURIをする

ファイルをするには、ファイルのをするがあります。これは、URIUniform Resource Identifierをしてわれます。

```
// 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);
```

コードでわかるように、まずファイルをすしいFileクラスをします。 URIをするには、FileProviderにいわせてください。 2のはですそれはFileProviderのをします。これは、マニフェストでされているFileProviderのとしくなければなりません。

のアプリとファイルをする

ShareCompatをってのアプリとファイルをします

```
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);
```

は、ユーザーがファイルをしたいアプリケーションをできるメニューです。フラグIntent.FLAG_GRANT_READ_URI_PERMISSIONは、URIへのなみりアクセスをえるためにです。

オンラインでFileProviderをむ https://riptutorial.com/ja/android/topic/6266/fileprovider

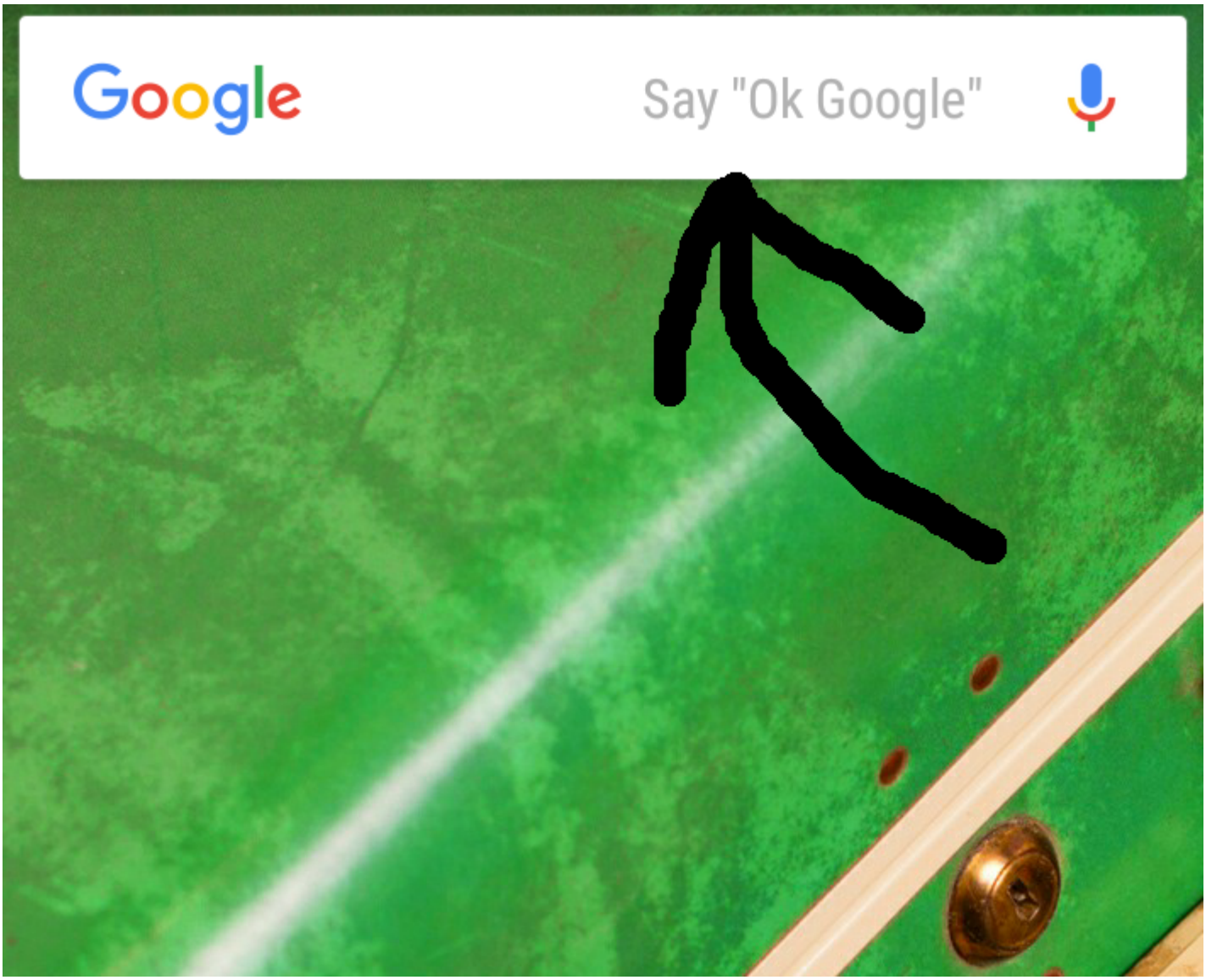
58: Firebase App Indexing

- App Indexingをすることをすると、させるかもしれないのブログ、ドキュメントがあります。この、Firebase-Googleがするドキュメントにすることをおめします。これをうにはをしたいとっても、まずこのにってください。これは、どのようにしているかをにしてくれるからです。

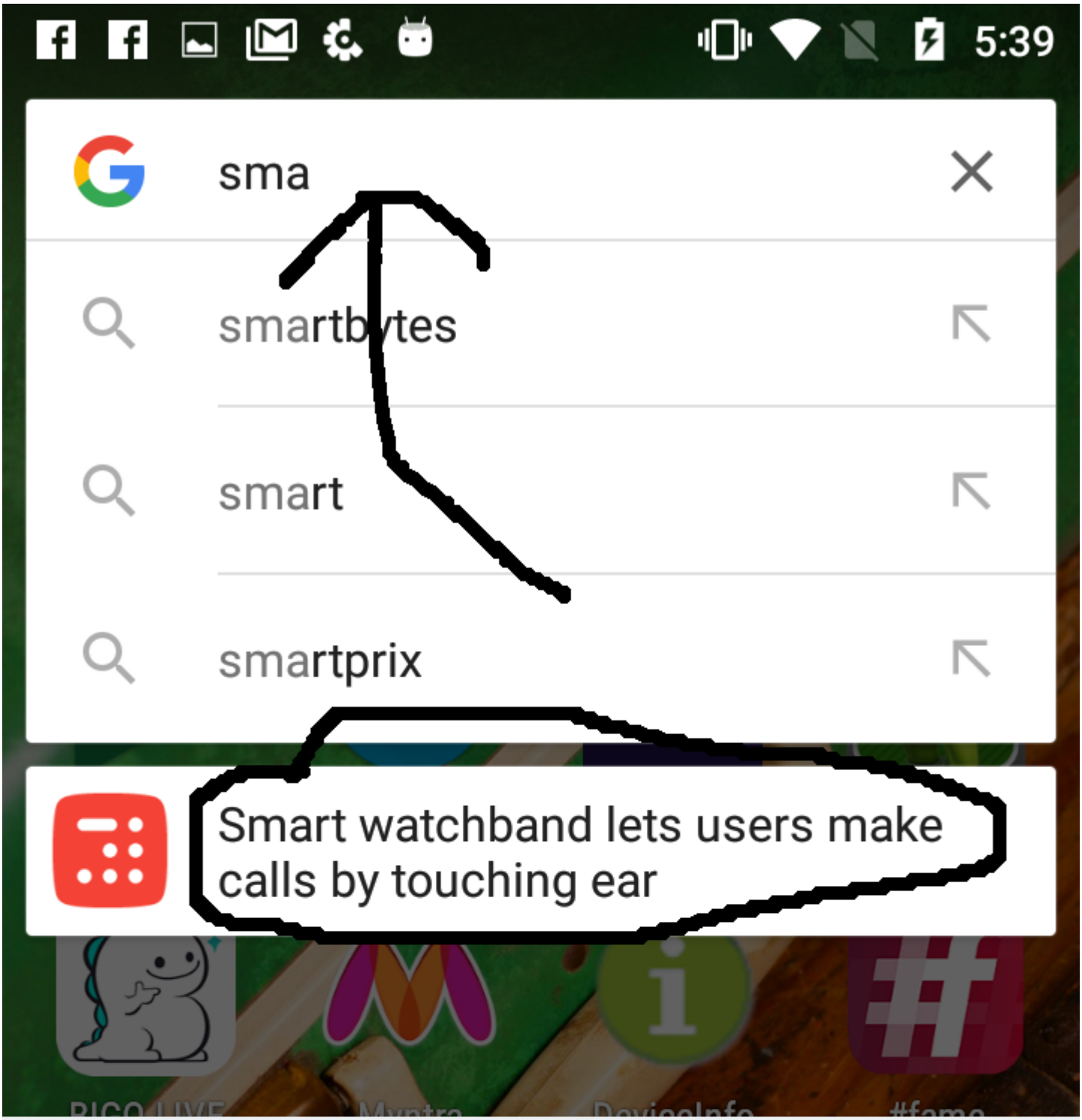
- Googleではコンテンツのインデックスに24をします。だから、をつけてください。あなたはすべてのことがあなたのでうまくいくようにテストすることができます。

- のでは、アプリケーションでリダイレクトするWebサイトのHTTP URLをサポートできます。これは、あなたがGoogleでクエリをした、にインストールされているあなたのアプリリンクにあなたのウェブサイトのURLの1つがされます。このURLをクリックすると、そのにするApp Screenにリダイレクトされます。それはがこれをしたことです。

- AppIndexing APIをすると、コンテンツがインデックスにされ、Googleバーのでされます。inShortsのをることができますページのしとさながあります。 2つまたは3つのしをみえたら、アプリケーションをじてgoogle searchbarにします。



ちょうどあなたがしたしをしようとすると、タイトルとしてそのしとのAppページのをします。これは、Appsのにされるappのとはなります。これは、こののページのAppIndexing APIコードをしていて、titleが`onCreate()`したものとじであるためにします。


5:39
sma
smartbytes
smart
smartprix
Smart watchband lets users make calls by touching ear

をする。このファイルをすることにより、コンテンツのGoogleクロールをできます。については、 このリンクをしてください。

ステップ2 - アプリケーションをあなたのウェブサイトにけます。assetlinks.jsonをめるWebサーバーの.wellのディレクトリにアップロードします。あなたのassetlinks.jsonのコンテンツは、

```
       [{
"relation": ["delegate_permission/common.handle_all_urls"],
"target" :
{ "namespace": "android_app",
"package_name": "<your_package_name>",
"sha256_cert_fingerprints": ["<hash_of_app_certificate>"] }
```

}]

ステップ3 - あなたのマニフェストファイルにアプリケーションリンクをめて、のようにURLをアプリケーションにリダイレクトし、

```
<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>
```

ここでタグについてびたいは、これをしてください。

<action> Google SearchからインテントフィルターにできるようにACTION_VIEWインテントアクションをします。

<data> 1つのタグをします。タグは、アクティビティをするURIをします。なくとも、タグにはandroidschemeがまれているがあります。をして、アクティビティーがけれるURIのタイプをさらにりむことができます。たとえば、したURIをけれるのアクティビティをつこともできますが、にパスにづいてなります。この、androidpathまたはそのバリエーションpathPatternまたはpathPrefixをして、システムがなるURIパスにしてくがあるアクティビティをします。

<category> BROWSABLEカテゴリをめます。 Webブラウザからインテントフィルタにアクセスするには、BROWSABLEカテゴリがです。それがなければ、ブラウザのリンクをクリックしてもアプリにすることはできません。 DEFAULTカテゴリはオプションですが、されています。この

カテゴリがないと、アプリコンポ―ネントをしてなインテントでアクティビティをすることはできません。

ステップ4 - URLをする

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

}

ステップ5 - Android Debug Bridgeコマンドまたはスタジオをしてこれをテストできます。 Adbコマンド - アプリケ―ションをし、のコマンドをします。

```
adb shell am start -a android.intent.action.VIEW -d "{URL}" < package name >
```

Androidスタジオの - **Android**スタジオ**>**ビルド**>**の**>**オプション**> URL**を**> URL**をここに**>**アプリをしてテストします。「」ウィンドウにエラ―がされたは、それのはにされ、されているはあなたのURLにされたペ―ジにリダイレクトされます。

AppIndexing APIをする

これをプロジェクトにすると、にのをつけることができますが、このでは、をするべきなのいくつかをします。

ステップ1 - Googleサ―ビスをする

```
dependencies {
      ...
      compile 'com.google.android.gms:play-services-appindexing:9.4.0'
      ...
    }
```

ステップ2 - クラスをインポ―トする

```
import com.google.android.gms.appindexing.Action;
import com.google.android.gms.appindexing.AppIndex;
import com.google.android.gms.common.api.GoogleApiClient;
```

ステップ3 - App Indexing APIびしをする

```
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();
}
```

これをテストするには、のの4にってください。

オンラインでFirebase App Indexingをむ https://riptutorial.com/ja/android/topic/5957/firebase-app-indexing

59: Firebase Cloud Messaging

き

Firebase Cloud MessagingFCMは、クロスプラットフォームのメッセージングソリューションであり、にメッセージをにできます。

FCMをすると、しいメールやそののデータができることをクライアントアプリケーションにできます。メッセージをして、ユーザーのとをすことができます。インスタントメッセージングなどのユースケースの、メッセージは4KBペイロードをクライアントアプリケーションにできます。

Examples

AndroidでFirebase Cloud Messaging Clientアプリケーションをする

1. インストールとのをして、あなたのアプリをFirebaseにします。
   これでFirebaseにプロジェクトがされます。

2. Firebase Cloud Messagingのをモジュールレベルのbuild.gradleファイルにします

```
dependencies {
     compile 'com.google.firebase:firebase-messaging:10.2.1'
}
```

これで、AndroidでFCMをするがいました。

FCMクライアントでは、 Android 2.3をし、Google Playストアアプリをインストールした、Android 2.3のGoogle APIをしたがです。

AndroidManifest.xmlファイルをする

```
<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>
```

トークン

FCM SDKは、アプリケーションのに、クライアントアプリケーションインスタンスのトークンをします。
のデバイスをターゲットにする、またはデバイスグループをするは、 `FirebaseInstanceIdService`をしてこのトークンにアクセスするがあります。

しいトークンがされるたびに`onTokenRefresh`コールバックが`onTokenRefresh`、`FirebaseInstanceID.getToken()`メソッドをしてのトークンをできます。

```
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);

    }

}
```

はあなたの**webView**でくための、メッセージ、およびリンクをプッシュするためにのアプリでしているこのコード

これはのFirebaseMessagingServiceです

```
    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;

    }
}}
```

これはMainActivityでWebViewやのブラウザのリンクをき、にじてインテントでリンクをきます。

```
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();
    }}
```

メッセ―ジをける

メッセ―ジをするには、 `FirebaseMessagingService`をして`onMessageReceived`メソッドをオ―バ―ライドするサ―ビスをします。

```
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());
        }

        //.....
    }
```

アプリがバックグラウンドにあるとき、Androidはシステムトレイにメッセージをします。ユーザーがをタップすると、デフォルトでアプリランチャーがきます。

これには、とデータペイロードおよびコンソールからされたすべてのメッセージのをむメッセージがまれます。このような、はデバイスのシステムトレイにされ、データペイロードはランチャーアクティビティののにされます。

ここでな

アプリ	おらせ	データ	
	`onMessageReceived`	`onMessageReceived`	`onMessageReceived`
バックグラウンド	システムトレイ	`onMessageReceived`	システムトレイ
			データのなもの。

トピックをする

クライアントアプリケーションはのトピックをすることも、しいトピックをすることもできます。クライアントアプリケーションがしいトピックにすると、そののしいトピックがFCMでされ、その、どのクライアントもそのトピックにサブスクライブできます。

トピックをするには、トピックをして`subscribeToTopic()`メソッドをし`subscribeToTopic()`。

```
FirebaseMessaging.getInstance().subscribeToTopic("myTopic");
```

オンラインでFirebase Cloud Messagingをむ https://riptutorial.com/ja/android/topic/8826/firebase-

cloud-messaging

60: Firebaseクラッシュレポート

Examples

Firebaseクラッシュレポートをアプリにする

*Firebase*クラッシュレポートをアプリにするには、のをします。

- ここで *Firebase*コンソールにアプリケーションをします。
- `google-services.json`ファイルをプロジェクトのin `app/`ディレクトリにコピーします。
- `google-services`プラグインをするには、ルートレベルの*build.gradle*ファイルにのルールをします。

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.0.0'
    }
}
```

- モジュールのGradleファイルで、Gradleプラグインをにするには、ファイルのに`apply plugin`をし`apply plugin`。

```
apply plugin: 'com.google.gms.google-services'
```

- クラッシュレポートのをアプリレベルの*build.gradle*ファイルにします。

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

- のをして、アプリケーションからカスタムをさせることができます。

```
FirebaseCrash.report(new Exception("Non Fatal Error logging"));
```

すべてのなは、 *Firebase*コンソールにされます。

- カスタムログをコンソールにするは、のコードをできます。

```
FirebaseCrash.log("Level 2 completed.");
```

については、をごください。

-
- スタックオーバーフロートピック

エラーをする

Firebaseクラッシュレポートは、なエラーまたはキャッチされていないのレポートをにします。

をしてカスタムレポートをできます。

```
FirebaseCrash.report(new Exception("My first Android non-fatal error"));
```

FirebaseCrashがモジュールをしたときに、ログをチェックインできます

> 07-20 085724.442 D / FirebaseCrashApiImpl **FirebaseCrash**レポート**API**の 07-20 08 5724.442 I / FirebaseCrash **FirebaseCrash**レポートの d com.google.firebase.crash.internal.zzg@3333d325 07-20 08 5724.442 D / FirebaseApp されたクラス**com.google.firebase.crash.FirebaseCrash**。

そして、それがをしたとき

> 07-20 085747.052 D / FirebaseCrashApiImplthrowable **java.lang.Exception**のの **Android**でではないエラー 07-20 085818.822 D / FirebaseCrashSenderServiceImpl レスポンスコード**200** 07-20 085818.822 D / FirebaseCrashSenderServiceImpl レポートがされました

カスタムログをしてレポートにすることができます

```
FirebaseCrash.log("Activity created");
```

オンラインでFirebaseクラッシュレポートをむ
https://riptutorial.com/ja/android/topic/5965/firebaseクラッシュレポート

61: Firebaseリアルタイムデータベース

そののトピック

- ファイアベース

Examples

Firebase Realtime DataBaseイベントハンドラ

にFirebaseDatabaseをする

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

あなたのデータベースにきみます

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

あなたのデータベースからむ

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

Androidイベントのデータをする

```
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);
```

クイックセットアップ

1. インストールとのをして、あなたのアプリをFirebaseにします。
   これでFirebaseにプロジェクトがされます。

2. Firebase Realtime Databaseのをモジュールレベルの`build.gradle`ファイルにします

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

3. Firebaseデータベースルールの

これで、Androidのリアルタイムデータベースでするがいました。

たとえば、 `Hello World`メッセージを`message`キーののデータベースにきみます。

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Firebaseデータベースからリアルタイムデータをするのと

このでは、Firebase Realtime Databaseをすでにしていることをとしています。のは、Androidプロジェクトに Firebaseをするについて、 ここにごください。

まず、Firebaseデータベースのをappレベルの*build.gradle*ファイルにします。

```
compile 'com.google.firebase:firebase-database:9.4.0'
```

さて、Firebaseデータベースにデータをするチャットアプリをってみましょう。

ステップ1チャットというのクラスをする

チャットにないくつかのをつクラスをするだけです

```
public class Chat{
    public String name, message;
}
```

ステップ2いくつかのJSONデータをする

Firebaseデータベースとのでデータのをうには、JSONをするがあります。いくつかのチャットがすでにデータベースのルートレベルにされているとします。これらのチャットのデータはのようになります。

```
[
    {
        "name":"John Doe",
        "message":"My first Message"
    },
    {
        "name":"John Doe",
        "message":"Second Message"
    },
    {
        "name":"John Doe",
        "message":"Third Message"
    }
]
```

ステップ3リスナーをする

リスナーには3つのタイプがあります。のでは、 `childEventListener`をし`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();
    }
});
```

ステップ4データベースにデータをする

チャットクラスオブジェクトをし、のようにをするだけです

```
Chat chat=new Chat();
chat.name="John Doe";
chat.message="First message from android";
```

セッションでわれたように、チャットノードへのをします。

```
DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference().child("chats");
```

データのをするに、チャットノードにはのノードがされているため、しいチャットをするとチャットのをむしいノードがされるため、さらにいがです。 `DatabaseReference`オブジェクトの`push()`をしてノードのしいのをできます。これは、の`DatabaseReference`をします。これは、しくされたノードをチャットデータをすることをします。

```
// The parameter is the chat object that was newly created a few lines above.
chatDb.push().setValue(chat);
```

`setValue()`は、アプリケーションのすべての`onDataChanged`がじデバイスをめてびされていることをします。これは、「チャット」ノードのリスナです。

フラットなデータベース

されたコールをにダウンロードするには、とフラットなデータベースがです。のでは、をすることもです。こののは、にのでデータをするがあることです。

たとえば、ユーザーがメッセージをメモにできるようにするアプリをえてみましょう。

ましいフラットデータベース

```
|--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"
```

されたメモクラス

```
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;
    }
}
```

ユーザーのメモをする

```
//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) { }
    }
```

メモをする

```
//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);
```

プッシュまたはデータベースがのようになったら

```
|--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"
```

firebase JSONデータベースについて

コードでをすに、データがfirebaseにどのようにされているかをするがあるとじています。リレーショナルデータベースとはなり、firebaseはJSONでデータをします。リレーショナルデータベースのはJSONオブジェクトにけられていないキーとのペアです。したがって、はキーになり、あるののにされているがになります。このようにして、がJSONオブジェクトとしてされ、これらのリストはデータベーステーブルをします。がこれをるとすぐにられるは、スキーマのがいRDBMSにべてはるかになになることです。テーブルをするよりも、JSONに2つのをするがです。

firebaseにデータがどのようにされるかをすJSONのサンプルをにします。

```
{
 "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
}
```

これは、リレーショナルデータベースにしたデータをJSONでするをにしています。に、アンドロイドデバイスでこのデータをむをてみましょう。

firebaseからのデータの

はあなたがすでにアンドロイドスタジオにgradlefirebaseをすることをっているとします。あなたがここからのガイドにっていないならば。をした、firebase console、gradle sync android studioであなたのアプリをしてください。すべてのは、firebaseデータベースとfirebase authだけではありません。

データがどのようにされ、どのようにgradleをするかをったので、インポートされたfirebase android SDKをってデータをするをてみましょう。

firebaseデータベースリファレンスをする

```
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)
```

ここから、のあるデータをすようにのchildメソッドびしをさせることができます。たとえば、のセクションでしたようにデータがされていて、Bruce Wayneユーザーをしすは、

```
DatabaseReference bruceWayneRef = userDBRef.child("371298");
// 371298 is key of bruce wayne user in JSON structure (previous section)
```

または、JSONオブジェクトへのをすだけです

```
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..."
```

フェッチするデータのができたので、リスナーをしてアンドロイドアプリでデータをできます。レトロフィットまたはボレーをしてREST APIコールをするのコールとはなり、ここではなコールバックリスナーがデータをするがあります。 Firebase sdkがコールバックメソッドをびすと、です。

に2つのタイプのリスナーがあります.1つはValueEventListenerで、もう1つはChildEventListenerです のセクションでします。のイベントリスナーは、JSONをし、イベントリスナーは、がしたのをします。これらのは、それぞれのでちます。 firebaseからデータをするには、firebaseデータベースほどしたuserDBRefのリストに1つまたはのリスナーをできます。

いくつかのサンプルコードコードののコードのはのとおりです。

```
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
    });
```

されたクラスのにづいたか DataSnapshotは、JSONデータをされたPOJOにできます。なしいクラスタイプをします。

しのがするたびにデータこのはuser_baseテーブルをとしないや、データをだけしたいは、データベースの***addListenerForSingleValueEvent***メソッドをできます。コールバックは1だけします。

```
userDBRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // Do something
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Do something about the error
    });
```

のサンプルは、JSONノーードのをします。キーをするには

```
String myKey = dataSnapshot.getKey();
```

のをく

チャットアプリやコラボレーティブなリストアプリには、ユーザーでオブジェクトのリストをさせるがありますのようなユースケースをしましょう。 firebaseデータベースをして、チャットノードまたはリストのノードにイベントリスナーをすると、チャットノードがされるたびにチャットのをするからチャットがしますつまりもがこんにちは。たちがんでいないこと、たちがをっているのは、しいノードだけ、あるいはまたはされたいノードだけです。されていないノードはされません。

この、 ChildEvenListenerをできます。それのコードサンプルはありませんサンプルJSONデータののセクションをしてください

```
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) {
    });
```

メソッドはです。しいユーザーがされたり、のユーザーののプロパティがされたり、ユーザーがまたはされたりすると、するデータとともにイベントリスナーのなコールバックメソッドがびされます。ですから、チャットアプリケーションのためにUIをリフレッシュしておくは、

onChildAddedからJSONをしてPOJOにパースし、UIにフィットさせてください。ユーザーがをれるときにリスナーをすることをれないでください。

onChildChangedは、されたプロパティしいものでをします。

onChildRemovedは、されたノードをします。

ページネーションによるデータの

なJSONデータベースをしている、イベントリスナーをすることはがありません。それはなJSONをし、それをすることはがかかるでしょう。このような、ページをしてデータのをりしてまたはすることができます。ユーザーがクリックしていチャットをすると、いチャットをフェッチするような、なローディングのようなものです。この、 Queryができます。

これまでのセクションのいをてみましょう。ユーザーベースには3のユーザーがまれています。ユーザーが30にえ、50のバッチでユーザーリストをするは、

```
// 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
    });
```

ここで、イベントまたはイベントをしてくことができます。もうクエリーをびしての50 をします。orderByChild メソッドをずしてください 。これは、これがなければしません。 Firebaseは、ページングするをるがあります。

オンラインでFirebaseリアルタイムデータベースをむ
https://riptutorial.com/ja/android/topic/5511/firebaseリアルタイムデータベース

62: FloatingActionButton

き

フローティングアクションボタンはなタイプのプロモートアクションにされ、デフォルトではするとしてにアニメーションされます。そののアイコンはアニメーションされてもよく、FABはなのためにのUIとははなるきをするかもしれない。フローティングアクションボタンは、アプリケーションのなアクションをし、アクションをトリガーするか、どこかをナビゲートするだけです。

パラメーター

パラメータ	
`android.support.design:elevation`	FABの。 "@ [+] [package] type / name"というのリソースや、 "[package] type / name"というのテーマへのであるがあります。
`android.support.design:fabSize`	FABのサイズ。
`android.support.design:rippleColor`	FABのリップルカラー。
`android.support.design:useCompatPadding`	compatめみをにします。

フローティングアクションボタンは、なタイプのプロモートアクションにされます。それらは、UIのにかんだのアイコンでされ、モーフィング、、およびアンカーポイントのにするなをえています。

もなアクションをすために、ごとに1つのフローティングアクションボタンのみをします。

`FloatingActionButton`をするに、 `build.gradle`ファイルにデザインサポートライブラリのをするがあります。

```
dependencies {
    compile 'com.android.support:design:25.1.0'
}
```

https://developer.android.com/reference/android/support/design/widget/FloatingActionButton.html

https://material.google.com/components/buttons-floating-action-button.html

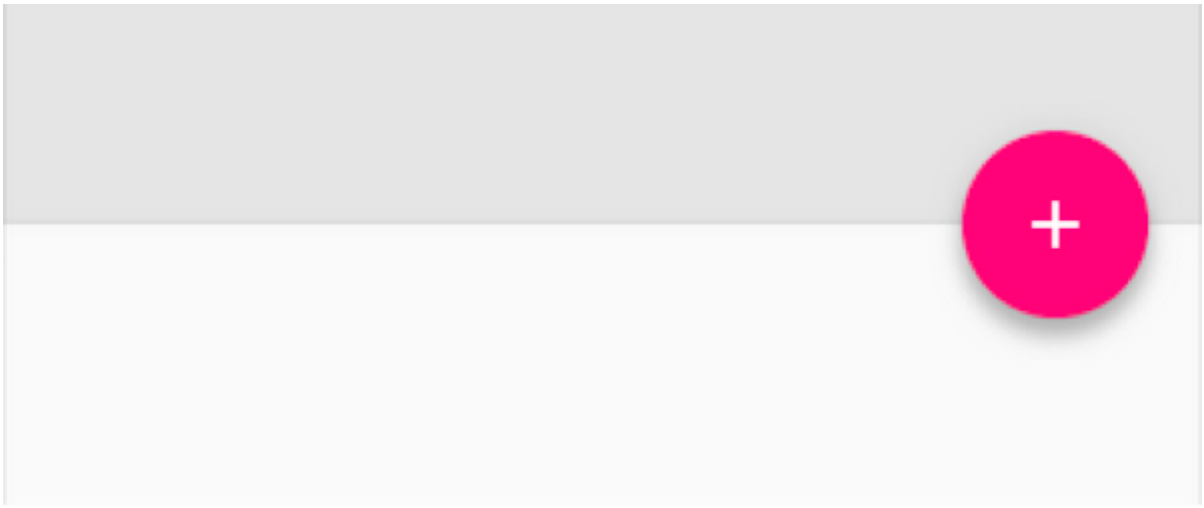
Examples

レイアウトにFABをする

FloatingActionButtonをするには、「`build.gradle`にって`build.gradle`ファイルにをするだけです。

に、レイアウトにします。

```
<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" />
```



このビューのは、デフォルトでテーマのcolorAccentになります。

のイメージでは、 `src`が+ iconデフォルトでは24x24 dpをしている、なのをするには、
`app:backgroundTint="@color/your_colour"`

できるコードのをしたいは、

```
myFab.setBackgroundTintList(ColorStateList.valueOf(your color in int));
```

されたでFABのをしたいは

```
mFab.setRippleColor(your color in int);
```

ポジショニング

では16dp、タブレット/デスクトップでは24dpをすることをおすすめします。

`FloatingActionButton`のをカバーするにsrcをしたら、のをるには、そのイメージのサイズがしいことをしてください。

デフォルトサークルサイズは56 x 56dpです



ミニサークルサイズ40 x 40 dp

インテリアアイコンのみをするは、デフォルトのサイズとして24 x 24 dpのアイコンをします

スワイプでの**FloatingActionButton**のと

`FloatingActionButton`をデフォルトのアニメーションでおよびにするには、 `show()`および`hide()`メソッドをびします。 `FloatingActionButton`をFragmentにくのではなく、Activityレイアウトにすることをおめします。これにより、とのときにデフォルトアニメーションがします。

`ViewPager`たをにし`ViewPager`。

- 3つのタブ
- と3のタブの`FloatingActionButton`を
- のタブで`FloatingActionButton`をにする

```
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();
                }
            }

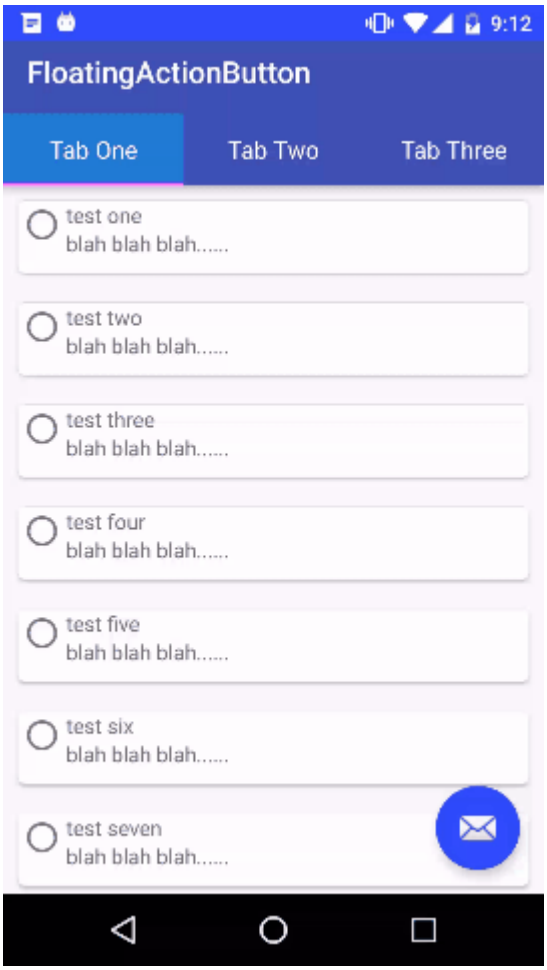
            @Override
            public void onPageScrolled(int position, float positionOffset, int
positionOffsetPixels) {}

            @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();
                }
            }
        });

    }
}
```



スクロールでのFloatingActionButtonのと

サポートライブラリバージョン22.2.1から、 `show()` `hide()`メソッドと`hide()`メソッドをする`FloatingActionButton.Behavior`サブクラスをして、スクロールからFloatingActionButtonをおよびにすることができます。

これは、 RecyclerViewやNestedScrollViewなど、ネストされたスクロールをサポートするビューとともにCoordinatorLayoutでのみすることにしてください。

この`ScrollAwareFABBehavior`クラスは、 `ScrollAwareFABBehavior`のAndroidガイド がなcc-wikiにあります。

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

FloatingActionButtonレイアウトxmlで、クラスが`ScrollAwareFABBehavior`の`app:layout_behavior`をします。

```
app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
```

たとえば、このレイアウトではのようになります。

```
<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"
            />

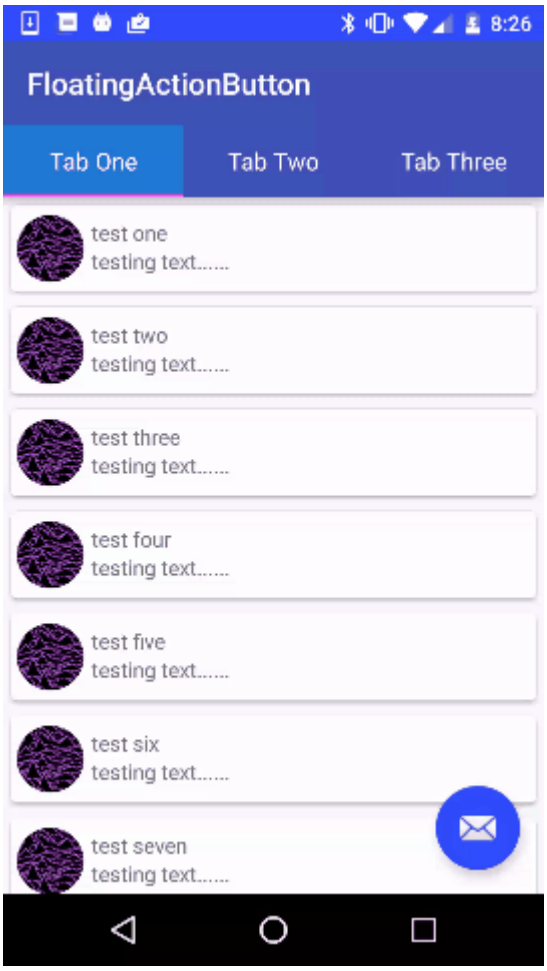
    </android.support.design.widget.AppBarLayout>

    <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>
```

はのとおりです。



FloatingActionButtonのの

XMLでFABのをできます。

えば

```
<android.support.design.widget.FloatingActionButton
   app:layout_behavior=".MyBehavior" />
```

または、をしてプログラムでできます。

```
CoordinatorLayout.LayoutParams p = (CoordinatorLayout.LayoutParams) fab.getLayoutParams();
p.setBehavior(xxxx);
fab.setLayoutParams(p);
```

オンラインでFloatingActionButtonをむ
https://riptutorial.com/ja/android/topic/2979/floatingactionbutton

63: Google Awareness APIs

Snapshot APIはのをするのにされ、 Fence APIはされたをにチェックし、アプリケーションがされていないときにコールバックをすることをれないでください。

として、Snapshot APIまたはFence APIをするためのながいくつかあります。

- Google Developers ConsoleからAPIキーをする
- マニフェストになとAPIキーをする

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

- できればアクティビティのonCreateメソッドで`GoogleApiClient`どこかで`GoogleApiClient`してください。

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

- あなたのしたAPIをびす
-

なユーザーをするなは、のようなです。

```
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

Snapshot APIをしてのユーザーアクティビティをする

ユーザーのにする1りのなについては、Snapshot APIをしてください。

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

Snapshot APIでヘッドホンのをする

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

Snapshot APIをしてのをする

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

Snapshot APIをしてのをする

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

```
    });
```

それらのでデ―タをする、いくつかのオプションがあります

```
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()));
```

Snapshot APIをしてのをする

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

Fence APIをしてユ―ザ―アクティビティのをする

ユ―ザ―が、、または`DetectedActivityFence`クラスののアクティビティなどのアクティビティをまたはしたときをするは、 `DetectedActivityFence`するアクティビティのフェンスをし、ユ―ザ―の/このアクティビティをします。 `BroadcastReceiver`をして、アクティビティをむデ―タをむ`Intent`をします。

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

これで、ユーザがアクティビティをしたときにコールバックをするために、 `BroadcastReceiver`インテントをけることができます。

```
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;
        }
    }
}
```

Fence APIをしてののをする

ユーザーがのをしたときにするは、ののフェンスをして、ユーザーがそのにりしたときにをけることができます。

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

に、ユーザーのをけるためのBroadcastReciverをします。

```
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;
        }
    }
}
```

オンラインでGoogle Awareness APIsをむ https://riptutorial.com/ja/android/topic/3361/google-awareness-apis

64: Googleドライブ API

き

Googleドライブは、Googleがしたファイルホスティングサービスです 。これは、ファイルストレージサービスをし、ユーザーがクラウドのファイルをアップロードし、のとすることができます。 GoogleドライブAPIをすると、パソコンやとGoogle Drive Cloudのでファイルをできます。

アプリケーションでGoogleドライブAndroid APIをするは、アプリケーションの「」セクションのとしてGoogle Playサービスのテキストをめるがあります。

をしたメニューとして、または「」メニューのとしてめることをおめします。

にテキストをするには、 `GooglePlayServicesUtil.getOpenSourceSoftwareLicenseInfo()` をびすことができます。

Examples

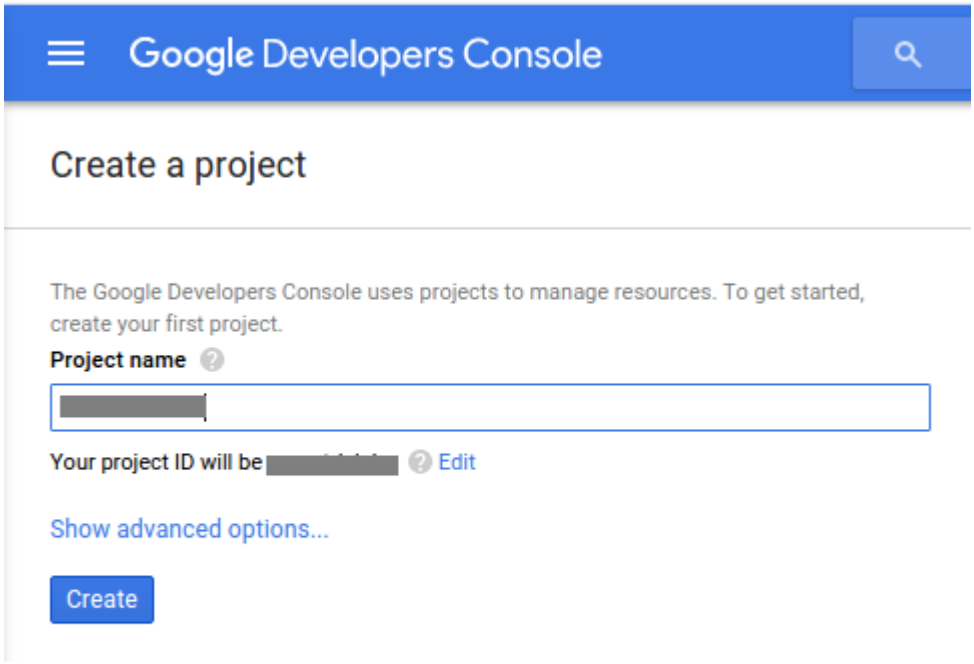
AndroidでGoogleドライブをする

Googleデベロッパーコンソールでしいプロジェクトをする

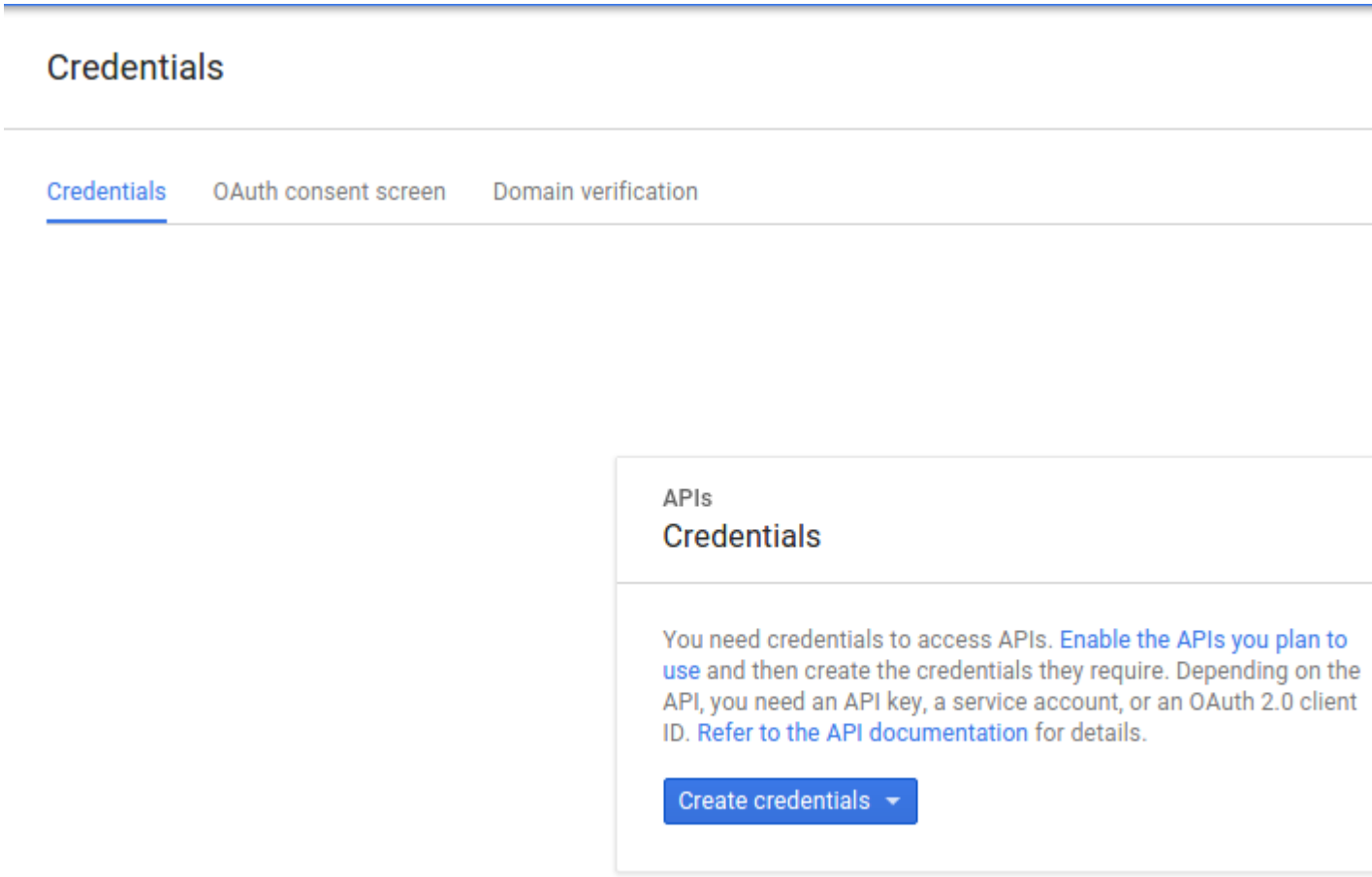
AndroidアプリケーションをGoogleドライブとするには、Google Developers Consoleでプロジェクトのをします。そのため、Google Developer Consoleでプロジェクトをするがあります。

Google Developer Consoleでプロジェクトをするには、のをします。

- AndroidGoogleデベロッパーコンソールにアクセスします。にプロジェクトをし、[ ]ボタンをクリックしてGoogleけコンソールにしいプロジェクトをします。

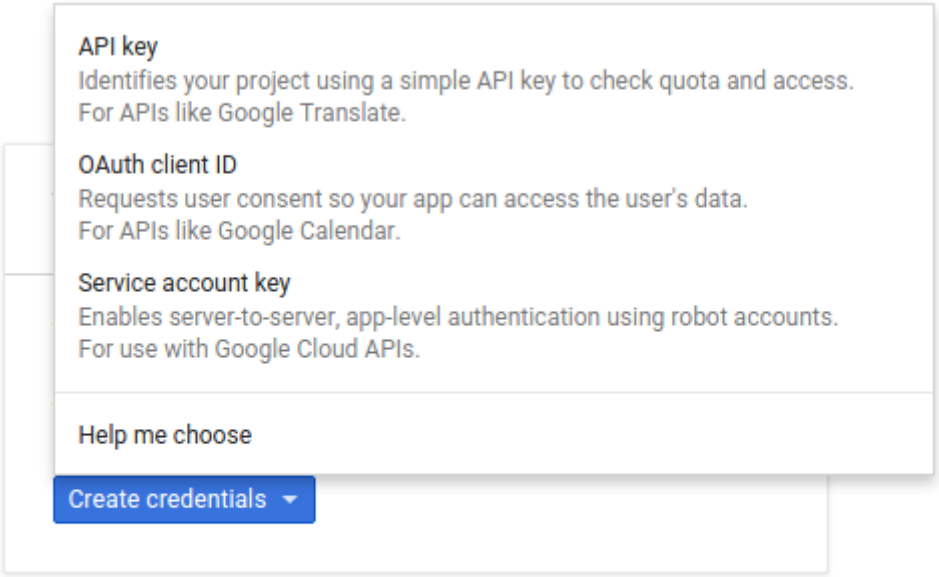


- APIにアクセスするためのをするがあります。したがって、 のボタンをクリックしてください。



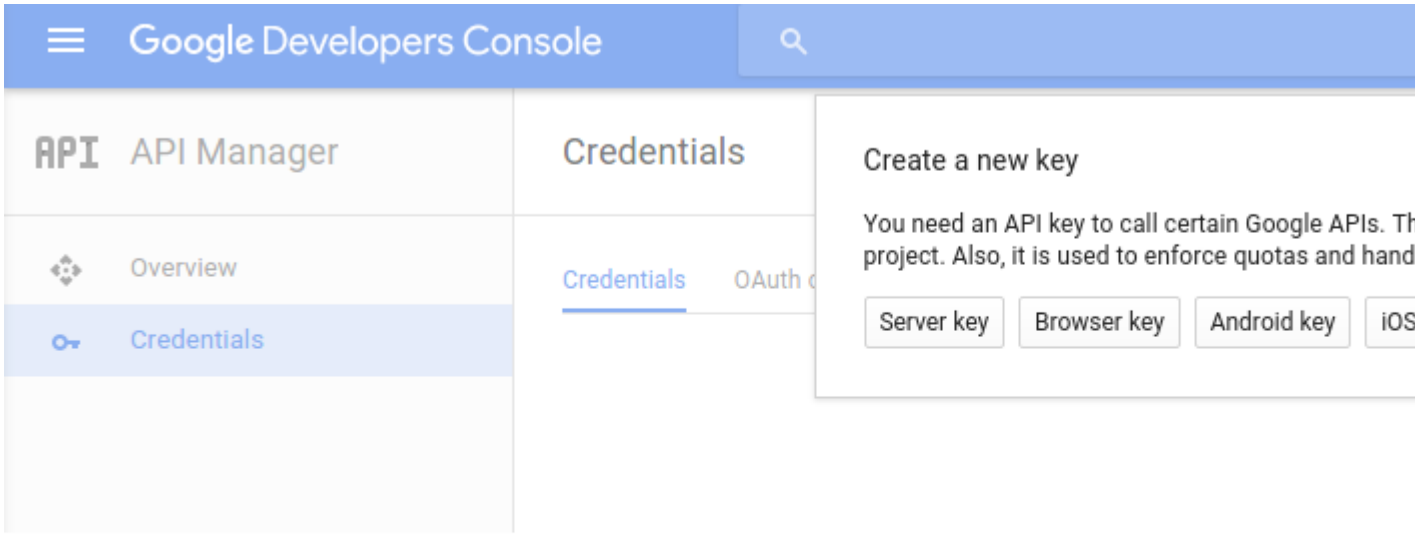
-

、ポップウィンドウがきます。 **API**キーをするには、リストの**API**キーオプションをクリッ

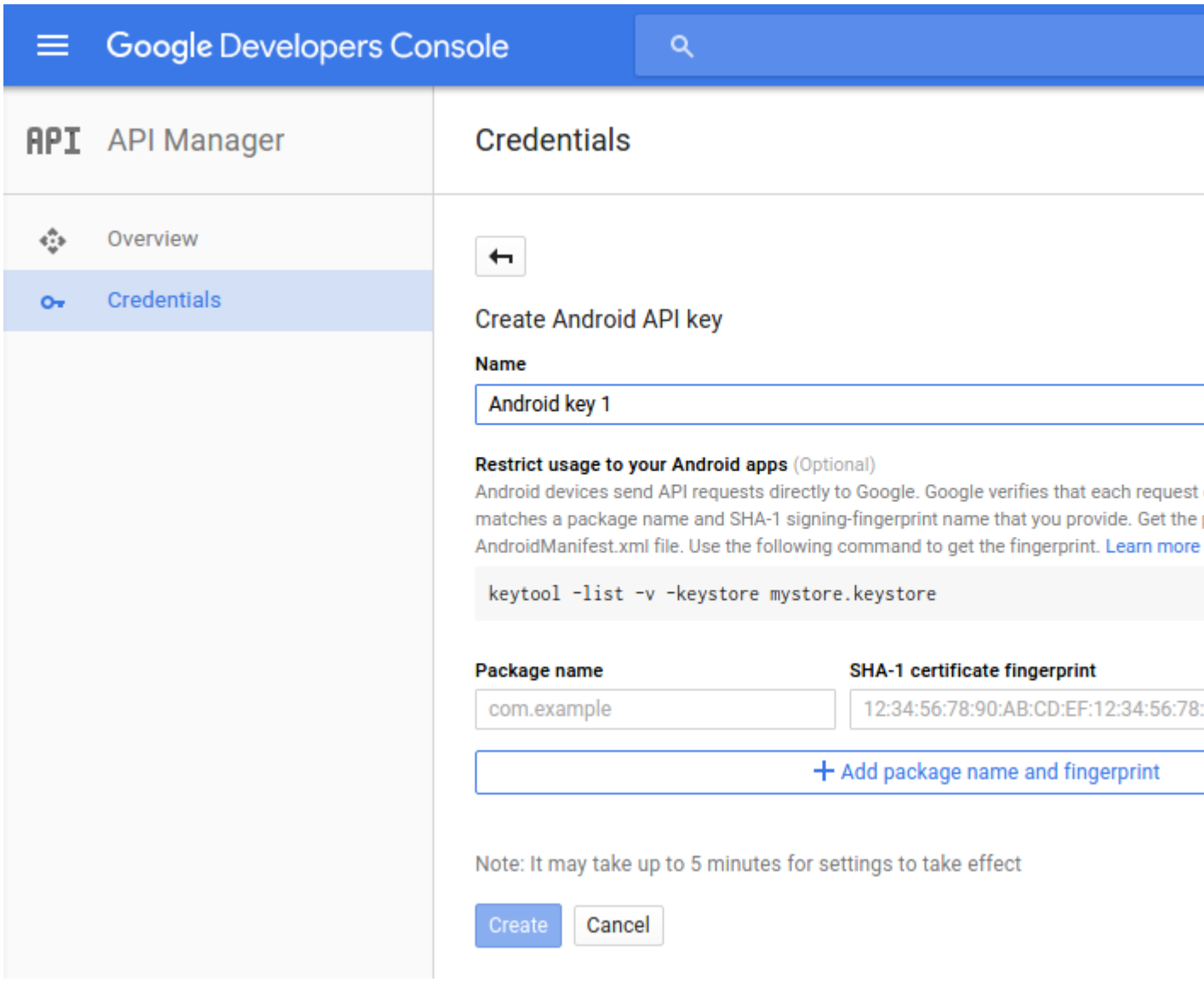


クします。

- AndroidGoogle APIをびすには、APIキーがです。 **Android Key**をクリックしてAndroidプロジェクトをします。



- に、APIキーをするために、フィールドにAndroid Projectのパッケージと**SHA-1**をするがあります。



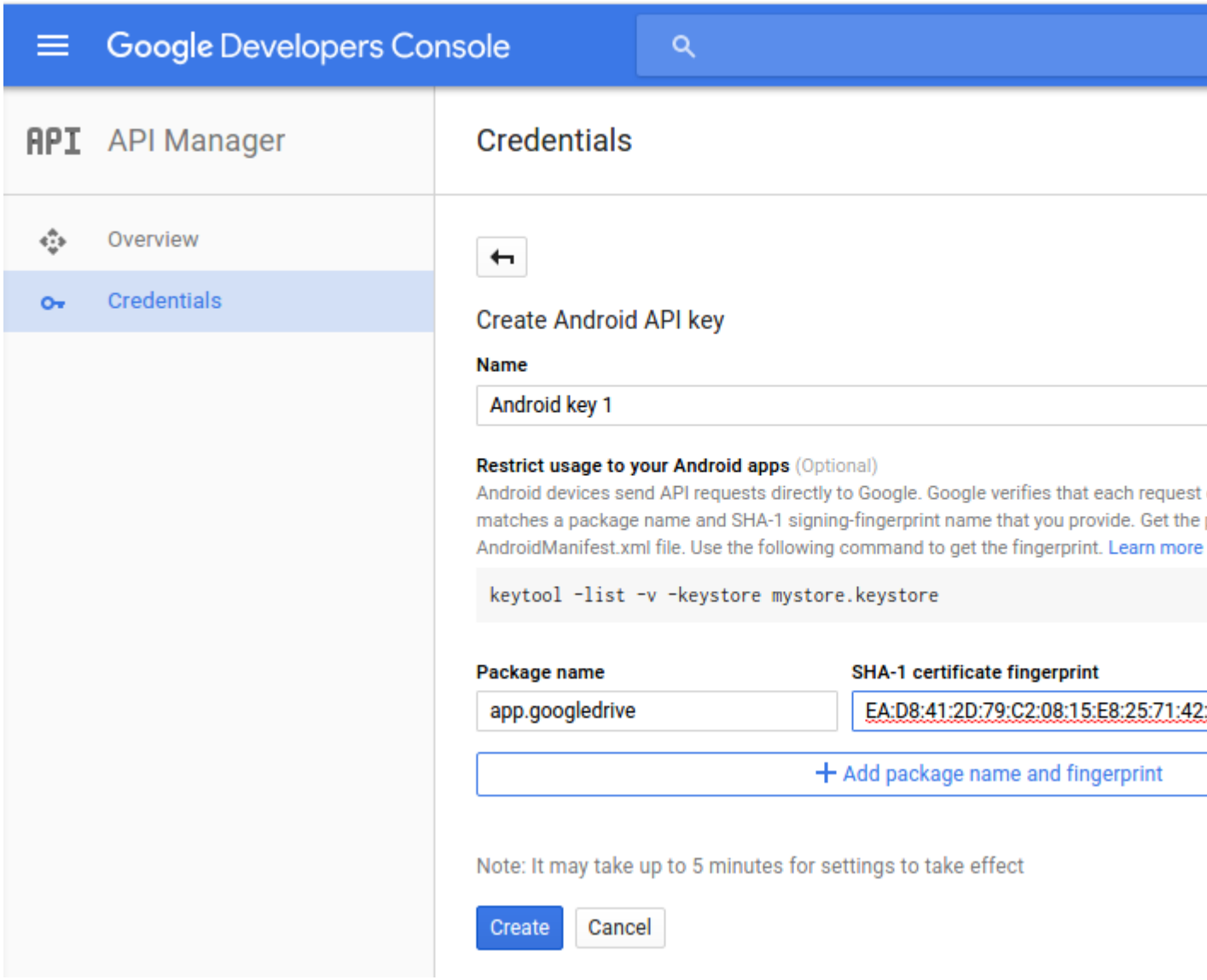
- **SHA-1**をするがあります。したがって、をいて**Keytool**ユーティリティをして、SHA1のをします。 Keytoolユーティリティをしているは、 キーストアのパスワードをするがあります。デフォルトのキーツールのパスワードは**"android"**です。 `keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -list -v`

```
[redacted]@[redacted]:~$ keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -list -v
Enter keystore password:
Alias name: androiddebugkey
Creation date: 18 Jul, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 3adbdb98
Valid from: Sat Jul 18 09:32:08 IST 2015 until: Mon Jul 10 09:32:08 IST 2045
Certificate fingerprints:
         MD5:  77:C7:A9:6A:30:0F:43:B9:84:E0:61:0F:B2:B6:22:74
         SHA1: EA:D8:41:2D:79:C2:08:15:E8:25:71:42:3F:0E:51:A5:52:4C:EF:40
         SHA256: A2:12:5A:18:E2:F3:FE:8B:93:E8:03:0C:12:3A:52:8D:B5:B0:70:32:C
F3:A7:C3:47:F0:9E:B6:8E:AF:33:68
         Signature algorithm name: SHA256withRSA
         Version: 3

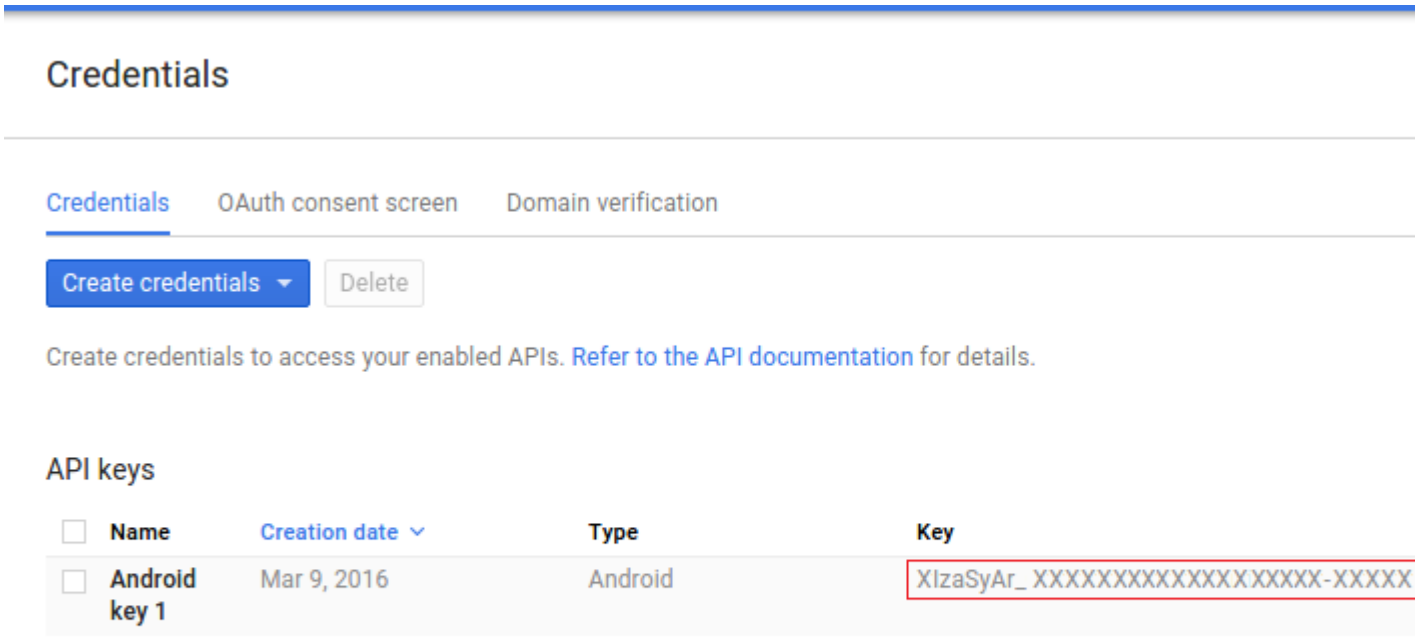
Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: D3 8F C7 0C 95 B4 DA 73   6B 67 99 5A A3 C0 05 4A  .......skg.Z...J
0010: 93 BE 25 4F                                        ..%O
]
]
```

- に、ページのフィールドにパッケージと**SHA-1**をします。に、ボタンをクリックしてAPIキーをします。



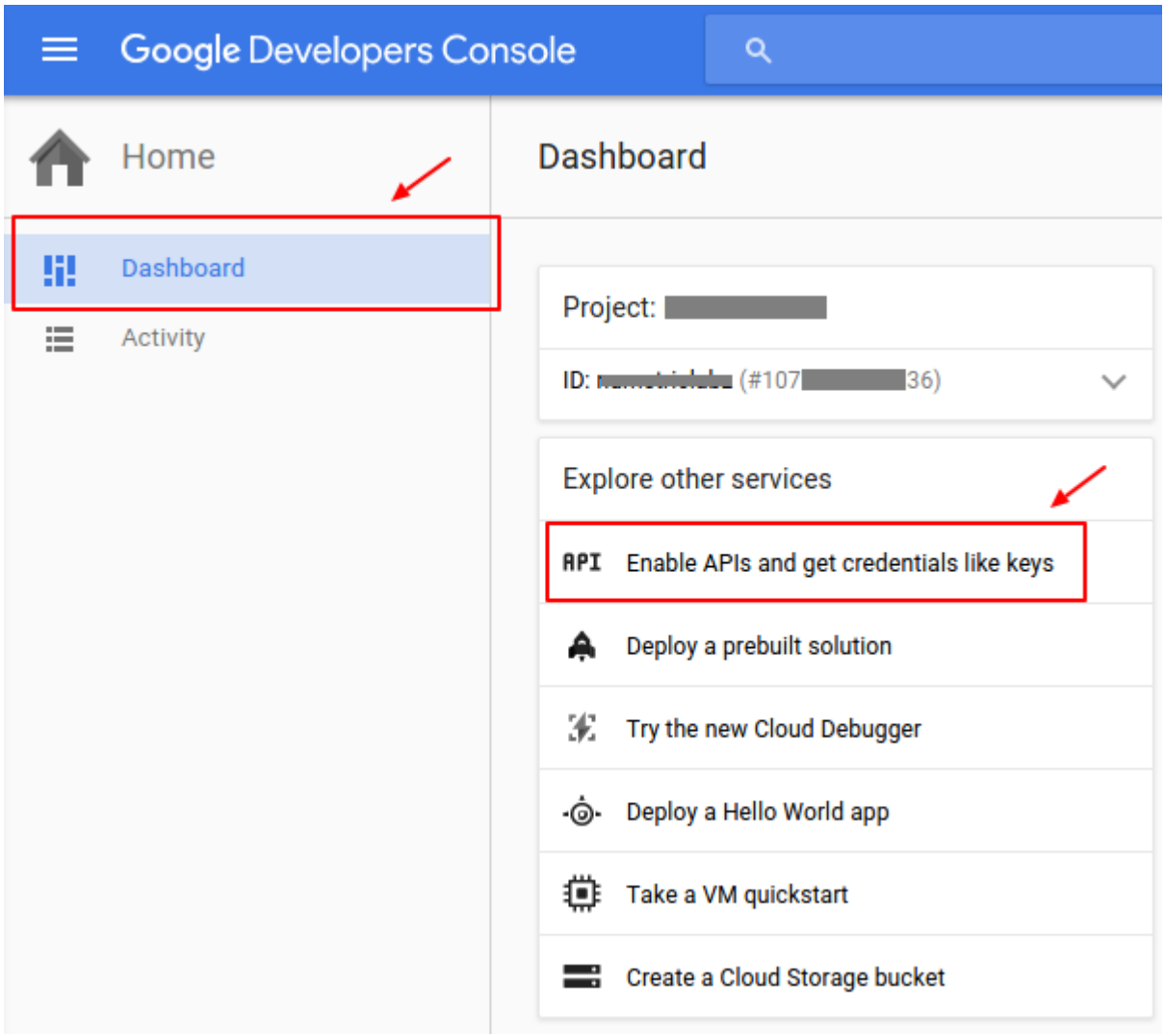
- これにより、AndroidのAPIキーがされます。このAPIキーをしてAndroidアプリとGoogleドライブをします。



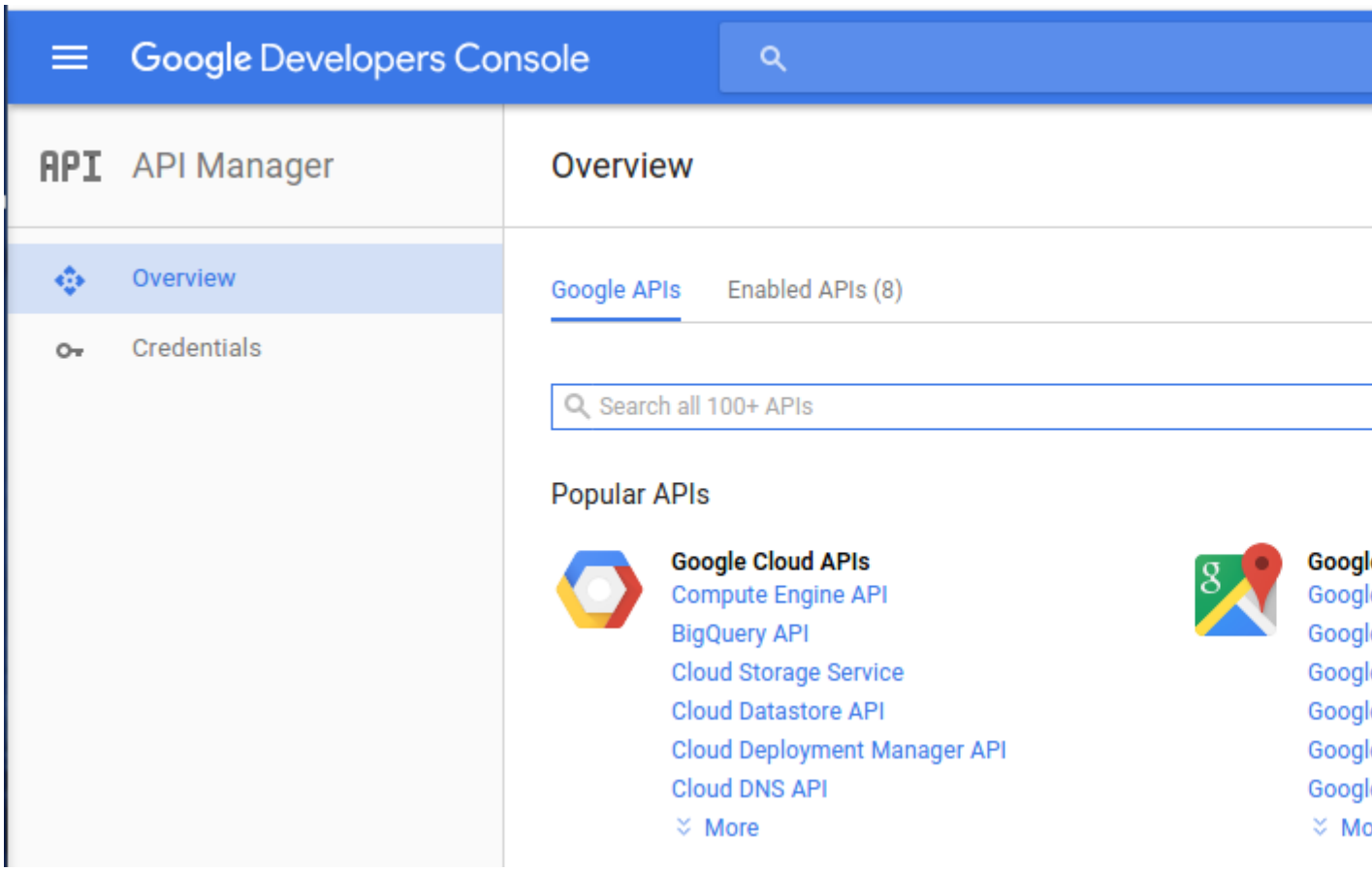
Googleドライブ**API**をにする

GoogleドライブAPIをにして、AndroidアプリケーションからGoogleドライブにされているファイルにアクセスするがあります。 GoogleドライブAPIをにするには、のをします。

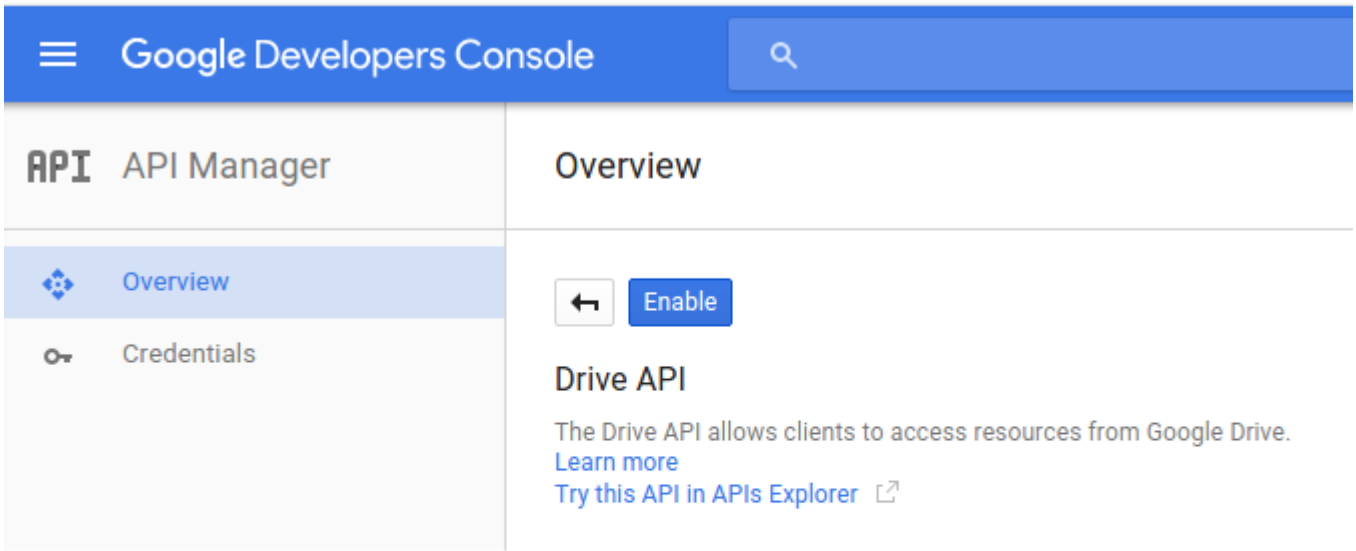
- Googleデベロッパーコンソールのダッシュボードにアクセスし 、[**API**をにする]をクリックすると、なGoogle APIがされます。



- ドライブ**API**リンクをクリックすると、Googleドライブ APIのページがきます。



- [にする]ボタンをクリックして、GoogleドライブAPIをにします。クライアントからGoogleドライブへのアクセスがされます。



インターネットをする

アプリケーションにインターネットアクセスGoogleドライブファイルがです。のコードをして、AndroidManifest.xmlファイルのインターネットアクセスをします。

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

Google Playサービスをする

Googleドライブ**Android API**を**Google Play**サービス**API**をします。 Androidアプリケーションで Google Playサービス SDKをセットアップするがあります。 `build.gradle` アプリモジュールファイルをき、Google Playサービス SDKをとしてします。

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

マニフェストファイルに**API**キーをする

AndroidアプリケーションでGoogle APIをするには、Google PlayサービスのAPIキーとバージョンをAndroidManifest.xmlファイルにするがあります。 AndroidManifest.xmlファイルのタグにしいメタデータタグをします。

Googleドライブ**Android API**のと

Googleドライブ**Android API**をし、Androidアプリケーションとするがあります。 **Google**ドライブの**Android API**は**GoogleApiClient**によってされます。たちは、onResumeメソッド**GoogleApiClient**をします。

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

```
    }
```

Google Deive Android APIのをする

アクティビティがすると、アクティビティの**onStop**メソッドで**disconnect**メソッドをびすことで、GoogleドライブAndroid APIとAndroidアプリケーションとのがされます。

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

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

コールバックとにしたリスナーをする

MainActivity.javaファイルにGoogle APIクライアントのConnection CallbackとConnection Failed Listenerをし、Google APIクライアントのをします。これらのリスナーは、 **onConnected**、onConnectionFailed、onConnectionSuspendedメソッドをして、appとDriveののをします。

ユーザーがアプリケーションをした、 **onConnected**メソッドがびされます。ユーザーがアプリケーションをしていないは、 **onConnectionFailed**メソッドがびされ、ユーザーにGoogleドライブへのアクセスがされていないというダイアログがされます。がされたは、**onConnectionSuspended**メソッドがびされます。

アクティビティに**ConnectionCallbacks**と**OnConnectionFailedListener**をするがあります。Javaファイルでのコードをします。

```
@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");
    }
```

Googleドライブにファイルをする

Googleドライブにファイルをします。 `Drive`オブジェクトの`createFile()`メソッドをして、プログラムでGoogleドライブにファイルをします。このでは、ユーザーのルートフォルダにしいテキストファイルをしています。ファイルがされると、メタデータのセット、ファイルの、およびフォルダをするがあります。

`CreateMyFile()`コールバックメソッドをし、このメソッドで`Drive`オブジェクトをして`DriveContents`リソースをするがあります。に、APIクライアントを`Drive`オブジェクトにし、 `DriveContents`をするために`driveContentsCallback`コールバックメソッドをびします。

`DriveContents`リソースには、アプリケーションでのみなファイルのバイナリストリームのコピーがまれています。

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

DriveContentのハンドラ

をするには、びしがしたかどうかをするがあります。びしがしたは、 `DriveContents`リソースをできます。

`DriveContents`ハンドラをします。このメソッドでは、 `CreateFileOnGoogleDrive()`メソッドをびし、`DriveContentsResult`のをします。

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

プログラムでファイルをする

ファイルをするには、 `MetadataChangeSet`オブジェクトをするがあります。このオブジェクトをして、タイトルファイルとファイルタイプをします。また、 `DriveFolder`クラスの`createFile()`メソッドをし、GoogleクライアントAPI、 `MetaDataChangeSet`オブジェクト、および`driveContents`を`driveContents`てファイルをするがあります。ハンドラコールバックをびして、されたファイルのをします。

のコードをして、ユーザのルートフォルダにしいテキストファイルをします。

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

されたファイルのをする

のコードは、されたファイルのをするコールバックメソッドをします。

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

オンラインでGoogleドライブAPIをむ https://riptutorial.com/ja/android/topic/10646/googleドライブapi

65: Googleログインの

- newInstance - Google Helperのインスタンスをする
- initGoogleSignIn - Googleログインをする
- getGoogleAccountDetails - ログインをする
- signOut - ユーザーをログアウトする
- getGoogleClient - GoogleApiClientをする

パラメーター

パラメータ	
タグ	ロギングにされる
GoogleSignInHelper	ヘルパーの
AppCompatActivity	アクティビティ
GoogleApiClient	GoogleAPIClientのリファレンス
RC_SIGN_IN	はアクティビティーをします。
isLoggingOut	ログアウトタスクがかどうかをするブール

Examples

ヘルパークラスによるGoogleログイン

`android`タグの`build.gradle`にをしてください

```
// Apply plug-in to app.
apply plugin: 'com.google.gms.google-services'
```

のヘルパークラスをあなたのutilパッケージにしてください

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

アクティビティファイルの`OnActivityResult`にのコーードをしてください

```
 // [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]
```

オンラインでGoogleログインのをむ https://riptutorial.com/ja/android/topic/2837/googleログインの

66: Googleの

Examples

な

にをするがあります

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

これをじファイルにれてください。

```
apply plugin: 'com.google.gms.google-services'
```

に、strings.xmlにをするがあります。

```
<string name="banner_ad_unit_id">ca-app-pub-####/####</string>
```

に、のビューとじように、どこでもきなにadviewをしてスタイルをします。

```
    <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>
```

に、あなたのonCreateにこれをげてください。

```
MobileAds.initialize(getApplicationContext(), "ca-app-pub-YOUR_ID");
AdView mAdView = (AdView) findViewById(R.id.adView);
    AdRequest adRequest = new AdRequest.Builder().build();
    mAdView.loadAd(adRequest);
```

コピーしてりけたは、さなバナーがされます。なにするだけで、よりくのAdViewをするだけです。

インターースティシャルの

インターースティシャルは、ホストアプリケーションのインターフェースをカバーするです。それらは、アクティビティやゲームのレベルのなど、アプリのフローのなポイントにされます。

Manifestファイルにながあることをしてください

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

1. AdMobアカウントにアクセスします。
2. [ ]タブをクリックします。
3. アプリをまたはし、プラットフォームをします。
4. インタースティシャルをし、ユニットをします。
5. ユニットがされると、ダッシュボードにユニットIDがされます。ca-app-pub-00000000000/000000000
6. をする

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

これはにあるはずです。

```
apply plugin: 'com.google.gms.google-services'
```

`strings.xml`ファイルにユニット**ID**をする

```
<string name="interstitial_full_screen">ca-app-pub-00000000/00000000</string>
```

マニフェストにConfigChangesとメタデータをする

```
<activity
            android:name="com.google.android.gms.ads.AdActivity"

android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScree

            android:theme="@android:style/Theme.Translucent" />
```

そして

```
<meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
```

アクティビティ

```
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();
        }
    }

}
```

このAdActivityはフルスクリーンをします。

オンラインでGoogleのをむ https://riptutorial.com/ja/android/topic/5984/googleの

67: GradleでApache Archivaに.aarファイルをする

Examples

な

```
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'
            }
        }
    }

}
```

オンラインでGradleでApache Archivaに.aarファイルをするをむ
https://riptutorial.com/ja/android/topic/6453/gradleでapache-archivaに-aarファイルをする

68: GreenDAO

き

GreenDAOは、がなローカルストレージにSQLiteデータベースをできるようにするオブジェクトリレーショナルマッピングライブラリです。

Examples

SELECT、INSERT、DELETE、UPDATEクエリのヘルパーメソッド

このでは、データのクエリをするときになメソッドをむヘルパークラスをします。にをたせるため、ここではJava Genericをしています。

```
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();
}
```

コンポジットプライマリキーをつGreenDAO 3.Xでエンティティをする

コンポジットキーをつのモデルをする、モデルEntityがそれらのをするためには、オブジェクトにしてのがです。

のSQLテーブルとエンティティのは、がオンラインストアのアイテムのレビューをするためのをしています。このでは、`customer_id`および`item_id`をキーにして、のとアイテムのに1つのレビューしかさせないようにします。

SQLテーブル

```
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)
);
```

は、エンティティクラスのフィールドのに`@Id`アノテーションと`@Unique`アノテーションをしますが、キーのはのようにします。

1. `@Index`クラスレベルののを`@Entity`。 valueプロパティには、キーをするフィールドのコンマりリストがまれます。キーのをするには、の`unique`プロパティをします。

2. GreenDAOは、すべてのエンティティをっているが`long`か、 `Long`キーとしてオブジェクトを。これをEntityクラスにするがありますが、これをするはありませんし、にをえるもありません。のでは、 `localID`とばれてい`localID`

エンティティ

```
@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() {}
}
```

GreenDao v3.Xをいめる

GreenDaoライブラリとGradleプラグインをしたら、にエンティティオブジェクトをするがあります。

エンティティ

エンティティは、データベースののデータをモデルするPlain Old Java Object *POJO*です。GreenDaoはこのクラスをしてSQLiteデータベースにテーブルをし、SQLをすることなくデータにアクセスしてするためのヘルパークラスをにします。

```
@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...

}
```

ワンタイム**GreenDao**セットアップ

アプリケーションがされるたびに、GreenDaoをするがあります。 GreenDaoは、このコードをApplicationクラスにすることをしています。

```
DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this, "mydatabase", null);
db = helper.getWritableDatabase();
DaoMaster daoMaster = new DaoMaster(db);
DaoSession daoSession = daoMaster.newSession();
```

GreenDaoヘルパークラス

エンティティオブジェクトの、GreenDaoはにデータベースとするヘルパークラスをします。これらは、されたエンティティオブジェクトのとにがけられ、 `Dao`がき、 `daoSession`オブジェクトからされます。

```
UsersDao usersDao = daoSession.getUsersDao();
```

このDaoオブジェクトとエンティティオブジェクトをして、くのなデータベースアクションをできるようになりました。

クエリ

```
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();
```

インサート

```
Users newUser = new User("John","Doe","jdoe@example.com");
```

```
usersDao.insert(newUser);
```

```
// Modify a previously retrieved user object and update
user.setLastname("Dole");
usersDao.update(user);
```

```
// Delete a previously retrieved user object
usersDao.delete(user);
```

オンラインでGreenDAOをむ https://riptutorial.com/ja/android/topic/1345/greendao

69: GreenRobot EventBus

- @SubscribethreadMode = ThreadMode.POSTINGpublic void onEventEventClassイベント{}

パラメーター

スレッドモード	
`ThreadMode.POSTING`	イベントがされたのとじスレッドでびされます。これがデフォルトモードです。
`ThreadMode.MAIN`	メインのUIスレッドでびされます。
`ThreadMode.BACKGROUND`	バックグラウンドスレッドでびされます。スレッドがメインスレッドでないは、そのスレッドがされます。メインスレッドの`EventBus`されているは、それがするのバックグラウンドスレッドがあります。
`ThreadMode.ASYNC`	のスレッドでびされます。

Examples

イベントオブジェクトの

イベントのには、にEventオブジェクトがです。イベントオブジェクトははにはなPOJOです。

```
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;
    }
}
```

イベントをける

イベントをけるためには、クラスを`EventBus`にするがあります。

```
@Override
public void onStart() {
    super.onStart();
    EventBus.getDefault().register(this);
```

```
}

@Override
public void onStop() {
   EventBus.getDefault().unregister(this);
   super.onStop();
}
```

そしてイベントにします。

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void handleEvent(ArbitaryEvent event) {
    Toast.makeText(getActivity(), "Event type: "+event.getEventType(),
Toast.LENGTH_SHORT).show();
}
```

イベントの

ィベントをするのは、Eventオブジェクトをしてするのとじくらいです。

```
EventBus.getDefault().post(new ArbitaryEvent(ArbitaryEvent.TYPE_1));
```

なイベントをす

まず、EventBusをモジュールのgradleファイルにするがあります。

```
dependencies {
    ...
    compile 'org.greenrobot:eventbus:3.0.0'
    ...
}
```

はイベントのモデルをするがあります。それは、たちがしたいものをんでいます。のところのクラスをるだけです。

```
public class DeviceConnectedEvent
{
}
```

これで、EventBusにしてイベントにするActivityコードをできます。

```
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
    }
}
```

この`Activity`は、 `onCreate()`メソッドで`EventBus`インスタンスをします。 `onStart()` / `onStop()`でイベントの/をいます。リスナーがスコープをったときにするか、 `Activity`がするがあることをえておくことがです。

に、イベントでびすメソッドをします。 `@Subscribe`アノテーションはEventBusに、イベントをするためにどのメソッドをすかをします。 `@Subscribe`にするには@Subscribeでされたメソッドをなくとも1つつがあります。そうしないとがスローされます。では、スレッドモードをします。これはEventBusにメソッドをびすスレッドをします。これはバックグラウンドスレッドからUIスレッドにをすになですそれがまさにたちがここでやっていることです。 `ThreadMode.MAIN`は、このメソッドがAndroidのメインUIスレッドでびされるため、なUIをうのはです。メソッドのははありません。 EventBusがしている`@Subscribe`アノテーションののえは、のです。がするり、イベントがポストされたときにびされます。

ィベントをするためににうがあります。このコードはの`Service`にまれます。

```
EventBus.getDefault().post(new DeviceConnectedEvent());
```

それはすべてそこにある EventBusはそのDeviceConnectedEventをけり、されているリスナーをべ、したメソッドをべ、DeviceConnectedEventをとしてし、びすスレッドでびすメソッドをします。

オンラインでGreenRobot EventBusをむ https://riptutorial.com/ja/android/topic/3551/greenrobot-eventbus

70: Gson

き

Gsonは、JavaオブジェクトをJSONにするためにできるJavaライブラリです。 Gsonは、これらのをになとえています。

Gsonの

な`toJson()`メソッドと`fromJson()`メソッドをして、JavaオブジェクトをJSONにします。もです。

のなオブジェクトをJSONとのでできるようにする

Java Genericsのなサポート

のなオブジェクトをサポートしますいとタイプのな

- Excluder excluder
- FieldNamingStrategy fieldNamingStrategy
- <T> fromJsonJsonElement json、Class <T> classOfT
- <T> T fromJsonJsonElement json、Type typeOfT
- <T> T fromJsonJsonReaderリーダー、Type typeOfT
- <T> T fromJsonReader json、Class <T> classOfT
- <T> T fromJsonReader json、Type typeOfT
- <T> T fromJsonString json、Class <T> classOfT
- <T> T fromJsonString json、Type typeOfT
- <T> TypeAdapter <T> getAdapterClass <T>
- <T> TypeAdapter <T> getAdapterTypeToken <T>
- <T> TypeAdapter <T> getDelegateAdapterTypeAdapterFactory skipPast、TypeToken <T>
- JsonReader newJsonReaderリーダーリーダー
- JsonWriter newJsonWriterライターライター
- JsonElement toJsonTreeオブジェクトsrc
- JsonElement toJsonTreeObject src、Type typeOfSrc
- ブールserializeNulls
- boolean htmlSafe
- String toJsonJsonElement jsonElement
- String toJsonObject src
- String toJsonObject src、Type typeOfSrc
- String toString
- void toJsonObject src、Type typeOfSrc、なライター
- void toJsonObject src、Type typeOfSrc、JsonWriterライター
- void toJsonJsonElement jsonElement、なライター
- void toJsonJsonElement jsonElement、JsonWriterライター

- void toJsonObject src、なライター

Examples

JSONをGsonでする

このでは、GoogleのGsonライブラリをしてJSONオブジェクトをしています。

オブジェクトの

```
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 = "";
    }


}
```

に、をうがあるは、のようにします。

```
String robotJson = "{
                        \"version\": \"JellyBean\",
                        \"age\": 3,
                        \"robotName\": \"Droid\"
                    }";

Gson gson = new Gson();
Robot robot = gson.fromJson(robotJson, Robot.class);
```

リストの

JSONオブジェクトのリストをするときに、それらをしてJavaオブジェクトにしたいことがよくあります。

しようとするJSONはのとおりです。

```
{
```

```
  "owned_dogs": [
    {
      "name": "Ron",
      "age": 12,
      "breed": "terrier"
    },
    {
      "name": "Bob",
      "age": 4,
      "breed": "bulldog"
    },
    {
      "name": "Johny",
      "age": 3,
      "breed": "golden retriever"
    }
  ]
}
```

こののJSONには3つのオブジェクトがまれています。 Javaコードでは、これらのオブジェクトを`Dog`オブジェクトにマップするがあります。 Dogオブジェクトはのようになります。

```
private class Dog {
    public String name;
    public int age;

    @SerializedName("breed")
    public String breedName;
}
```

JSONを`Dog[]`するには

```
Dog[] arrayOfDogs = gson.fromJson(jsonArrayString, Dog[].class);
```

`Dog[]`をJSONにする

```
String jsonArray = gson.toJson(arrayOfDogs, Dog[].class);
```

JSONを`ArrayList<Dog>`にするには、のようにします。

```
Type typeListOfDogs = new TypeToken<List<Dog>>(){}.getType();
List<Dog> listOfDogs = gson.fromJson(jsonArrayString, typeListOfDogs);
```

Typeオブジェクト`typeListOfDogs`は、 `Dog`オブジェクトのリストがどのようにえるかをします。 GSONはこのオブジェクトをして、JSONをしいにマップできます。

また、 `List<Dog>`をJSONにするもです。

```
String jsonArray = gson.toJson(listOfDogs, typeListOfDogs);
```

GsonでJSONプロパティをenumにする

GsonでするStringをするは、のようにします。

{"status" "open"}

```
public enum Status {
    @SerializedName("open")
    OPEN,
    @SerializedName("waiting")
    WAITING,
    @SerializedName("confirm")
    CONFIRM,
    @SerializedName("ready")
    READY
}
```

リストの Gsonと

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());
```

これは、パラメータされたのクラスをできないためつまり、 `List<String>.class`びすことができないため、もなコンテナクラスにです。

2

```
public class StringList extends ArrayList<String> { }

...

List<String> members = gson.fromJson(json, StringList.class);
```

あるいは、なをにサブクラスし、そのクラスをすこともできます。しかし、 `StringList`のオブジェクトをすので、これはにベストプラクティスではありません。

AutoValueとGsonをしたJSONシリアライゼーション/デシリアライゼーション

あなたのgradleルートファイルでのインポート

```
classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
```

あなたのgradleアプリケーションファイルでのインポート

```
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'
```

でオブジェクトをする

```
@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);
    }
}
```

GsonBuilderでGsonコンバ―タをする

```
Gson gson = new GsonBuilder()
                .registerTypeAdapterFactory(
                    new AutoValueGsonTypeAdapterFactory())
                .create());
```

デシリアライズする

```
String myJsonData = "{
    \"signin_token\": \"mySigninToken\",
    \"username\": \"myUsername\" }";
SignIn signInData = gson.fromJson(myJsonData, Signin.class);
```

シリアライズ

```
Signin myData = SignIn.create("myTokenData", "myUsername");
String myJsonData = gson.toJson(myData);
```

Gsonをすると、POJOオブジェクトをしてシリアルおよびシリアルコ―ドをすることができます。は、リフレクションがコストパフォ―マンスにれていることです。そのため、AutoValue-Gsonをしてカスタムタイプアダプターをすると、リフレクションコストがされ、APIのがしたときにするのはにです。

GsonをったJSONのクラスオブジェクトへの

JSONがあるとします。

```
["first","second","third"]
```

このJSONをして`String`にすることができます

```
Gson gson = new Gson();
```

```
String jsonArray = "[\"first\",\"second\",\"third\"]";
String[] strings = gson.fromJson(jsonArray, String[].class);
```

しかし、それを`List<String>`オブジェクトにするは、 `TypeToken`をするがあります。

ここにサンプルがあります

```
Gson gson = new Gson();
String jsonArray = "[\"first\",\"second\",\"third\"]";
List<String> stringList = gson.fromJson(jsonArray, new TypeToken<List<String>>()
{}.getType());
```

の2つのクラスがあるとします。

```
public class Outer<T> {
    public int index;
    public T data;
}

public class Person {
    public String firstName;
    public String lastName;
}
```

JSONは、 `Outer<Person>`という`Outer<Person>`オブジェクトにするがあります。

のは、このJSONをするクラスオブジェクトにするをしています。

```
String json = "......";
Type userType = new TypeToken<Outer<Person>>(){}.getType();
Result<User> userResult = gson.fromJson(json,userType);
```

JSONを`Outer<List<Person>>`オブジェクトにするがある

```
Type userListType = new TypeToken<Outer<List<Person>>>(){}.getType();
Result<List<User>> userListResult = gson.fromJson(json,userListType);
```

あなたのプロジェクトに**Gson**をする

```
dependencies {
    compile 'com.google.code.gson:gson:2.8.1'
}
```

Gsonのバ―ジョンをするには

のはコンパイルするたびにgsonライブラリのバ―ジョンをコンパイルしますが、バ―ジョンをするはありません。

の、スピ―ド、バグのないをできます。
コ―ドとのがなわれるがあります。

```
compile 'com.google.code.gson:gson:+'
```

Gsonをしてディスクから JSONファイルをロードする

これにより、ディスクからJSONファイルがロードされ、されたタイプにされます。

```
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);
}
```

Gsonにカスタムコンバータをする

によっては、いくつかのフィールドをのフォーマットでまたはするがあります。たとえば、バックエンドでに "YYYY-MM-dd HHmm"というをし、POJOSでJoda TimeでDateTimeクラスをすることができます。

これらのをDateTimesオブジェクトににするには、カスタムコンバータをできます。

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

Gsonにしくされたコンバータをさせるには、Gsonオブジェクトのにそれをりてるがあります

```
    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
```

そのでをシリアルするには、DateTimeのフィールドをするだけです。

```
public class SomePojo {
    private DateTime someDate;
}
```

GsonはDateTimeのフィールドをすると、フィールドをシリアルするためにコンバータをびします。

Retrofitでシリアライザとして Gsonをする

まず、build.gradleファイルに`GsonConverterFactory`をするがあります

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

に、Retrofit Serviceをするときにコンバータファクトリをするがあります。

```
Gson gson = new GsonBuilder().create();
new Retrofit.Builder()
        .baseUrl(someUrl)
        .addConverterFactory(GsonConverterFactory.create(gson))
        .build()
        .create(RetrofitService.class);
```

ファクトリにすGsonオブジェクトをするときにカスタムコンバータをすることができます。カスタムをできるようにします。

Gsonをしてジェネリッククラスにjsonをする

jsonがあるとします。

```
{
  "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"
    }
  ]
}
```

このをカスタムツイートツイートのリストコンテナオブジェクトにですることはできますが、`fromJson`メソッドを`fromJson`するがです。

```
Gson gson = new Gson();
String jsonArray = "....";
Tweets tweets = gson.fromJson(jsonArray, Tweets.class);
```

の2つのクラスがあるとします。

```
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;
}
```

jsonをするだけであれば、このコードをすることができます

```
String tweetsJsonArray = "[{.....},{.....}]"
List<Tweet> tweets = gson.fromJson(tweetsJsonArray, new TypeToken<List<Tweet>>()
{}.getType());
```

GsonをしたカスタムJSONデシリアライザ

たとえば、`/Date(1465935152)/`ように、すべてののすべてのがあるカスタム･フォーマットであるとし、すべてのJsonのをJavaの`Date`インスタンスにシリアルするがあるとします。このは、カスタムJson Deserializerをするがあります。

jsonの

```
{
```

```
  "id": 1,
  "created_on": "Date(1465935152)",
  "updated_on": "Date(1465968945)",
  "name": "Oleksandr"
}
```

のクラスがあるとします。

```
class User {
    @SerializedName("id")
    long id;
    @SerializedName("created_on")
    Date createdOn;
    @SerializedName("updated_on")
    Date updatedOn;
    @SerializedName("name")
    String name;
}
```

カスタムデシリアライザ

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

そして

```
Gson gson = new GsonBuilder()
                .registerTypeAdapter(Date.class, new DateDeSerializer())
                .create();
String json = "....";
User user = gson.fromJson(json, User.class);
```

Jacksonの**JSON**を**Date**でシリアルおよびシリアルする

これは、たとえば、GsonのをJacksonとをたせたいにもてはまります。

Jacksonは、、Dateを "エポックからのミリ"にシリアルしますが、Gsonは`Aug 31, 2016 10:26:17`ようなみりなをしてDateをします。ジャクソンのをシリアルしようとすると、GsonのJsonSyntaxExceptionsがします。

これをするために、カスタムシリアライザとカスタムデシリアライザをできます。

```
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();
```

Gsonをでする

Gsonはボックスからのをサポ―トしていません。

たちはのクラスをっているとしましょう

```
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;
    }
}
```

そして、 `DerivedClass1`インスタンスをJSONにシリアルしたい

```
DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;

Gson gson = new Gson();
String derivedClass1Json = gson.toJson(derivedClass1);
```

さて、のでは、このjsonをけり、それをシリアルしたいのですが、コンパイルには、 BaseClassインスタンスであることがBaseClassます。

```
BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());
```

しかしGSONはりませんderivedClass1JsonもともとのインスタンスだったDerivedClass1 、これは10をプリントアウトします。

これをするには

そのようなケースをするのJsonDeserializerをするがあります。はにきれいではありませんが、はよりいものをえすことができませんでした。

まず、クラスにのフィールドをします

```
@SerializedName("type")
private String typeName;
```

そしてそれをクラスのコンストラクタでする

```
public BaseClass() {
    typeName = getClass().getName();
}
```

のクラスをします。

```
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);
 }
```

```
}
```

すべては、すべてをフックすることがっている -

```
GsonBuilder builder = new GsonBuilder();
 builder
 .registerTypeAdapter(BaseClass.class, new JsonDeserializerWithInheritance<BaseClass>());
 Gson gson = builder.create();
```

そして、のコーードをして、

```
 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());
```

5がされます。

オンラインでGsonをむ https://riptutorial.com/ja/android/topic/4158/gson

71: HttpURLConnection

- abstract void disconnect
- abstract boolean usingProxy
- static boolean getFollowRedirects
- static void setFollowRedirectsboolean set
- String getHeaderFieldint n
- String getHeaderFieldKeyint n
- String getRequestMethod
- String getResponseMessage
- int getResponseCode
- long getHeaderFieldDateString name、longデフォルト
- boolean getInstanceFollowRedirects
- パーミッションgetPermission
- InputStream getErrorStream
- void setChunkedStreamingModeint chunklen
- void setFixedLengthStreamingModeint contentLength
- void setFixedLengthStreamingModelong contentLength
- void setInstanceFollowRedirectsboolean followRedirects
- void setRequestMethodString method

`HttpURLConnection`は、Webでデータをするためにされる、AndroidのHTTPクライアントです。HTTPRFC 2616のURLConnectionのなです。

Examples

HttpURLConnectionの

しいAndroid HTTPクライアント`HttpURLConnection`をするには、URLインスタンスで`openConnection()`をびします。 `openConnection()`は`URLConnection`すので、されたをにキャストするがあります。

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
// do something with the connection
```

しい`URL`するは、URLにするもするがあります。

```
try {
    URL url = new URL("http://example.com");
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    // do something with the connection
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

レスポンスボディがみまれ、がになったら、 `disconnect()`びすことで`disconnect()`するがあります。

にをします。

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
try {
    // do something with the connection
} finally {
    connection.disconnect();
}
```

HTTP GETリクエストをする

```
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();
}
```

のではがされないことにしてください。 なをむなはのとおりです。

```
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();
    }
}
```

HTTP GETリクエストのをむ

```
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();
}
```

のではがされないことにしてください。

multipart / form-dataにHttpURLConnectionをする

multipart / form-data HttpURLConnectionリクエストをびすカスタムクラスをする

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;
        }
    }
```

それをするの

```
    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;
                }
```

パラメータきHTTP POSTリクエストの

HashMapをして、POSTパラメーターをしてサーバーにするパラメーターをします。

```
HashMap<String, String> params;
```

`params` HashMapにがされたら、それらをサーバーにするためにされるStringBuilderをします。

```
    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++;
    }
```

に、HttpURLConnectionをし、をき、POSTパラメータをします。

```
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();
}
```

に、サーバーからされたをけります。

```
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();
    }
}
```

HttpURLConnectionをしたアップロードPOSTファイル

リモートサーバにファイルを/アップロードするためには、イメージ、ビデオ、オーディオ、またはアプリケーションデータベースのバックアップをリモートプライベートサーバにするがあります。サーバーがコンテンツのPOSTリクエストをしているとして、Androidでこのタスクをするのなをにします。

ファイルアップロードは、 `multipart/form-data` POSTリクエストをしてされます。がにです

```
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
}
```

もちろん、はスロ―されるか、されるがあります。このコ―ドについてするはのとおりです。

1. `postTarget`はPOSTのURLです。 `oauthToken`はトークンです。 `fileDescription`はファイルので、フィ―ルドの`description`としてされます。 `file`はされるファイルです。タイプは`java.io.File`です。ファイルパスがあるは、わりに`new File(filePath)`できます。
2. oAuthの`Authorization`ヘッダ―をします。
3. これはApache Common `FileUtil`をしてファイルをバイトにみみます。ファイルのがにバイトやそののでメモリにされているは、みむはありません。

すべてのタイプの**HTTP**をするための**HttpURLConnection**クラス

のクラスは、 `GET` 、 `POST` 、 `PUT` 、 `PATCH` 、およびそののをできるのクラスとしてできます。

```
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();
    }
}
```

`POST`データをするか、なヘッダをるがあるかにじて、クラスのされたコンストラクタをします。

データのフェッチがすると、 `onComplete()`メソッドがびされます。データは`APIResponseObject`クラスのオブジェクトとしてされます。このクラスには、のHTTPステータスコードとレスポンスをむをすステータスコードがまれています。このレスポンスは、クラスXMLまたはJSONでできます。

のにすように、クラスのオブジェクトにして`execute()`をびしてをします。

```
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();
}
```

オンラインでHttpURLConnectionをむ https://riptutorial.com/ja/android/topic/781/httpurlconnection

72: ImageView

き

ImageView `android.widget.ImageView` は、DrawablesやBitmapsなどのリソースをおよびするためのビューです。

このトピックでするいくつかのエフェクトをイメージにすることができます。イメージソースは、XMLファイル `layout` フォルダにすることも、プログラムコードでJavaコードですることもできます。

- `setImageResource(int resId)` メソッドは、Drawableをこの `ImageView` としてし `ImageView` 。
- `imageView.setImageResource(R.drawable.anyImage)`

パラ メーター

パラ メータ	
`resId`	`res` フォルダのファイルは `drawable` フォルダ

Examples

リソースをする

```
<ImageView
 android:id="@+id/imgExample"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 ...
 />
```

XMLをして `ImageView` を `ImageView` コンテンツとしてする

```
android:src="@drawable/android2"
```

プログラムでドロアブルをする

```
 ImageView imgExample = (ImageView) findViewById(R.id.imgExample);
 imgExample.setImageResource(R.drawable.android2);
```

アルファをする

「アルファ」は、のをするためにされます。

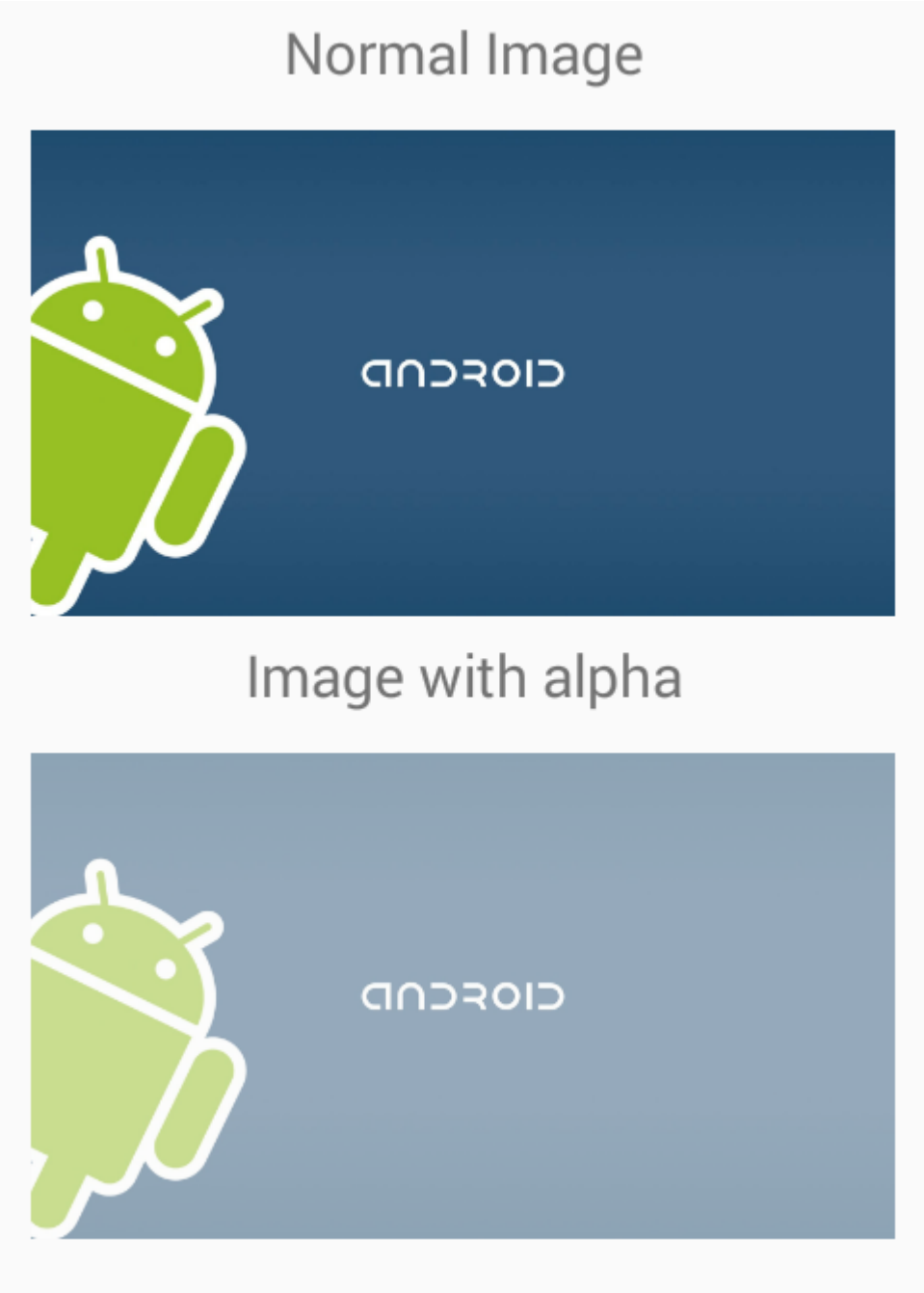
XMLをしてアルファをする

```
android:alpha="0.5"
```

0から1にまでのfloatをとります。

プログラムでアルファをする

```
imgExample.setAlpha(0.5f);
```



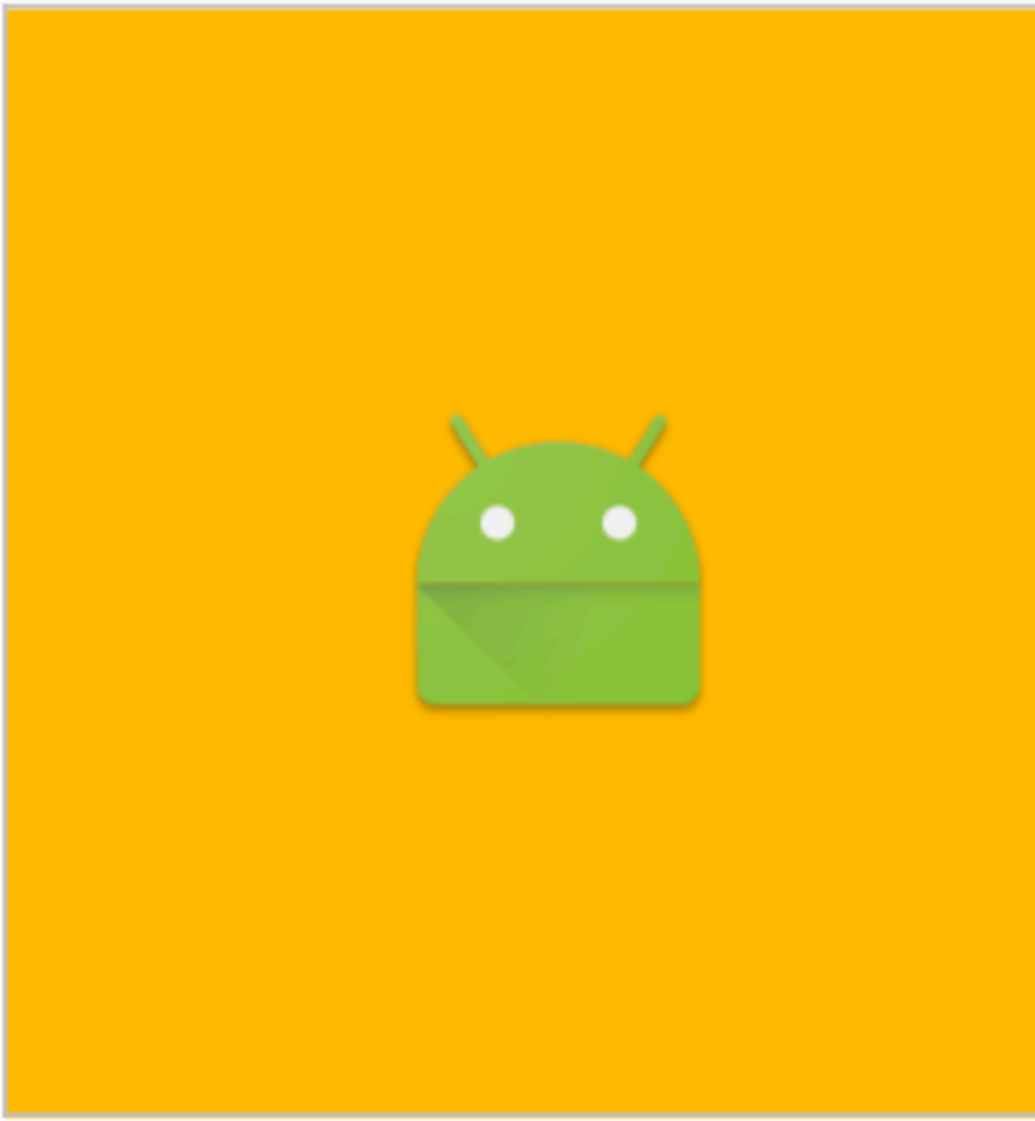
ImageView ScaleType - センター

ImageViewにまれるイメージは、コンテナにえられたなサイズにまらないがあります。その、フレームワークでさまざまなでのサイズをできます。

センター

```
    <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"/>
```

これはイメージのサイズをせず、コンテナのにしますオレンジ=コンテナ



ィメージをに`centerCrop`ますイメージのをします。ィメージは、ビュのにされます。



3. `centerInside` イメージのサイズとさがビュのするパディングなしにしくなるように、イメージをに/ `centerInside`ますイメージのをします。ィメージは、ビュのにされます。



4. `matrix` にイメージをしてスケルします。



5. `fitXY` FILLをしてを`fitXY`ます。



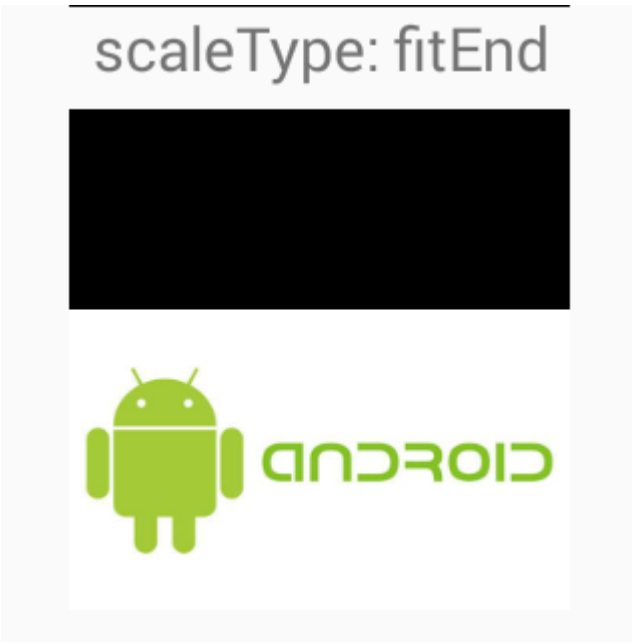
6. `fitStart` STARTをしてを`fitStart`ます。



7. `fitCenter` CENTERをしてを`fitCenter`ます。



8. `fitEnd` ENDをしてイメージを`fitEnd`ます。



いをする

のいをします。デフォルトでは、いは`SRC_ATOP`モードをしてブレンドされます。

XMLをしていをする

```
android:tint="#009c38"
```

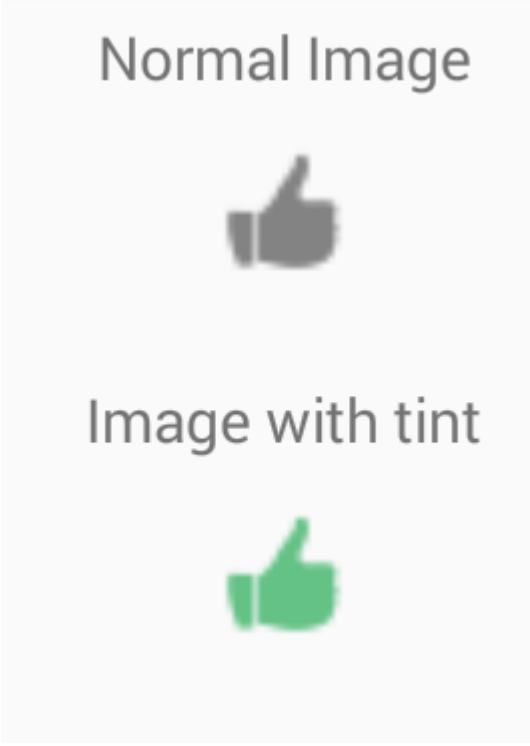
`"#rgb"`、 `"#argb"`、 `"#rrggbb"`、または`"#aarrggbb"`ののカラーでなければなりません。

プログラムでティントをする

```
imgExample.setColorFilter(Color.argb(255, 0, 156, 38));
```

このカラーフィルタをクリアすることができます

```
imgExample.clearColorFilter();
```



MLRoundedImageView.java

パッケージののクラスをコピーペーストする

```
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;
    }
}
```

`ImageView`ではなくパッケージでこのクラスをXMLでする

```
<com.androidbuts.example.MLRoundedImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />
```

オンラインでImageViewをむ https://riptutorial.com/ja/android/topic/4709/imageview

73: IntentService

4. <service androidname = "。UploadS3IntentService" androidexported = "false" />

`IntentService`は、バックグラウンドスレッドでをオフロードするなをします。リクエストをし、キューにれたり、をしたりするなど、すべてのをいます。がで、メインUIスレッドにさないにがかかるにするのにです。

Examples

IntentServiceの

、IntentServiceをしするクラスをするために`IntentService`、オーバーライドする、びその`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.
     }
}
```

サンプル インテント サービス

ここでは、バックグラウンドでをロード`IntentService`うとする`IntentService`をします。`IntentService`をするためになのは、 `super(String)`コンストラクタをびすコンストラクタをすることだけで、 `onHandleIntent(Intent)`メソッドをするがあります。

```
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
    }
}
```

IntentServiceをするには、 IntentをIntentServiceにするがあります。 Activityからうことができます。もちろん、あなたはそれにされません。ここでは、しいServiceをActivityクラスからびすのをします。

```
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はIntentからのデータをIntentServiceするため、のIntentをいにさせるなしにできます。に1つのIntentのみがされ、りはキューにれられます。すべてのジョブがすると、IntentServiceはにシャットダウンします。

なIntentServiceの

クラスIntentServiceは、サービスのクラスであり、ユーザーインターフェイスなしでバックグラウンドでされます。したがって、UIをするには、をするがあります。は、 BroadcastReceiverまたはResultReceiverいずれかResultReceiver。

- サービスがをリッスンしたいのコンポーネントとするがあるは、 BroadcastReceiverをするがあります。
- ResultReceiver あなたのサービスがアプリケーションつまりあなたのアプリケーションだけとするがあるにします。

IntentServiceには、 onHandleIntent()なメソッドが1つあります。このメソッドでは、の、アラームのなど、すべてのをいます。

あなたのIntentServiceをいたいは、のようにするがあります

```
public class YourIntentService extends IntentService {
    public YourIntentService () {
        super("YourIntentService ");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        // TODO: Write your own code here.
    }
}
```

アクティビティのびし/は、のようにうことができます。

```
Intent i = new Intent(this, YourIntentService.class);
startService(i);  // For the service.
startActivity(i); // For the activity; ignore this for now.
```

のアクティビティとに、バンドルデータなどのをのようにすことができます。

```
Intent passDataIntent = new Intent(this, YourIntentService.class);
msgIntent.putExtra("foo","bar");
startService(passDataIntent);
```

ここで、いくつかのデータを`YourIntentService`クラスにしたとします。このデータについて、のようにアクションをできます。

```
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.
        }
    }
}
```

のコードでは、 `OnHandleIntent()`メソッドでをするもしています。

オンラインでIntentServiceをむ https://riptutorial.com/ja/android/topic/5319/intentservice

74: JCodec

Examples

MavenをってJCodecをにすることができます。このためには、のスニペットをpom.xmlにしてください。

```
<dependency>
    <groupId>org.jcodec</groupId>
    <artifactId>jcodec-javase</artifactId>
    <version>0.1.9</version>
</dependency>
```

ムービーからフレームをする

ムービーから1つのフレームをするAVC、MP4、ISO BMF、QuicktimeコンテナのH.264のみサポート

```
int frameNumber = 150;
BufferedImage frame = FrameGrab.getFrame(new File("filename.mp4"), frameNumber);
ImageIO.write(frame, "png", new File("frame_150.png"));
```

ムービーからのフレームをするAVC、MP4、H.264、ISO BMF、Quicktimeコンテナのみサポート

```
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);
}
```

オンラインでJCodecをむ https://riptutorial.com/ja/android/topic/9948/jcodec

75: JSON with Android with org.json

- オブジェクト オブジェクトはとのペアのけられていないセットです。オブジェクトは{でまり}でわります。それぞれののにコロンと/のペアが、カンマでられます。
- はけられたのです。は[でまり、]でわります。は、カンマでられます。
- は、でまれた、、またはまたはまたはヌル、またはオブジェクトまたはです。これらのはれにすることができます。
- は、バックスラッシュエスケープをしてでまれた0のUnicodeのシーケンスです。はのとしてされます。はCまたはJavaににています。
- **Number** 8と16のがされていないことをいて、はCまたはJavaのによくています。

このトピックでは、Android SDKにまれている`org.json`パッケージのについてします。

Examples

なJSONオブジェクトをする

のJSONをえてみましょう

```
{
  "title": "test",
  "content": "Hello World!!!",
  "year": 2016,
  "names" : [
        "Hannah",
        "David",
        "Steve"
   ]
}
```

このJSONオブジェクトは、のコードをしてできます。

```
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());
}
```

JSONObjectのにネストされた**JSONArray**をしたのをにします。

```
{
    "books":[
      {
        "title":"Android JSON Parsing",
        "times_sold":186
      }
    ]
}
```

これは、のコードでできます。

```
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");
```

シンプルなJSONオブジェクトの

のコンストラクタをしてJSONObjectをし、なるタイプでできるようにオーバーロードされたput()メソッドをしてフィールドをします。

```
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);
}
```

のJSONはのようになりJSON。

```
{
   "name":"test",
   "content":"Hello World!!!1",
   "year":2016,
   "value":3.23,
   "member":true,
   "null_value":null
}
```

JSONObjectにJSONArrayをする

```
// 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();
```

のJSONはのようになります。

```
{
   "the_array":[
      "ASDF",
      "QWERTY"
   ]
}
```

nullをつJSONをします。

のようにが`null` JSONをするがある

```
{
   "name":null
}
```

に、なJSONObject.NULLをするがあります。

```
jsonObject.put("name", JSONObject.NULL);
```

jsonのにnull-stringをする

```
{
    "some_string": null,
    "ather_string": "something"
}
```

このでする

```
JSONObject json = new JSONObject(jsonStr);
String someString = json.optString("some_string");
```

々はをっています

```
someString = "null";
```

したがって、このをするがあります。

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

それから、

```
JSONObject json = new JSONObject(jsonStr);
String someString = optNullableString(json, "some_string");
String someString2 = optNullableString(json, "some_string", "");
```

どおりのをあげることができます

```
someString = null; //not "null"
someString2 = "";
```

JsonReaderをしてストリームからJSONをみむ

JsonReaderはJSONでエンコードされたをトークンのストリームとしてみみます。

```
   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);
  }
```

ネストされたJSONオブジェクトをする

ネストされたJSONオブジェクトをするには、のJSONオブジェクトをするだけです。

```
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";
}
```

`mainObject`とばれるキーがま`claim`で`requestObject`としてを。

JSONレスポンスのキーの

これは、のキーをするのです。ここで`A`と`B`はキーです。でもかまいません

```
{
  "response": [
    {
      "A": [
        {
          "name": "Tango"
        },
        {
          "name": "Ping"
        }
      ],
      "B": [
        {
          "name": "Jon"
        },
        {
          "name": "Mark"
        }
      ]
    }
  ]
}
```

Javaコード

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

JSONでのフィールドのをする

JSONExceptionは、コードにJSONExceptionがするのをけるために、フィールドがJSONにするかどうかをするとです。

これをするには、ののように、 JSONObject#has(String)またはメソッドをします。

サンプルJSON

```
{
   "name":"James"
}
```

Javaコード

```
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".
```

JSONのをする

サンプルjsonをする

```
 {
 "student":{"name":"Rahul", "lastname":"sharma"},
 "marks":{"maths":"88"}
 }
```

jsonのをするには、をりててするがあります。

```
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);
}
```

された

```
 {
 "student":{"name":"Kumar", "lastname":"sharma"},
 "marks":{"maths":"88"}
 }
```

オンラインでJSON with Android with org.jsonをむ https://riptutorial.com/ja/android/topic/106/json-with-android-with-org-json

76: JUnitによるAndroidのユニットテスト

- Vogella JUnitによるユニットテスト
- Junit java2novice.com
- アサートクラス junit.org
- JUnit Api tutorialspoint.com
- Anroid Testing Medium.comの

Examples

ローカルユニットテストの

テストクラスは`/src/test/<pkg_name>/`

テストクラスの

```
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
    }
}
```

す

```
public class ExampleUnitTest {
  ...
}
```

テストクラスでは、いくつかのテストクラスをし、それらをテストパッケージにすることができます。

```
@Test
public void addition_isCorrect() {
   ...
}
```

テストメソッド、いくつかのテストメソッドは、テストクラスですることができます。

`@Test`してください。

Testアノテーションは、JUnitに、それがされているpublic voidメソッドをテストケー

スとしてできることをJUnitにえます。

`@Before` 、 `@After`などのなアノテーションがいくつかあります。 このページは、するのにしています。

```
assertEquals(9, c); // This test passes
assertEquals(10, c); //Test fails
```

これらのメソッドは、 `Assert`クラスのメンバです。いくつかののなである`assertFalse()` `assertNotNull()` `assertTrue`などはここでだ。

JUnitテストのアノテーション

@Test TestアノテーションはJUnitに、それがされているpublic voidメソッドをテストケースとしてできることをJUnitにえます。メソッドをするために、JUnitはにクラスのしいインスタンスをし、きメソッドをびします。

@Beforeテストをくとき、いくつかのテストがされるにされたのオブジェクトをとすることがです。 `@Before` public voidメソッドにを`@Before`と、そのメソッドはTestメソッドのにされます。

@After Beforeメソッドでリソースをりてるは、テストのにリソースをするがあります。 `@After` public voidメソッドにを`@After`と、そのメソッドはTestメソッドのにされます。 `@After`メソッドまたはTestメソッドがをスローしたでも、すべての`@After`メソッドがされることがされています。

ヒントに**Android Studio**でテストクラスをする

- テストクラスをするクラスのにカーソルをきます。
- Alt + EnterWindowsをします。
- Create Testをし、Returnをします。
- テストメソッドをするメソッドをし、[OK]をクリックします。
- テストクラスをするディレクトリをします。
- あなたはです、これはあなたののテストです。

ヒントに**Android Studio**でテストをする

- パッケージをクリックしてテストします。
- テストをしてください...
- パッケージのすべてのテストがにされます。

Androidコンポネックスからビジネスロジックへの

ローカルJVMテストのくのは、アプリケーションのにあります。 Androidコンポーネントからビジネスロジックをりすことができるようなでするがあります。ここでは、 Model-View-Presenter

パターンをしたのをします 。これをするには、ユーザーとパスワードのみをするなサインアップをします。たちのAndroidアプリは、ユーザがするユーザがではなく、パスワードがなくとも8でなくとも1のであることをするがあります。ユーザー/パスワードがなは、サインアップAPIコールをします。それのは、エラーメッセージがされます。

ビジネスロジックが**Android**コンポーネントとにされている。

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

ビジネスロジックが**Android**コンポーネントからりされている。

ここでは、さまざまなクラスのさまざまなをするのクラスLoginContractをします。

```
public interface LoginContract {
    public interface View {
        performSignUpApiCall(String username, String password);
        displayInvalidCredentialsErrorMessage();
    }
    public interface Presenter {
        void validateUserCredentials(String username, String password);
    }
}
```

たちのLoginActivityは、ユーザーのサインアップフォームビジネスロジックのをらなくてはならないというをいて、ほとんどじです。 LoginActivityは、しいLoginPresenterをしてをします。

```
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);
    }
```

```
    ...
}
```

ビジネスロジックはしいLoginPresenterクラスにします。

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

これで、しいLoginPresenterクラスにしてローカルJVMユニットテストをできます。

```
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);
    }
}
```

ごのように、LoginActivityからビジネスロジックをし、それをLoginPresenter POJOにしたとき。ビジネスロジックにしてロ―カルJVMテストをできるようになりました。

たちが1つのをつクラスごとにすること、クラスをすることなど、ア―キテクチャ―のにはにもさまざまながあることにしてください。これはがこれをするのですMVPスタイルによるデカップリング。 MVPはこのをするの1つにぎませんが、 MVVMのようなのがあります。あなたはあなたのためにくのシステムをぶだけです。

JUnitをいめる

セットアップ

JUnitをしてAndroidプロジェクトのユニットテストをするには、プロジェクトにJUnitをするがあります。ユニットテストのソ―スコ―ドをむテストソ―スセットをするがあります。 Android Studioでされたプロジェクトには、すでにJUnitのとテストソ―スセットがまれていることがよくあります

あなたのモジュ―ルにのをします`build.gradle`のファイル`Closure`

```
testCompile 'junit:junit:4.12'
```

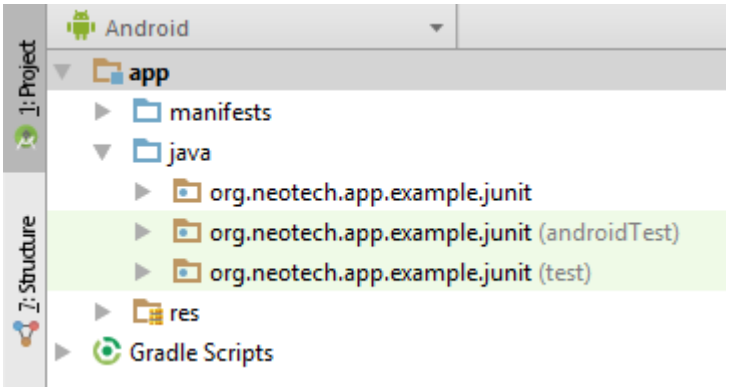
JUnitテストクラスは、 `test`というのなソ―スセットにあり`test` 。このソ―スセットがしないは、でしいフォルダをするがあります。デフォルトのAndroid StudioGradleベ―スプロジェクトのフォルダはのようになります。

```
<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)
```

プロジェクトに`/app/src/test`フォルダがないは、でするがあります。 `test`フォルダには、 `java`フォルダもですしないはしてください。 `test`ソースセットのjavaフォルダには、 `main`ソースセットとじパッケージがまれているがあります。

しくしたAndroid StudioのAndroidビューで、プロジェクトのはのようになります。



*`androidTest`*ソースセットをずしもつはありませんが、このソースセットは*Android*スタジオでされたプロジェクトでつかることがく、にここにまれています。

テストの

1. **`test`**ソースセットにしいクラスをします。
   プロジェクトビューでテストソースセットをクリックし、 `New` > `Java class`します。
   もよくされるパターンは、 `Test`するテストのクラスをすることです。だから、`StringUtilities`なり`StringUtilitiesTest`。

2. **`@RunWith`**アノテーションをする
   `@RunWith`アノテーションは、テストクラスでするテストをJUnitにさせるためにです。デフォルトのJUnitランナーJUnit 4は`BlockJUnit4ClassRunner`が、これをするわりに、デフォルトのJUnitランナーのである`JUnit4`をするがです。

   ```
   @RunWith(JUnit4.class)
   public class StringUtilitiesTest {

   }
   ```

3. テストをする
   テストは、になるテストであり、ほとんどの、するとしないはずです。いえれば、をスロー
   するべきではありません。テストメソッドのでは、ほとんどの、のがたされているかどうか
   をするアサーションがあります。アサーションがした、それはメソッド/テストをさせるを
   スローします。テストメソッドはに`@Test`アノテーションでアノテートされます。このがな
   いと、JUnitはにテストをしません。

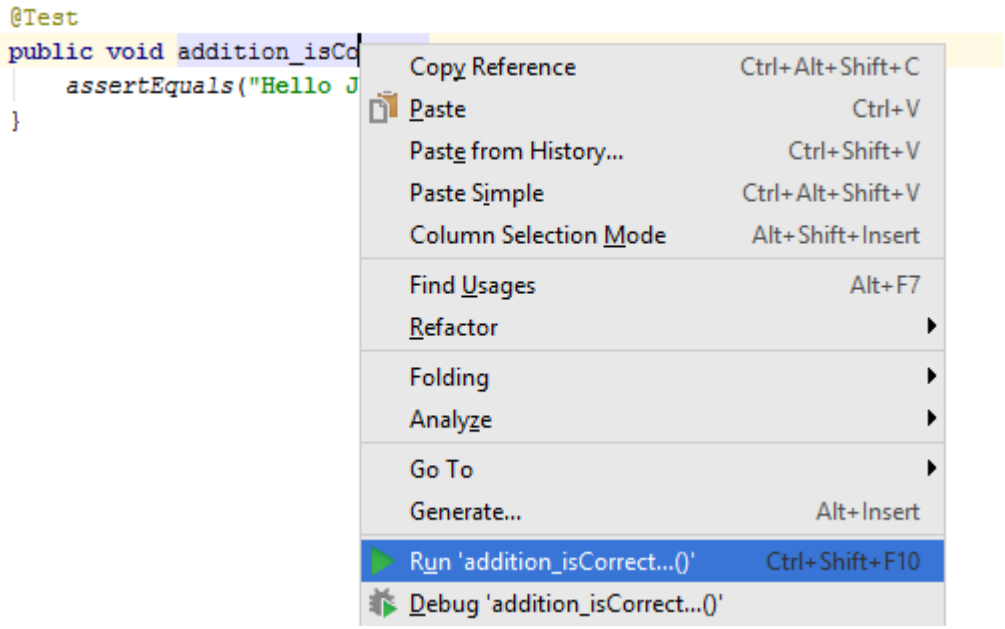
```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

    @Test
    public void addition_isCorrect() throws Exception {
        assertEquals("Hello JUnit", "Hello" + " " + "JUnit");
    }
}
```

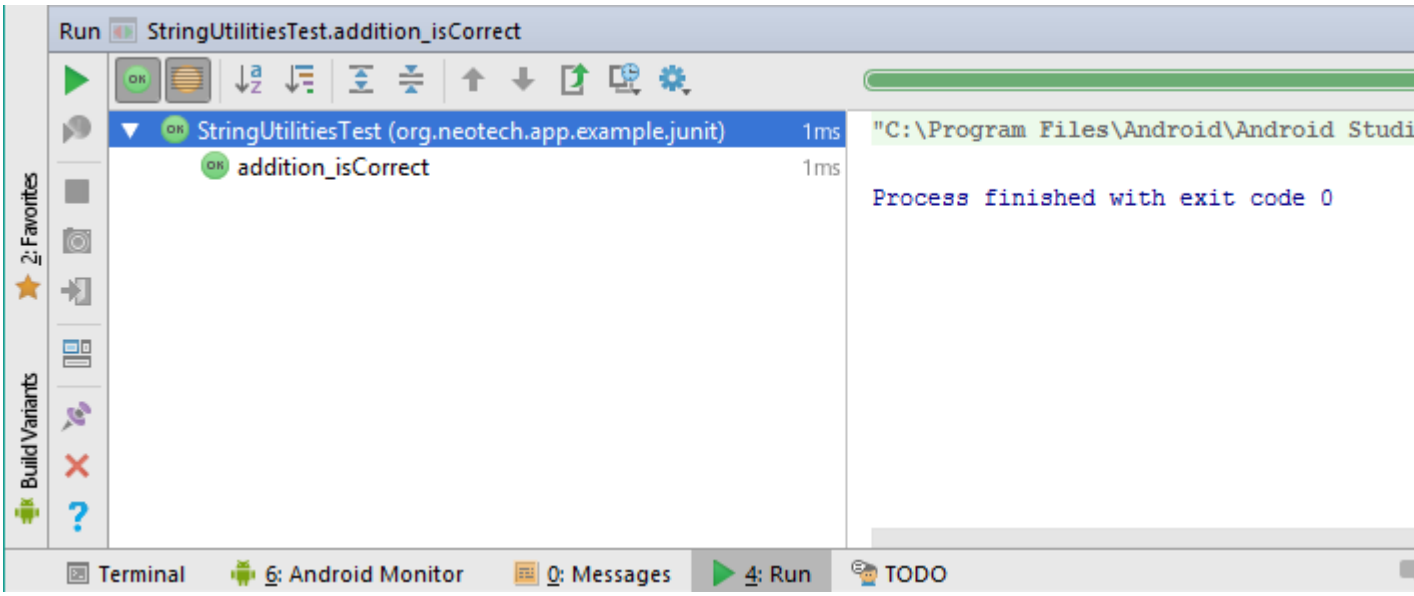
   のJavaメソッドのとはなり、テストメソッドにはアンダースコアがまれることがよくあり
   ます。

テストの

1.
   のテストメソッドをするには、メソッドをクリックし、 `Run 'addition_isCorrect()'`か、キー
   ボードショートカット`ctrl+shift+f10`ます。



   すべてがしくされていれば、JUnitはメソッドのをし、Android Studioにのインターフェース
   がされます。



2. クラス
   プロジェクトビューでクラスをクリックし、 `Run 'StringUtilitiesTest '`をクリックするか、プロジェクトビューでクラスをしたはキーボードショートカット`ctrl+shift+f10`して、のクラスでされたすべてのテストをすることもできます。

3. パッケージすべて
   プロジェクトまたはパッケージでされたすべてのテストをしたくないは、のクラスでされたすべてのテストをするのとじように、パッケージをクリックして[`Run ...`]をクリックし`Run ...`。

JUnitは、メソッドがのにしてのをスローするかどうかをテストするためにもできます。

このでは、ブールがされないかな、のメソッドがにをスローするかどうかをテストします。

```
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);
    }
}
```

`@Test`アノテーションに`expected`パラメータをすることによって、スローされるとされるをすることができます。このがしない、ユニットテストはし、がにスローされたはします。

```
@Test(expected = IllegalArgumentException.class)
public void parseBoolean_parsesInvalidFormat_throwsException(){
    StringUtilities.parseBoolean("Hello JUnit");
}
```

これはうまくいきますが、メソッドのテストケースが1つにされます。 1つのでのケースをテスト

したいがあります。このをするためによくされるは、 `try-catch`ブロックと`Assert.fail()`メソッドをすることです。

```
@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){
    }
}
```

ユニットテストので1つのケースをテストすることはいとえるもいます。

インポート

JUnitは、プリミティブにしてなくとも1つのかなりの`assertEquals`メソッドをし、Objectsにして1つのメソッドをしています。これらのメソッドは、デフォルトではびすことができず、`Assert.assertEquals`ようにびすがあります。しかし、これらのメソッドはににされるため、メソッドはなインポートをするため、メソッドがクラスのであるかのようにできます。

`assertEquals`メソッドのインポートをするには、のimportをします。

```
import static org.junit.Assert.assertEquals;
```

のインポートをして、 `assertArrayEquals` 、 `assertNotNull` 、および`assertFalse`などのすべてのアサートメソッドをにインポートすることもできます。

```
import static org.junit.Assert.*;
```

インポートをしない

```
@Test
public void addition_isCorrect(){
    Assert.assertEquals(4 , 2 + 2);
}
```

インポートの

```
@Test
public void addition_isCorrect(){
    assertEquals(4 , 2 + 2);
}
```

オンラインでJUnitによるAndroidのユニットテストをむ
https://riptutorial.com/ja/android/topic/3205/junitによるandroidのユニットテスト

77: KotlinによるAndroidプログラミング

き

Androidスタジオでkotlinをするのは、KotlinがJetBrainsによってされているため、なです。これはIntelliJ IDEAのろにあるじで、Android StudioのIDEです。そのため、にはほとんどはありません。

Kotlin Programming Languageのについては、 ドキュメンテーションをしてください。

Examples

Kotlinプラグインのインストール

まず、Kotlinプラグインをインストールするがあります。

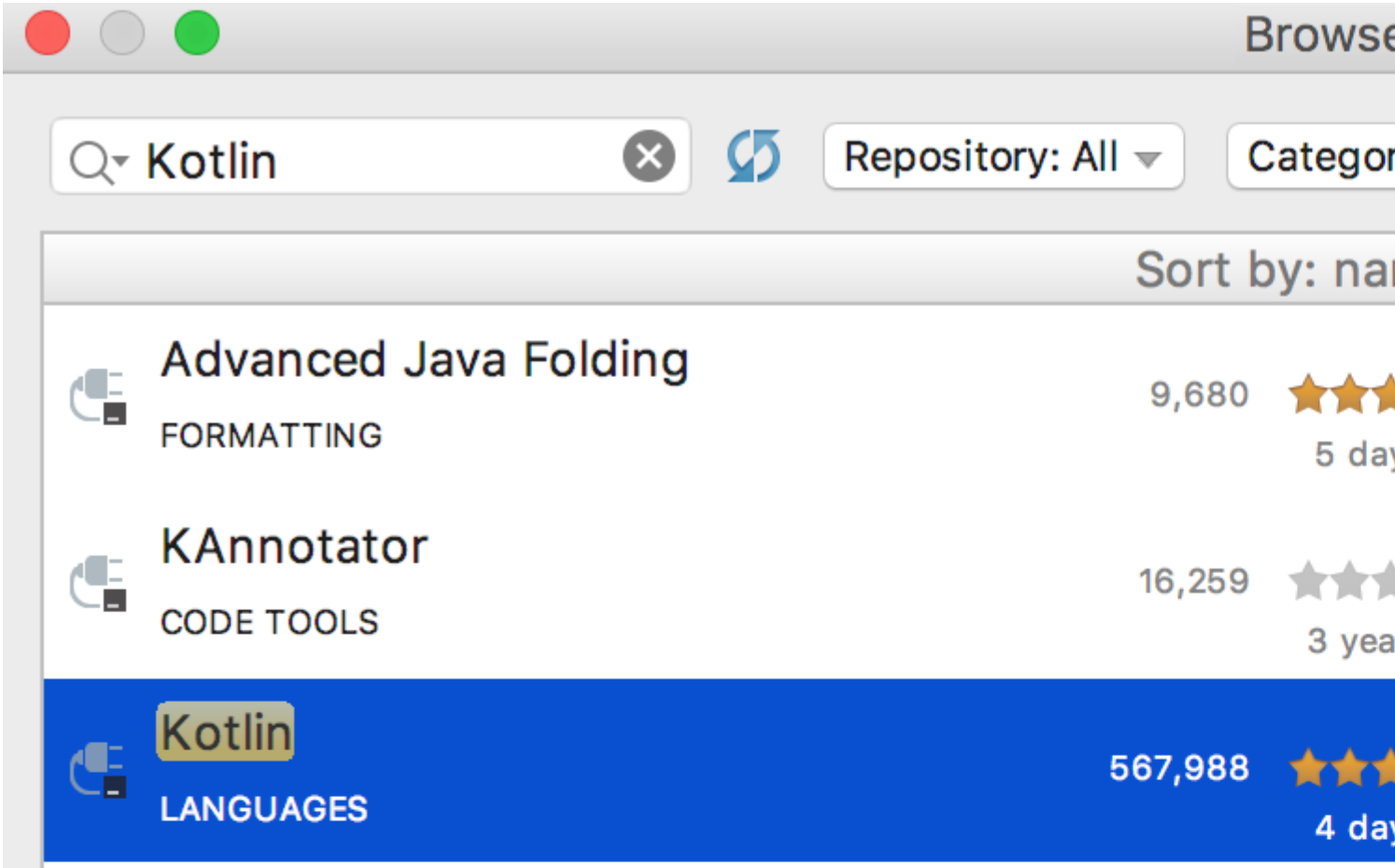
Windowsの

- `File` → `Settings` → `Plugins` → `Install JetBrains plugin`

Mac

- `Android Studio` → `Preferences` → `Plugins` → `Install JetBrains plugin`

そして、Kotlinをしてインストールします。これがしたら、IDEをするがあります。


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

したりすることができます。これをうには、のことがです。

1. ルート `kotlin-android` ファイルにをする - `kotlin-android`プラグインのをルート`build.gradle` ファイルにするがあります。

```
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. **Kotlin Android**プラグインを`apply plugin: 'kotlin-android'` - モジュール`build.gradle`ファイルに`apply plugin: 'kotlin-android'apply plugin: 'kotlin-android'`だけです。

3. を**Kotlin stdlib**にする - モジュール`build.gradle`ファイルのセクションに `'org.jetbrains.kotlin:kotlin-stdlib:1.1.2'`へのをする。

しいプロジェクトの、 `build.gradle`ファイルはのようになります。

```
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'
        }
    }
}
```

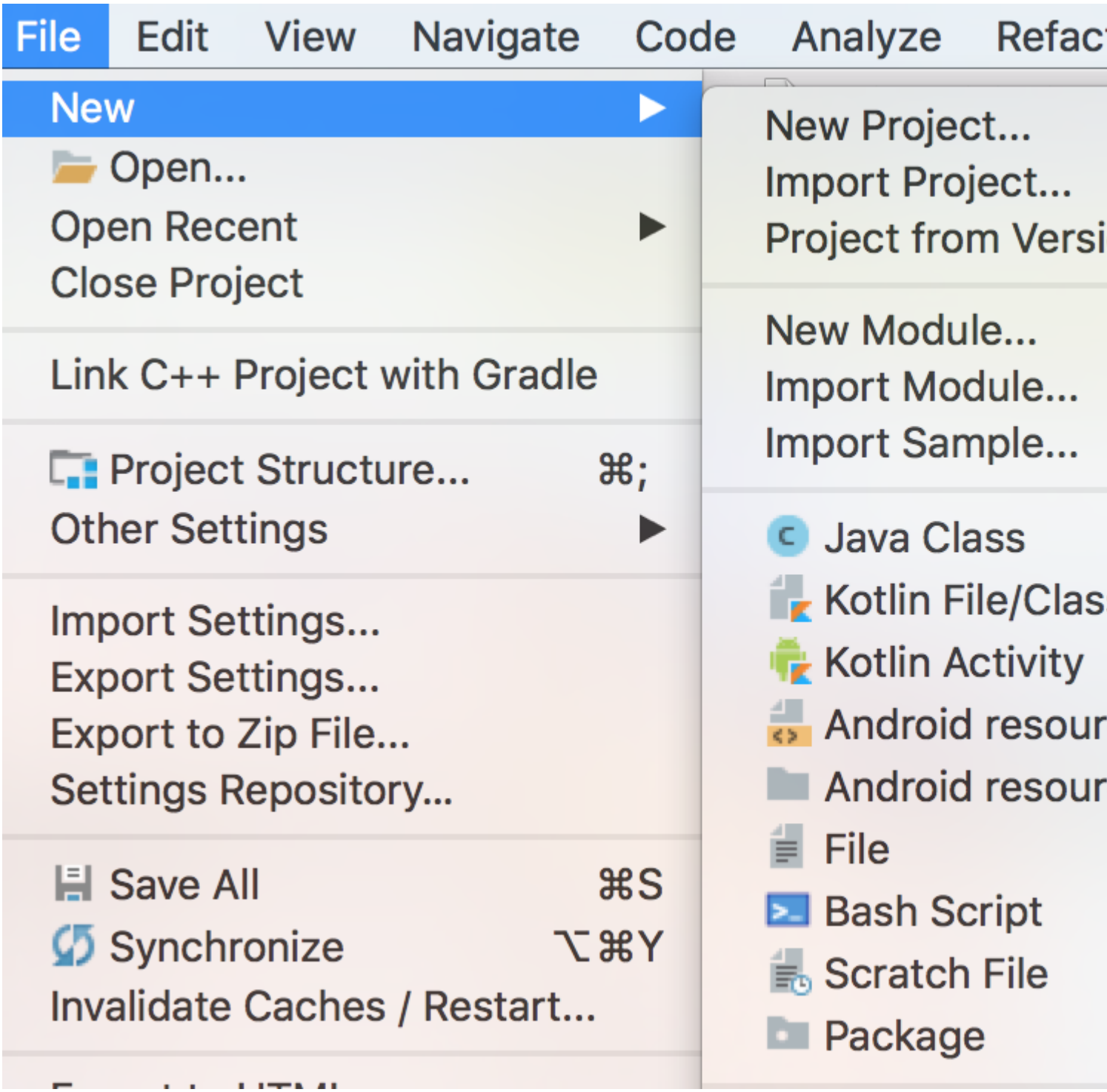
```
dependencies {
    compile 'org.jetbrains.kotlin:kotlin-stdlib:1.1.1'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    compile 'com.android.support:appcompat-v7:25.3.1'

    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })

    testCompile 'junit:junit:4.12'
}
```

しい**Kotlin**アクティビティの

1. クリック→ `File` → `New` → `Kotlin Activity`
2. アクティビティ―のタイプをします。
3. アクティビティのとそののパラメ―タをします。
4. 。



なクラスはのようになります。

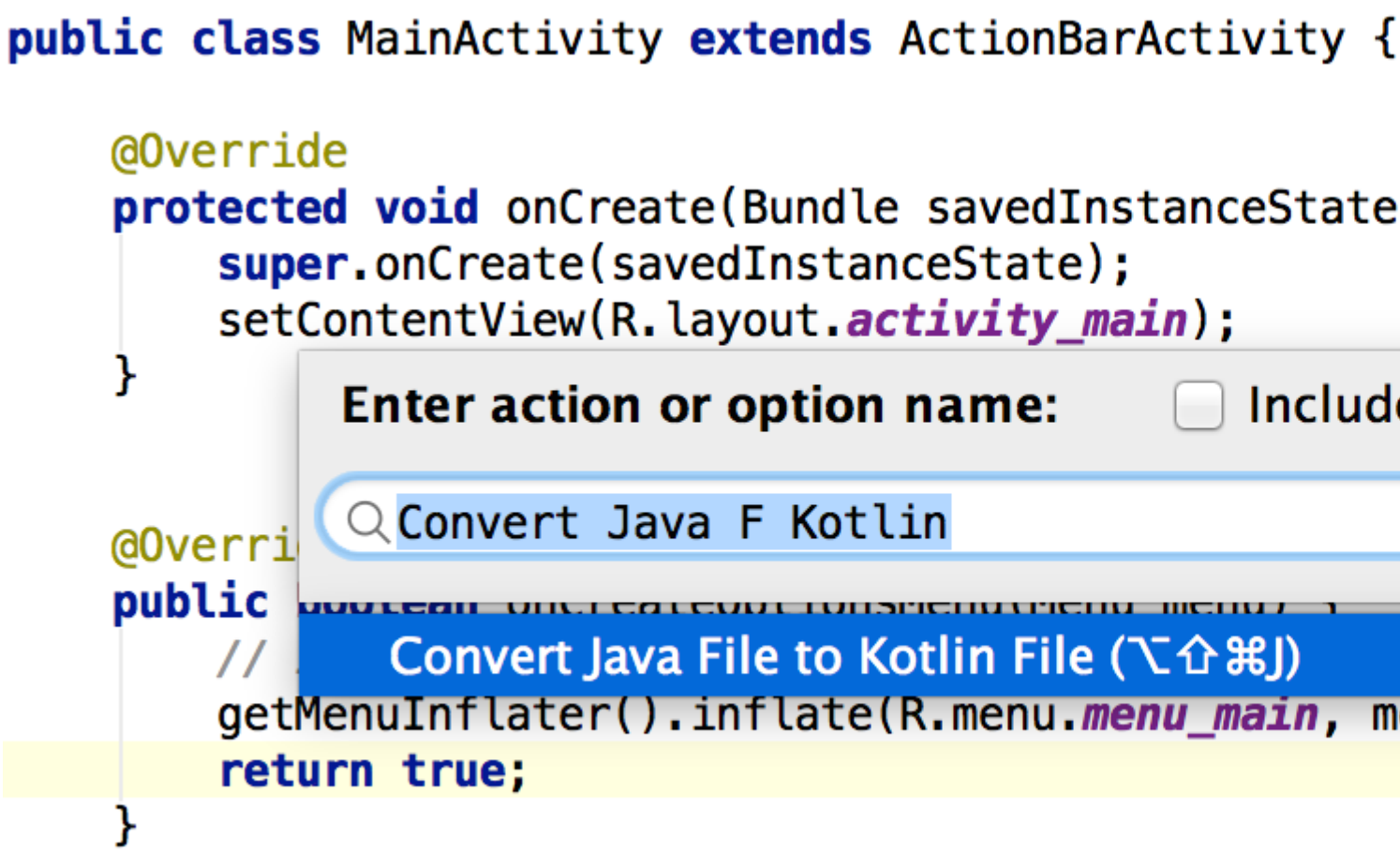
```
import android.support.v7.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

のJavaコードをKotlinにする

Android StudioKotlin Pluginは、のJavaファイルをKotlinファイルにします。 Javaファイルをし、アクションをびすJavaファイルをKotlinファイルに



しいアクティビティの

```
fun startNewActivity(){
    val intent: Intent = Intent(context, Activity::class.java)
    startActivity(intent)
}
```

インテントには、Javaのとにエクストラをできます。

```
fun startNewActivityWithIntents(){
    val intent: Intent = Intent(context, Activity::class.java)
    intent.putExtra(KEY_NAME, KEY_VALUE)
    startActivity(intent)
}
```

オンラインでKotlinによるAndroidプログラミングをむ
https://riptutorial.com/ja/android/topic/9623/kotlinによるandroidプログラミング

78: LruCache

りしのリソースがスムーズなアプリケーションのにするアプリケーションでは、Lru Cacheをするがあります。たとえば、きなサムネイル128x128のフォトギャラリーです。

それをくするとアプリにするがあるので、Lruキャッシュのサイズにはにしてください。

Lruキャッシュがもはやにたなくなった、ガベージコレクタがメモリからそれをクリーンアップできるようにするために、それへのをしないようにしてください。

のパフォーマンスをるには、なinSampleSizeをしてLruキャッシュにするなどのベストプラクティスをして、ビットマップなどのリソースをロードすることをれないでください。

Examples

キャッシュの

Larキャッシュは、メモリーにするまで、アクセスにされたすべてのリソースをします。その、されていないリソースをドロップしてしいリソースをします。

Lruキャッシュをするには、メモリーをするがあります。このはアプリケーションのと、スムーズなアプリのをするためのリソースのによってなります。たとえば、イメージギャラリのは、なメモリの1/8になります。

また、Lruキャッシュはキーバリューベースですることにもしてください。のでは、キーは`String`あり、は`Bitmap`です。

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

ビットマップリソースをキャッシュにする

キャッシュにリソースをするには、キーとリソースをするがあります。にがキャッシュにないことをしてください

```
public void addResourceToMemoryCache(String key, Bitmap resource) {
    if (memoryCache.get(key) == null)
        memoryCache.put(key, resource);
}
```

キャッシュからビットマップリソースをする

キャッシュからリソースをするには、リソースのキーをすだけですこのではString

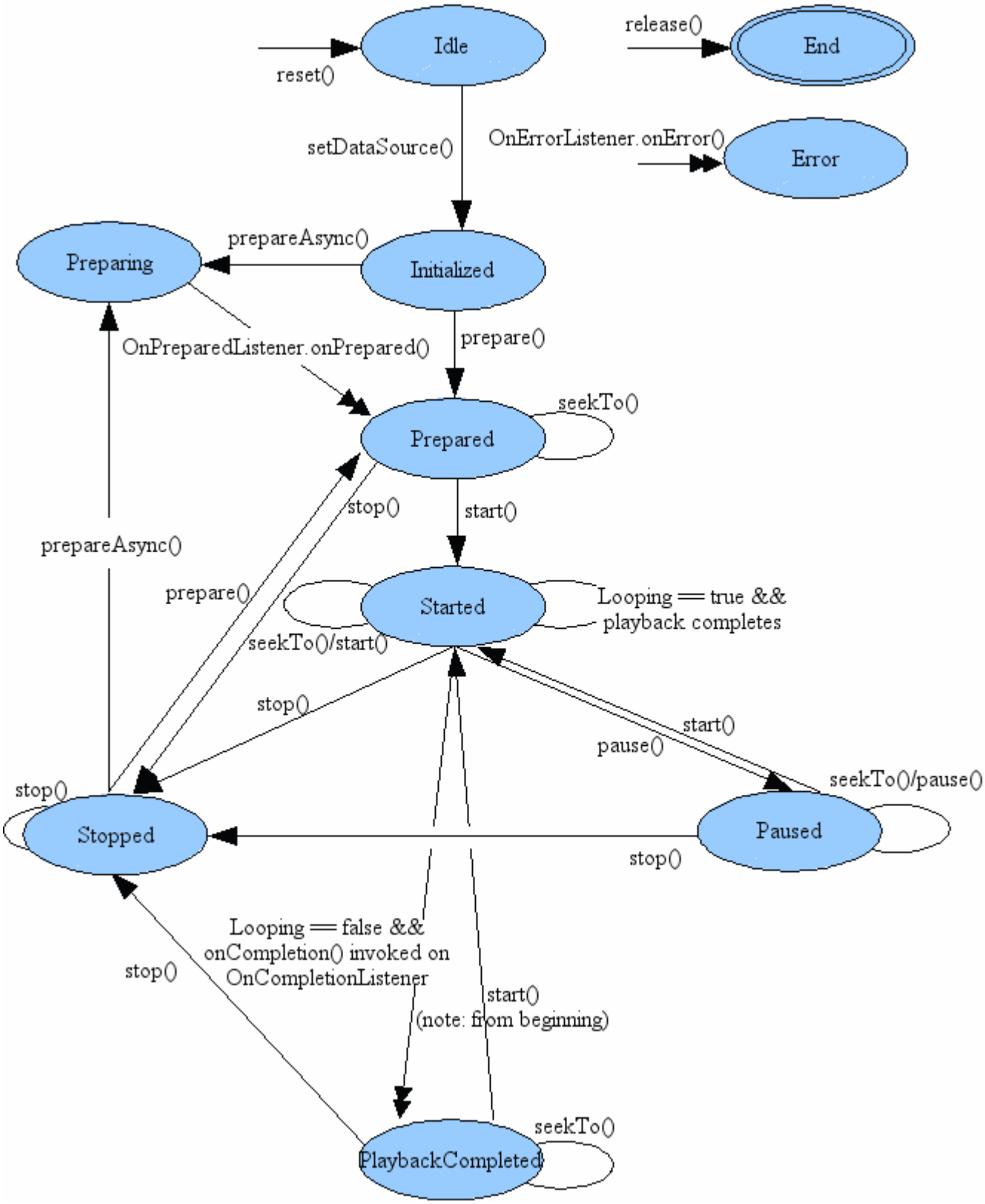
```
public Bitmap getResourceFromMemoryCache(String key) {
    memoryCache.get(key);
}
```

オンラインでLruCacheをむ https://riptutorial.com/ja/android/topic/7709/lrucache

79: MediaPlayer

- void setAudioStreamTypeint streamtype
- void setDataSourceコンテキストコンテキスト、URI uri
- void prepare
- void prepareAsync
- void start
- void stop

`MediaPlayer`は、にについています。



つまり、オーディオ/ビデオをするには、されたのがのでわれなければなりません。また、どのでどのようなをとることができるかをべています 。

MediaPlayer APIにはがなくカスタムデコーダとレンダリングロジックがされています、HTTP DASHとSmoothStreamingをしたアダプティブストリーミングのサポートがありません。これらについては、 ExoPlayerをてください 。

Examples

なと

MediaPlayerクラスは、オーディオ/ビデオファイルとストリームのをするためにできます。

MediaPlayerオブジェクトのには、の3つのタイプがあります。

1. ローカルリソースからのメディア

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.resource);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

2. ローカルURIからContentResolverから

```
Uri myUri = ....; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

3. URLから

```
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();
```

`MediaPlayer$prepare()`はブロッキングびしであり、がするまでUIをフリーズします。このをするには、 `MediaPlayer$prepareAsync()`をできます。

```
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
```

では、エラーはまたはエラーコードでされますが、リソースをするはに、アプリケーションにエラーがにされるようにするがあります。 MediaPlayerの、

```
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
        }
});
```

システムメロをする

このでは、システムのURI `RingtoneManager.TYPE_RINGTONE` をするをします。

```
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;
    }
```

このリストは、されたのによってなります。はのとおりです。

- `RingtoneManager.TYPE_RINGTONE`
- `RingtoneManager.TYPE_NOTIFICATION`
- `RingtoneManager.TYPE_ALARM`
- `RingtoneManager.TYPE_ALL = TYPE_RINGTONE | TYPE_NOTIFICATION | TYPE_ALARM`

`android.media.Ringtone`としてRingtonesをするには、すべての`Uri RingtoneManager`でするがあります。

```
android.media.Ringtone osRingtone = RingtoneManager.getRingtone(context, uri);
```

サウンドをするには、のをします。

```
public void setDataSource(Context context, Uri uri)
```

`android.media.MediaPlayer`から。 `MediaPlayer`は、 にってしてするがあります

オーディオストリームの

ストリームのさまざまなプロファイルがあります。それぞれのはそれぞれっています。

ここのすべてのは、 `AudioManager.STREAM_RING`ストリームタイプのためにかれて`AudioManager.STREAM_RING`ます。しかし、これだけではありません。なストリームのはのとおりです。

- `STREAM_ALARM`
- `STREAM_DTMF`
- `STREAM_MUSIC`
- `STREAM_NOTIFICATION`
- `STREAM_RING`
- `STREAM_SYSTEM`
- `STREAM_VOICE_CALL`

ボリュームをする

のプロファイルのボリュームをするには、のようにびします。

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int currentVolume = audioManager.getStreamVolume(AudioManager.STREAM_RING);
```

このは、ストリームのがわからないはほとんどにちません。

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int streamMaxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM_RING);
```

これらの2つののは、0 <<1をえる。

```
float volume = ((float) currentVolume) / streamMaxVolume
```

1ステップでをする

ストリームのを1ステップくするには、のようにびします。

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_RAISE, 0);
```

ストリームのを1ステップげるには、のようにびします。

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_LOWER, 0);
```

のストリームタイプをするように**MediaPlayer**をする

これをうには、 `MediaPlayer`クラスのヘルパーがあります。
ちょうど`void setAudioStreamType(int streamtype)`びして`void setAudioStreamType(int streamtype)`

```
MediaPlayer mMedia = new MediaPlayer();
mMedia.setAudioStreamType(AudioManager.STREAM_RING);
```

バッファのとをつ**Media Player**

```
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" />

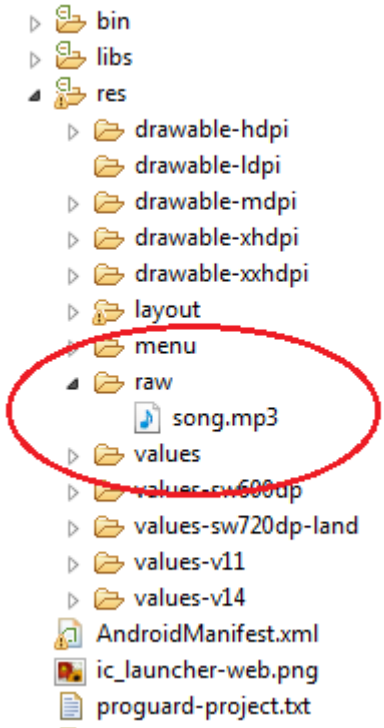
        <ProgressBar
            android:padding="8dp"
            android:progress="0"
            android:id="@+id/progress_bar"
            style="@style/Widget.AppCompat.ProgressBar.Horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center" />

    </LinearLayout>

</LinearLayout>
```

androidstudioにオーディオをインポートしてする

これは、あなたのPC /ラップトップににあるオーディオファイルをするのです。にresのにしいディレクトリをし、このようにのをけます



このフォルダにするオーディオをコピーします。これは.mp3または.wavファイルです。

ここではえばボタンをクリックしてこのサウンドをしたいは、ここでそのをします

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

これは、ボタンをクリックしたときにだけをします。すべてのボタンでをするには、このようなコードをクリックします

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

オンラインでMediaPlayerをむ https://riptutorial.com/ja/android/topic/1851/mediaplayer

80: MediaSession

- void mediaSessionCompat.setFlagsint flags
- void mediaSessionCompat.setMediaButtonReceiverPendingIntent mbr
- void mediaSessionCompat.setCallbackMediaSessionCompat.Callbackコールバック
- void mediaSessionCompat.setActiveboolean active
- MediaSessionCompat.Token mediaSessionCompat.getSessionToken
- void mediaSessionCompat.release
- void mediaSessionCompat.setPlaybackStatePlaybackStateCompat
- void mediaSessionCompat.setMetadataMediaMetadataCompat metadata

ベストプラクティスのために、 **media-compat**ライブラリをしてください。ライブラリは、メディアセッションメソッドをいプラットフォームのバージョンとのメソッドにすることで、をします。

Examples

ボタンイベントのと

このでは、 `Service`に`MediaSession`オブジェクトをします。 `MediaSession`オブジェクトは、 `Service`がされるとされます。

```
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();
    }
}
```

のメソッドは、 `MediaSession`ボタンのコールバックをインスタンスしてします。

```
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);
}
```

のメソッドは、A2DPをして `HashMap`されているメタデータをデバイスにします。

```
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);
}
```

のメソッドは、 `PlaybackState` します。 `MediaSession` がどのボタンアクションにするかをします

```
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);
}
```

オンラインでMediaSessionをむ https://riptutorial.com/ja/android/topic/6250/mediasession

81: MediaStore

Examples

デバイスののフォルダからオ—ディオ/ MP3ファイルをフェッチするか、すべてのファイルをフェッチする

まず、デバイスのストレ—ジアクセスをにするために、プロジェクトのマニフェストにのをします。

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

に、 *AudioModel.class*ファイルをし、リストのとをにするために、のモデルクラスをそのにします。

```
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;
    }
}
```

に、のをして、デバイスのフォルダからすべてのMP3ファイルをみむか、デバイスのすべてのファイルをみみます。

```
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;
}
```

のコードは、の、パス、アーティスト、アルバムをむすべてのMP3ファイルのリストをします。
については、 Media.Store.Audioのドキュメントをしてください。

のフォルダのファイルをみむには、のクエリをしますフォルダをきえるがあります。

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    MediaStore.Audio.Media.DATA + " like ? ",
    new String[]{"%yourFolderName%"}, // Put your device folder / file location here.
    null);
```

デバイスからすべてのファイルをするは、のクエリをします。

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    null,
    null,
    null);
```

ストレージアクセスをにすることをれないでください。

さて、あなたがしなければならないのは、のメソッドをびしてMP3ファイルをすることだけです

。

```
getAllAudioFromDevice(this);
```

アクティビティの

```
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;
    }
```

```
}
```

オンラインでMediaStoreをむ https://riptutorial.com/ja/android/topic/7136/mediastore

82: MultidexとDexメソッドのき

DEXは、アプリをするためにされるコンパイルされたコードをむDalvik ExecutableDEXファイルのでAndroidアプリのAPKなバイトコードファイルをします。

Dalvikファイルでは、DEXファイルでできるメソッドのが、のコードでAndroidフレームワークメソッド、ライブラリメソッド、およびメソッドをむ65,53664Kにされています。

このをするには、のDEXファイルMultidexをするようにアプリケーションビルドプロセスをするがあります。

dexとはですか

Dexは、Android Javaコードがコンパイルされるファイルとエンコーディングのです。 Androidののバージョンでは、Dalvikというマシンに`dex`バイナリをロードしてします。 Androidののバージョンでは、AndroidランタイムARTがされています。これは、 `dex`ファイルをとしてい、アプリケーションをするにそれをさらにコンパイルします。

Dexはスマートフォンののでにいファイルであり、メモリがメガバイトでされたデバイスにされています。のデザインのがもっています。

`dex`ファイルは、1つのバイナリでできるメソッドののをエンコードします。のをするファイルのは2バイトであるため、メソッドのは`0xFFFF` 、つまり65535です。アプリケーションにそのよりもくのメソッドがまれていると、コンパイルがします。

それについてをすべきか

GoogleはMultidexとばれるこのをするをしています。コンパイルおよびのコンポーネントがあります。そのがすように、コンパイルには、1つの`dex`ファイルでコードをします。に、デフォルトの`ClassLoader`これらのファイルからクラスをルックアップするをえます。

このアプローチは、よりしいデバイスではうまくいくが、いくつかのきながある。これにより、アプリケーションのがにされ、いデバイスでは`Application Not Responding`がするがあり`Application Not Responding`。

であれば、Multidexはですがけてください。

をける

64Kのメソッドをできるようにアプリをするに、アプリコードでびされたのをらすためのをする

があります。のは、dexをするのにちます。

- アプリのおよびなをする - アプリケーションにするきなライブラリを、アプリケーションにするコードのよりもくするようにします。なパターンは、いくつかのユーティリティメソッドがであったため、にきなライブラリをめることです。アプリケーションコードのをらすと、dexのをけることができます。
- **ProGuard**をしてのコードをする - ProGuardをするようにアプリケーションのProGuardをし、リリースビルドでがになっていることをします。をにすると、のコードをAPKでしないようにすることができます。

のは、のでとをとします。サードパーティライブラリをみむは、ライブラリのサイズをするがあります。たとえば、のあるJSONライブラリの2つは、JacksonとGsonです。にはまったくていますが、GsonはAndroidでのがえています。 1つのは、ジャクソンが9,000のでをするのにし、Gsonは1,900にしているということです。

がアプリケーションのサイズをするのにつツールがいくつかあります。

- dexcount-gradle-pluginは、ビルドのAPKまたはAARのメソッドのをレポートします
- dex-method-countsは、APKのメソッドのをえるコマンドラインツールです
- www.methodscount.comはアップロードするAPKのメソッドをえるWebサービスです。

Examples

MultiDexApplicationをしたMultidex

`Application`サブクラスをとしないは、このオプションをします。

これはもなオプションですが、の`Application`サブクラスをすることはできません。 `Application`サブクラスがなは、のオプションの1つにりえるがあります。

このオプションの、AndroidManifest.xmlの`application`タグの`android:name`プロパティにクラス`android.support.multidex.MultiDexApplication`をするだけ`application`。

```
<?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>
```

アプリケーションをすることによるMultidex

プロジェクトに`Application`サブクラスがなは、このオプションをします。

`application`

タグのマニフェストファイルの`android:name`プロパティをして、この`Application`サブクラスをし`Application`。

`Application`サブクラスで、 `attachBaseContext()`メソッドのオーバーライドをし、そのメソッドで`MultiDex.install()`びします。

```
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);
    }
}
```

`Application`サブクラスがAndroidManifest.xmlの`application`タグでされていることをします。

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

マルチディックスの

multidexをにするには、のものがです。

- あなたのGradleビルドをする
- `MultiDexApplication`をするか、 `Application`クラスでMultiDexをにする

グラデル

`app/build.gradle`にのをします

```
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'
}
```

アプリケーションでMultiDexをにする

に、3つのオプションのいずれかをします。

- アプリケーションをすることによるMultidex
- MultiDexApplicationをしたMultidex
- MultiDexApplicationをしたMultidex

これらのがアプリにされると、Androidビルドツールはにじてdexclasses.dexとサポート classes2.dex、classes3.dexをします。
ビルドシステムは、それらをAPKファイルにパッケージしてします。

すべてのビルドでメソッドのをカウントするDexcount Gradle Plugin

dexcountプラグインは、ビルドがしたのメソッドとクラスリソースのをカウントします。

プラグインを`app/build.gradle`します

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

buildscript {
    repositories {
        mavenCentral() // or jcenter()
    }

    dependencies {
        classpath 'com.getkeepsafe.dexcount:dexcount-gradle-plugin:0.5.5'
    }
}
```

プラグインを`app/build.gradle`ファイルにします。

```
apply plugin: 'com.getkeepsafe.dexcount'
```

プラグインによってされたデータをします

../app/build/outputs/dexcount

になのは、.htmlチャートです。

../app/build/outputs/dexcount/debugChart/index.html

MultiDexApplicationをしたMultidex

これは、 `Application`サブクラスをして`attachBaseContext()`メソッドをオーバーライドする`attachBaseContext()`とによくています。

ただし、このメソッドをすると、 `attachBaseContext()`はすでに`MultiDexApplication`スーパークラスでオーバーライドされているため、オーバーライドするはありません。

`Application`わりに`MultiDexApplication`をする

```
package com.example;

import android.support.multidex.MultiDexApplication;
import android.content.Context;

/**
 * Extended MultiDexApplication
 */
public class MyApplication extends MultiDexApplication {

     // No need to override attachBaseContext()

     //..........
}
```

アプリケーションをしているとまったくじように、このクラスをAndroidManifest.xmlにしてください

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

オンラインでMultidexとDexメソッドのをむ https://riptutorial.com/ja/android/topic/1887/multidexとdexメソッドの

83: MVPアーキテクチャ

き

このトピックでは、AndroidのMVPModel-View-Presenterアーキテクチャにさまざまなをします。

Androidアプリをするはたくさんあります。しかし、それらのすべてがテストであるとはらず、アプリケーションをテストするのがなようにコードをすることができます。テストなアーキテクチャーのなアイデアは、アプリケーションのをして、、、およびテストをいにしやすくすることです。

MVP

モデル

がなアプリケーションでは、このモデルはドメインまたはビジネスロジックへのゲートウェイにすぎません。ビューにするデータのプロバイダとしてしてください。

ビュー

、 `Activity`または`Fragment`によってされるビューには、 プレゼンターへのがまれます。ビューがうのことは、インターフェイスアクションがあるたびにPresenterからメソッドをびすことです。

プレゼンター

プレゼンターは、ビューとモデルののとしてするがあります。モデルからデータをし、ビューにフォーマットされたデータをします。しかし、なMVCとはなり、Viewとするときにがこるかをします。

* Antonio Leivaのの。

されるアプリのではありません

アプリはごとにパッケージされているがあります 。これにより、みやすさがし、アプリのがいにしてできるようにモジュールされます。アプリのは、のJavaパッケージにまれています。

Examples

Model View PresenterMVPパターンのログイン

なログインをってMVPをにてみましょう。ログインアクションには2つの`Button`があり、には2つ

の`Button`があります。メールの2つの`EditText`とパスワードのもう1つの`EditText`。

LoginFragmentビュー

```
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プレゼンター

```
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モデル

```
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));

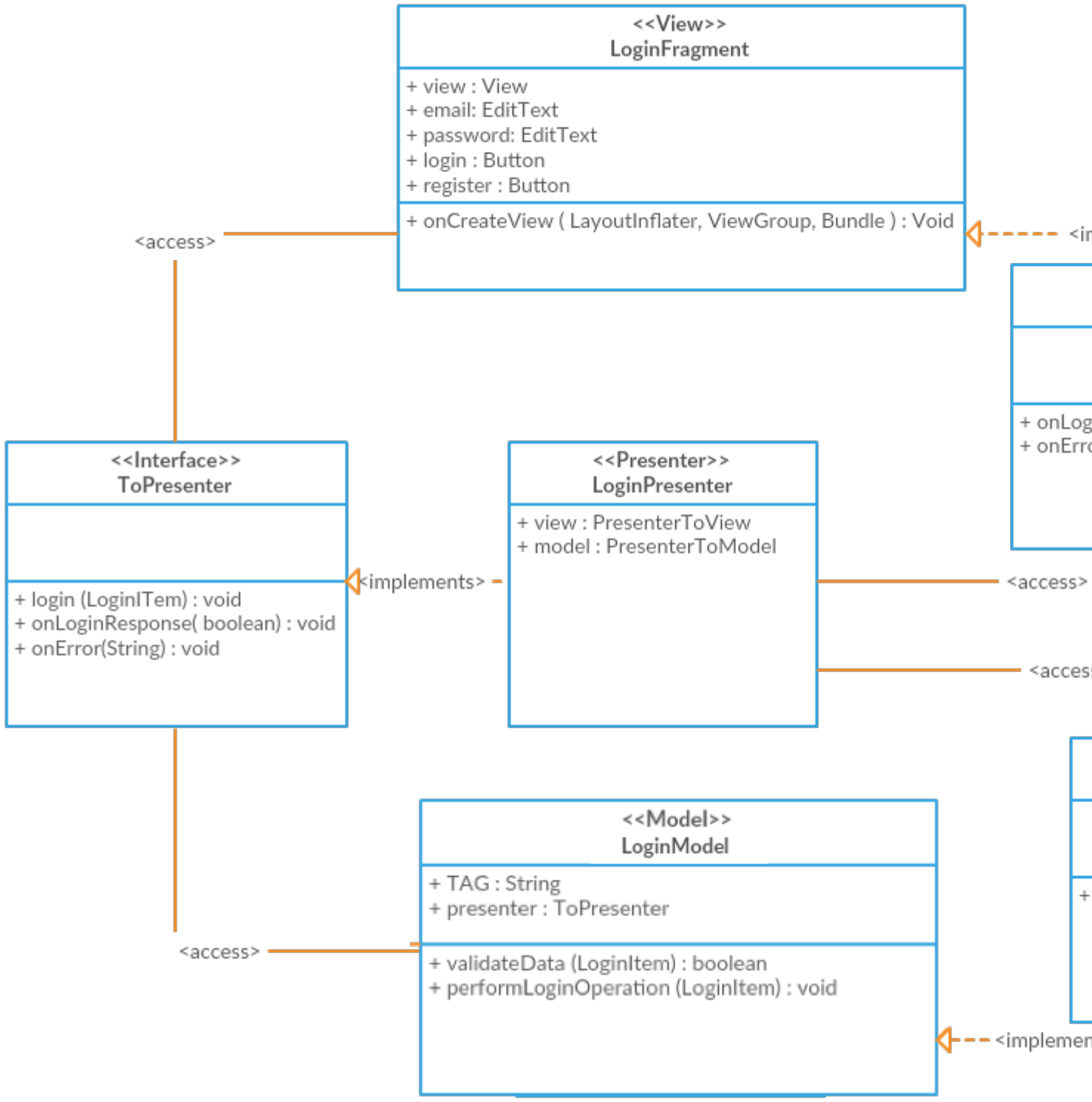
        RequestQueue queue = Volley.newRequestQueue(BaseContext.getContext());
        queue.add(request);
    }

    @Override
    public void onError(String message) {
        presenter.onError(message);
```

```
    }
}
```

クラス

アクションをクラスのでてみましょう。

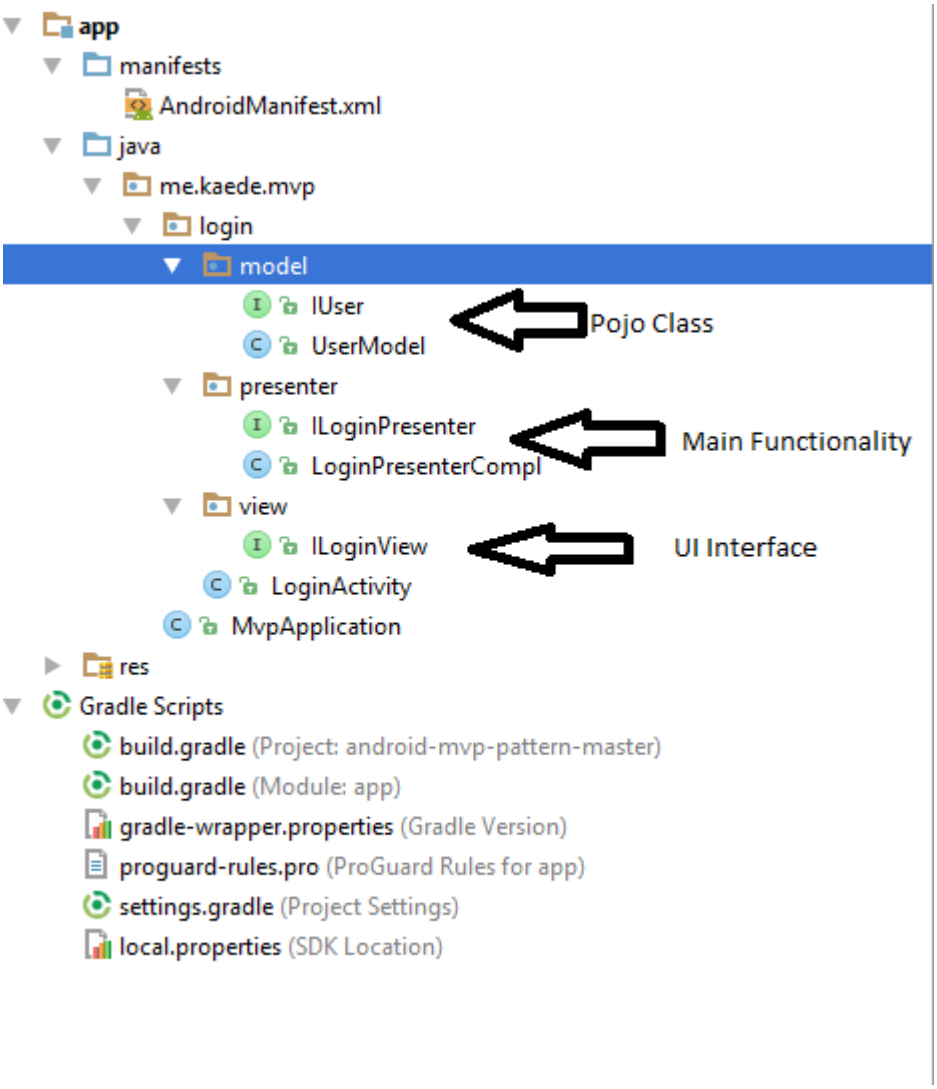


ノート

- このでは、 Volleyをネットワークにしていますが、このライブラリはMVPにはありません
- `UrlUtils`はAPIエンドポイントのすべてのリンクをむクラスです
- `ResponseErrorListener.ErrorListener`は、Volleyの`Response.ErrorListener`を`implements`する`ErrorResponse`エラーをリスンする`interface`です。これらのクラスは、このにまれていないため、ここにはまれていません

MVPにおけるなログインの

なパッケージ



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>
```

アクティビティクラス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);
    }
}
```

ILoginViewインターフェイスの

のように、ビューフォルダのPresenterからののILoginViewインタフェースをします。

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

ILoginPresenterインターフェイスの

LoginActivity Viewsとするために ILoginPresenterインタフェースをし、ログインをしてアクティビティにるためのLoginPresenterComplクラスをします。 LoginPresenterComplクラスは、ILoginPresenterインターフェイスをしてILoginPresenterます。

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");
    }
}
```

ユーザーモデルの

`LoginActivity` Pojoクラスのような`UserModel`をします。 `IUser`のための`IUser`インターフェイスをします。

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();

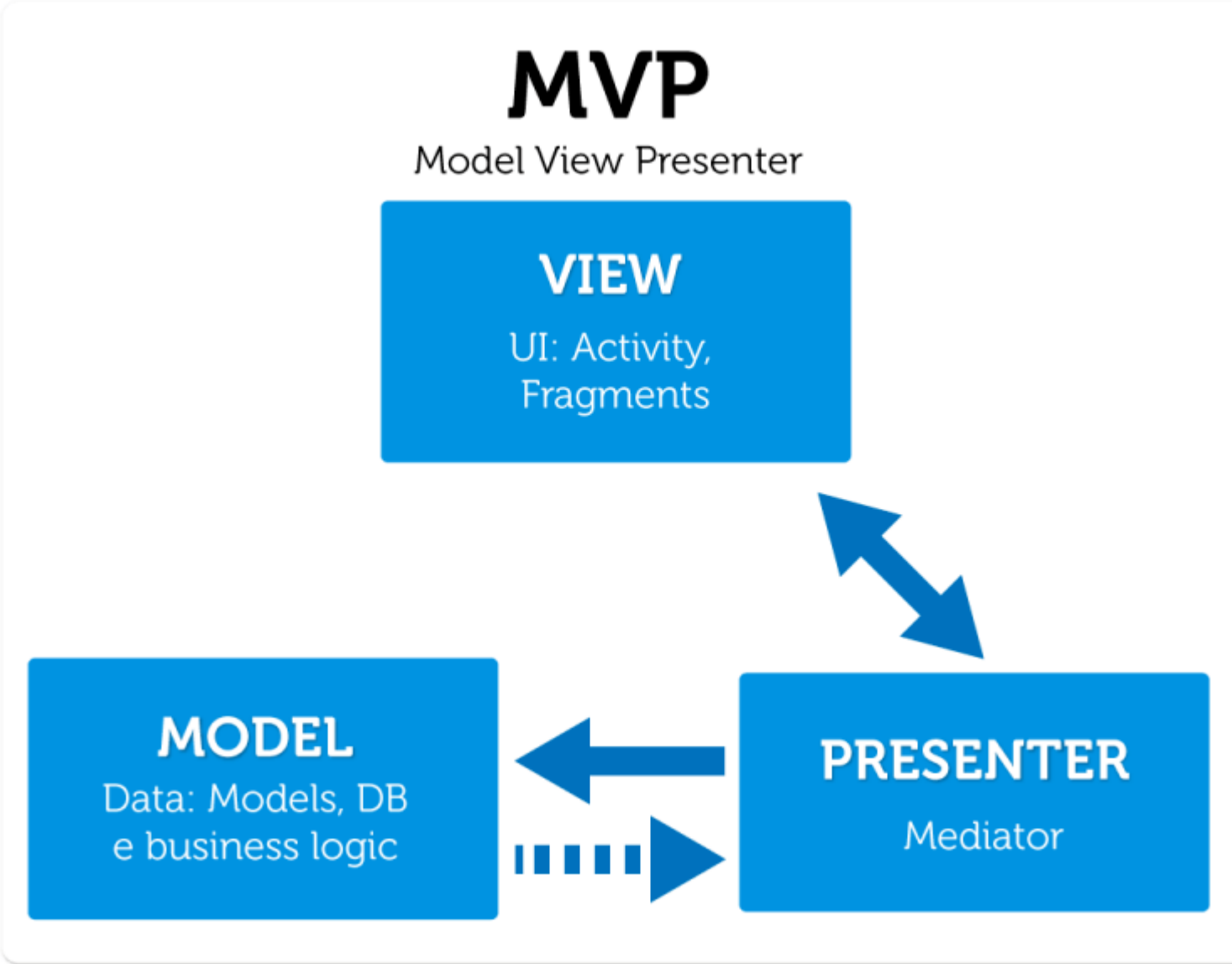
    String getPasswd();

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

MVP

モデルビューープレゼンターMVPは、モデルビューーコントローラーMVCアーキテクチャーパターンのです。にユーザーインターフェイスをするためにされ、のようながあります。

- ビューはモデルからよりれています。プレゼンターは、モデルとビューののメディエーターです。
- ユニットテストをするがです。
- に、ViewとPresenterのには11のマッピングがあり、なビューにしてのPresenterをできます。



オンラインでMVPアーキテクチャをむ https://riptutorial.com/ja/android/topic/4615/mvpアーキテクチャ

84: MVVMアーキテクチャ

データバインディングでのの

viewModelをxmlのプロパティにバインドすると、 `get`や`is`ようなのがされます。えば。 `ViewModel::getFormattedText`は、xmlのプロパティにバインドするときに`@{viewModel.formattedText}` になります。に`ViewModel::isContentVisible` - > `@{viewModel.contentVisible}` Java Bean

`ActivityMainBinding`ようなされたバインディングクラスは、Javaクラスではなく、バインディングをするXMLのについています。

カスタムバインディング

activity_main.xmlでは、 `android`ではなく、 `app` textColorをしました。ですか ViewModelによってのにされるColorResリソースIDをする`textColor`にされたカスタムセッターがするためです。

```
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);
    }

}
```

こののについては、 DataBinding LibraryCustom Settersをしてください。

って...あなたの**XML**にはがあります**!!!**

あなたはのビューにビューロジックがあるので、はXMLで`android:visibility`と`app:textColor`をMVVMコンテキストでっている/パターンしているとできます。しかし、テストのからViewModelからアンドロイドをりくことがであるとはします。

それに、に`app:textColor`はをするのですかそれは、それにけられたのへのソースポインタをするだけです。したがって、ViewModelは、らかのについてされるをします。

`android:visibility`については、メソッドがどのようにけされているかのために、ここではをにしてもいません。 `isLoadingVisible`と`isContentVisible`というため、それぞれのがどのようにされ

るべきかについていはありません。だからはにビューロジックをするよりもむしろViewModelによってえられたコマンドをしているとじています。

、は`viewModel.isLoading ? View.VISIBLE : View.GONE`をすることにするでしょう`viewModel.isLoading ? View.VISIBLE : View.GONE`はビューにとってそののをにしているため、するのがいことです。

な

のリソースは、がこのコンセプトをしようとするのをけてくれました。

- Jeremy Likness - モデルビューViewModelMVVMの C08.2010
- Shamlia Shukkur - MVVMデザインパターン C のをする 03.2013
- Frode Nilsen - Androidデータバインディングさようならプレゼンター、こんにちは ViewModel 07.2015
- Joe Birch - MVVMをしたAndroidへのアプローチ 09.2015
- Florina Muntenescu - Androidアーキテクチャパターンパート3Model-View-ViewModel 10.2016

Examples

データバインディングライブラリをしたMVVMの

MVVMののポイントは、ロジックをむレイヤーをビューレイヤーからすることです。

Androidでは、データバインディングライブラリをしてこれをけし、Androidのをにすることなく、ほとんどのロジックをユニットテストにすることができます。

このでは、のことをするかなアプリケーションのなコンポーネントをします

- にネットワークコールをし、ローディングスピナーをする
- クリックカウンタTextView、メッセージTextView、およびカウンタをインクリメントするボタンをむビューをする
- ボタンをクリックしてカウンタをクリックし、カウンタがあるにするとカウンタのとメッセージテキストをする

ビューレイヤーからめましょう

`activity_main.xml`

DataBindingのみにれていないは、おそらくそれにするために10かかるでしょう。ごのとおり、setterでするすべてのフィールドは、viewModelのにバインドされています。

`android:visibility`または`app:textColor`プロパティにするがあるは、「」セクションをしてください。

```
<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>
```

に、モデルレイヤ―。はここにいる

- アプリのをす2つのフィ―ルド
- gettersはクリックとをむ
- クリックをやす
- のをするのために

また、ここでは、クリックにする「の」をします。これはでビュ―のとメッセ―ジをするためにされます。

がどのようにユーザーにされるかについて、モデルにはがないことにすることがです。

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;
        }
    }
}
```

にViewModel。

これにより、モデルのがトリガーされ、モデルのデータがビューにされます。ここでは、モデルによってえられた `resolveCounterColor`および`resolveLabelText` にして、どのGUIがであるかをします。たとえば、ViewModelまたはビューのコードにれることなく、ののしきいがい`UnderachieverClickerModel`をにできます。

また、ViewModelはビューオブジェクトへのをしていないことにもしてください。すべてのプロパティは`@Bindable`アノテーションでバインドされ、 `notifyChange()` すべてのプロパティをするがあることを`notifyPropertyChanged(BR.propertyName)` または`notifyPropertyChanged(BR.propertyName)` このプロパティをするがあることを`notifyPropertyChanged(BR.propertyName)`いずれかのときにされます。

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;
        }
    }

}
```

アクティビティですべてをにしてください

ここでは、なすべてのをつviewModelをし、アンドロイドのコンテキストからインスタンスするがあるビューをします。

viewModelがされると、DataBindingUtilされたクラスのについては "のセクションをしてくださいをしてxmlレイアウトにバインドされます。

サブスクリプションは、メモリリークやNPEをけるために、アクティビティがまたはされたときにサブスクライブするがあるため、このレイヤでサブスクライブされます。 OrientationChangesでのviewStateのとロードもここでトリガーされます

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;
    }
}
```

アクションのすべてをるには、このサンプルプロジェクトをチェックしてください。

オンラインでMVVMアーキテクチャをむ https://riptutorial.com/ja/android/topic/7549/mvvm-アーキテクチャ-

85: NavigationView

NavigationViewは、アプリケーションのナビゲーションメニューをします。メニューのは、メニューリソースファイルによってすることができます。

`NavigationView`をするに、build.gradleファイルにデザインサポートライブラリのをするがあります。

```
 dependencies {
    compile 'com.android.support:design:24.2.0'
}
```

https://developer.android.com/reference/android/support/design/widget/NavigationView.html

https://material.google.com/patterns/navigation-drawer.html#navigation-drawer-content

Examples

NavigationViewをする

NavigationViewをするには、「」のにされているように`build.gradle`ファイルにをするだけです

に、レイアウトに`NavigationView`をします

```
<?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` きしのにされるビュー

```
<?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*ツールバーがコンテンツからりすためのレイヤー

```
<?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*デモのアクティビティののコンテンツです。ここではのレイアウトxmlをします

```
<?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>
```

メニューファイルを*res/menu/activity_main_drawer.xml*としてします。

```
<?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>
```

そしてに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();

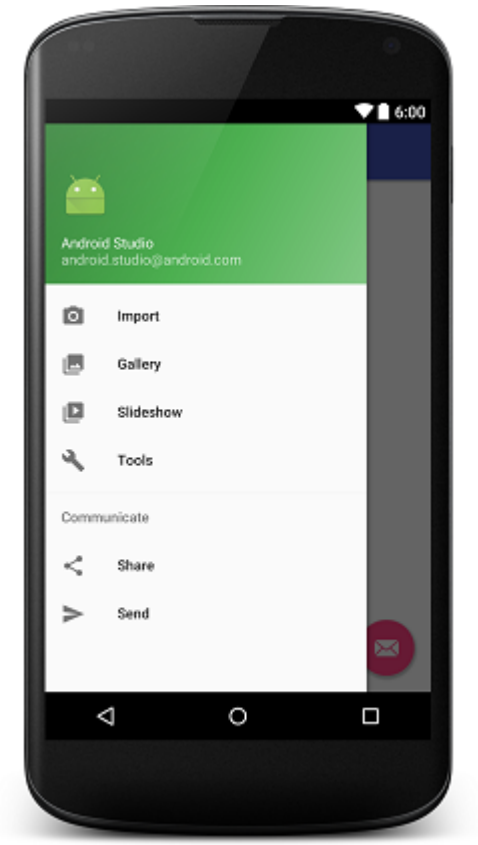
        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        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;
    }
}
```

それはのようになります



メニューにをする

グループはりでわります。メニューのにのグループがあるは、のグラフィックをします。あなたのなるグループがなる`android:id`っているにのみします。また、 `menu.xml`は、の`android:checkable="true"`は`android:checkable="true"` `menu.xml` `android:checkable="true"` 、のグループのは`android:checkableBehavior="single"`とすることをれないで`menu.xml`。

```
<?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" />

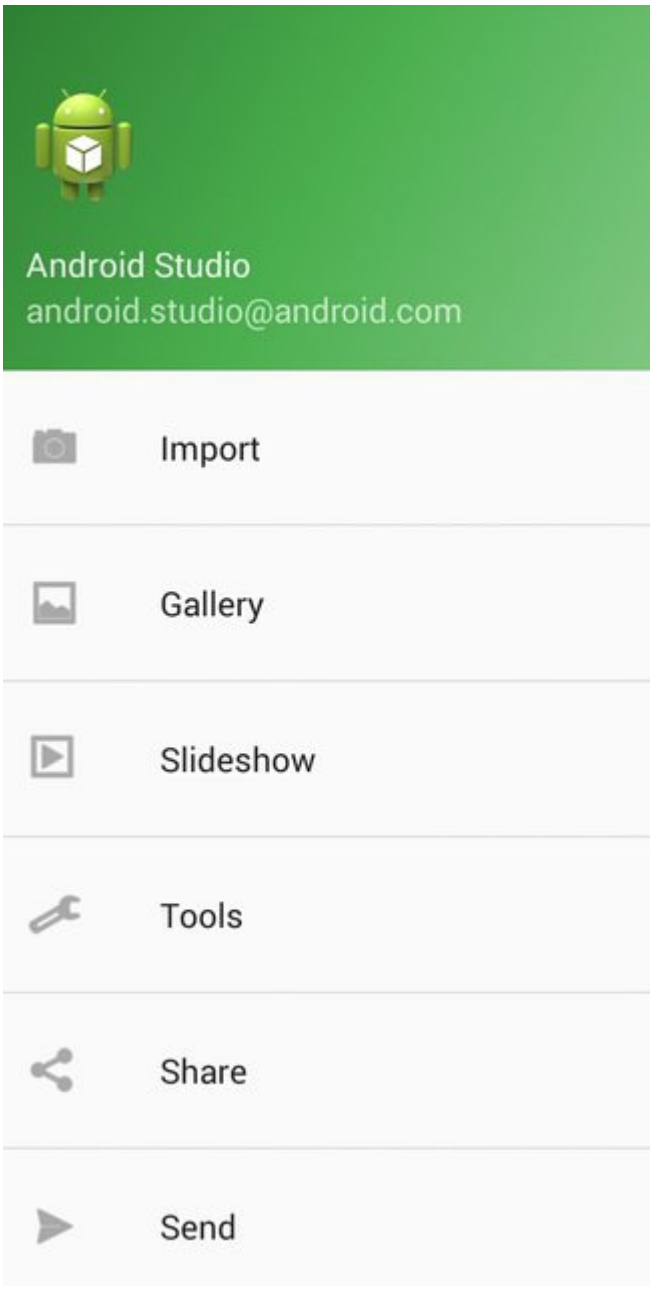
    <item
        android:id="@+id/pos_item_orders"
        android:checkable="true"
        android:title="Orders" />

       <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>
```



セパレータをメニューにする

NavigationViewのRecyclerViewにアクセスし、 ItemDecorationをします。

```
NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new DividerItemDecoration(this));
```

DividerItemDecorationのコード

```
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();

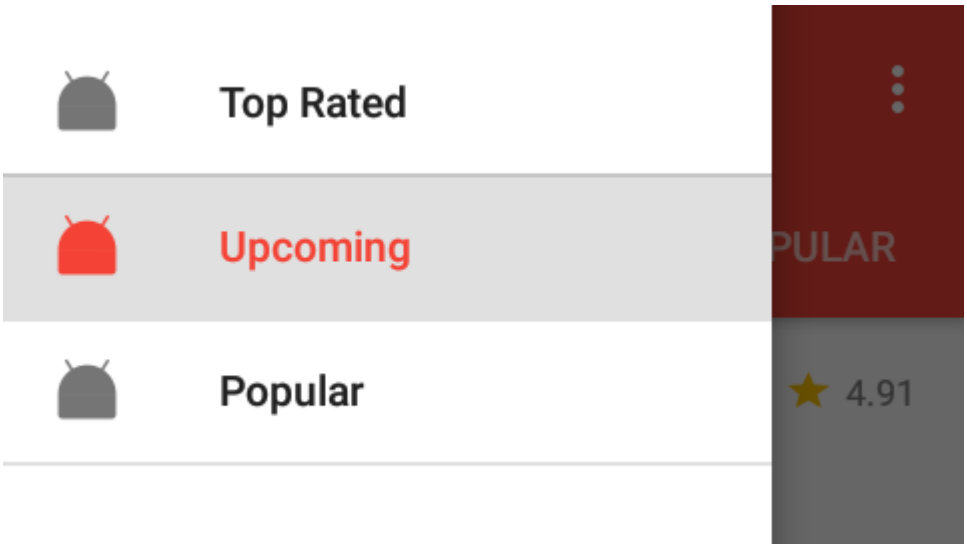
        int childCount = parent.getChildCount();
        for (int i = 1; i < childCount; i++) {
            View child = parent.getChildAt(i);

            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);
        }
    }
}
```

プレビュー



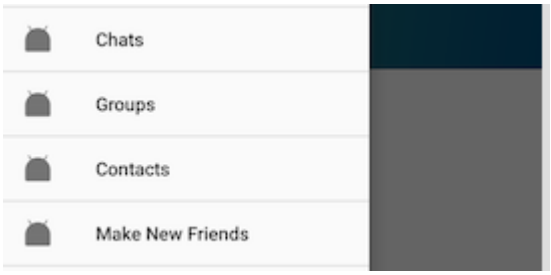
デフォルトの**DividerItemDecoration**をして、ディバイダをします。

デフォルトのDividerItemDecorationクラスをしてください

```
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation);
```

```
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new
DividerItemDecoration(context,DividerItemDecoration.VERTICAL));
```

プレビュー



オンラインでNavigationViewをむ https://riptutorial.com/ja/android/topic/97/navigationview

86: OkHttp

Examples

インターセプタのロギング

`Interceptors`は、 `OkHttp`びしをインターセプトするためにされます。するは、コールを、きえ、することができます。またはのにできます。

```
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;
  }
}
```

レスポンスのきえ

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

な

はのラップがき`OkHttp`ばれるクラスに`HttpClient`えば、このクラスでは、は、なHTTPのそれぞれのためのをっている`post` 、 `get` 、 `put`と`delete`もに、。 は、インプリメンテーションをするために、にじてにのにできるようにするためにインタフェースをみます。

```
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();
}
```

は1 `.put(body)` をいて`put` 、 `get` 、 `delete`とじですので、そのコードもするのは`.put(body)`です。いはかなりシンプルですが、いくつかの`json`ペイロードでいくつかの`url`なメソッドをびすだけで、メソッドはでしてできるとしてをします。が`json`であるとしましょう`JSONObject`にできます。

```
String response = httpClient.post(MY_URL, JSON_PAYLOAD);
JSONObject json = new JSONObject(response);
// continue to parse the response according to it's structure
```

コール

```
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());
  }
```

コール

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

フォームパラメータの

```
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());
  }
```

マルチパートリクエストの

```
 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());
  }
```

OkHttpをする

Mavenでグラブ

```
<dependency>
  <groupId>com.squareup.okhttp3</groupId>
  <artifactId>okhttp</artifactId>
  <version>3.6.0</version>
</dependency>
```

またはGradle

```
compile 'com.squareup.okhttp3:okhttp:3.6.0'
```

オンラインでOkHttpをむ https://riptutorial.com/ja/android/topic/3625/okhttp

87: Okio

Examples

ダウンロード/

のJARをダウンロードするか、Mavenでしてください

```
<dependency>
    <groupId>com.squareup.okio</groupId>
    <artifactId>okio</artifactId>
    <version>1.12.0</version>
</dependency>
```

またはGradle

```
compile 'com.squareup.okio:okio:1.12.0'
```

PNGデコーダ

PNGファイルのチャンクをデコードすることはにはOkioをしています。

```
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);
  }
}
```

バイトストリングとバッファ

バイトストリングとバッファ

Okioは、くのをなAPIにまとめる2つのタイプについてされています。

ByteStringはバイトののシーケンスです。データの、Stringがです。 ByteStringはStringのいで、バイナリデータをとしていやすくしています。このクラスはについており、16、64、およびUTF-8としてエンコードおよびデコードするをっています。

バッファはバイトです。 ArrayListとに、にバッファのサイズをするはありません。バッファはキューとしてみきします。にデータをきみ、からみみます。ポジション、リミット、またはキャパシティをするはありません。

には、 `ByteString`と`Buffer`はCPUとメモリをするためになことをします。 ByteStringとしてUTF-8をエンコードすると、そのへのがキャッシュされるため、でデコードするとができなくなります。

`Buffer`は、セグメントのリンクリストとしてされます。あるバッファからのバッファにデータをすると、データをコピーするのではなく、セグメントのがりてされます。このアプローチは、マルチスレッドプログラムにとってにちます。ネットワークとするスレッドは、コピーやセレモニーなしでワーカースレッドとデータをできます。

オンラインでOkioをむ https://riptutorial.com/ja/android/topic/9952/okio

88: OpenGL ES 2.0をいめる

き

このトピックでは、AndroidでOpenGL ES 2.0+をしてするについてします。 OpenGL ESは、コンソール、スマートフォン、アプライアンス、をむみみシステムの2Dおよび3Dアクセラレーションググラフィックのです。

Examples

GLSurfaceViewおよびOpenGL ES 2.0+の

アプリケーションでOpenGL ESをするには、これをマニフェストにするがあります

```
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
```

されたGLSurfaceViewをします

```
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);
        }
    }
}
```

`MyGLSurfaceView`をレイアウトにする

```
<com.example.app.MyGLSurfaceView
```

```
    android:id="@+id/gles_renderer"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

しいバージョンのOpenGL ESをするには、マニフェストのなインポートとの`setEGLContextClientVersion`のバージョンをする`setEGLContextClientVersion`です。

アセットファイルからのGLSL-ESシェーダのコンパイルとリンク

Assetsフォルダは、GLSL-ESシェーダファイルをするもなです。 OpenGL ESアプリケーションでそれらをするには、ににロードするがあります。このは、アセットファイルからをします。

```
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;
    }
}
```

は、スティングにされたシェーダをコンパイルするをするがあります。

```
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;
}
```

シェーダをロード、コンパイル、リンクすることができます

```
// 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);
}
```

エラーがなければ、あなたのシェーダプログラムをするができています

```
glUseProgram(shaderProgram);
```

オンラインでOpenGL ES 2.0をいめるをむ https://riptutorial.com/ja/android/topic/8662/opengl-es-2-0をいめる

89: PackageManager

Examples

アプリケーションのバージョンをする

```
public String getAppVersion() throws PackageManager.NameNotFoundException {
    PackageManager manager = getApplicationContext().getPackageManager();
    PackageInfo info = manager.getPackageInfo(
                            getApplicationContext().getPackageName(),
                            0);

    return info.versionName;
}
```

バージョンとバージョンコード

アプリケーションののビルドの`versionName`と`versionCode`をするには、Androidのパッケージマネージャーにいわせるがあります。

```
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
}
```

インストールと

あなたのアプリがインストールまたはされたをるには、Androidのパッケージマネージャーにいわせるがあります。

```
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
}
```

PackageManagerをしたユ―ティリティメソッド

ここでは、PackageManagerをっていくつかのなメソッドをつけることができます。

のメソッドは、パッケ―ジをしてアプリをするのにちます

```
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;
}
```

のメソッドは、パッケ―ジをしてアプリアイコンをするのにちますが、

```
private Drawable getAppIcon(String packageName, Context context) {
    Drawable appIcon = null;
    try {
        appIcon = context.getPackageManager().getApplicationIcon(packageName);
    } catch (PackageManager.NameNotFoundException e) {
    }

    return appIcon;
}
```

のメソッドは、インスト―ルされているアプリケ―ションのリストをするのにちます。

```
public static List<ApplicationInfo> getLaunchIntent(PackageManager packageManager) {

    List<ApplicationInfo> list =
packageManager.getInstalledApplications(PackageManager.GET_META_DATA);

    return list;
}
```

のでランチャ―アプリケ―ションもされます。

のメソッドは、ランチャ―からアプリアイコンをすのにちます。

```
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);
    }
}
```

デバイスのスイッチをオフにしてスイッチをオンにすると、このアイコンがランチャーにります。

オンラインでPackageManagerをむ https://riptutorial.com/ja/android/topic/4670/packagemanager

90: Ping ICMP

き

ICMP Pingリクエストは、Androidでpingリクエストをするためのしいプロセスをすることでできます。のは、プロセスからのpingのにできます。

Examples

のPingをします。

このでは、1のPingをみます。 `runtime.exec`メソッドびしのpingコマンドは、コマンドラインでできるなpingコマンドにできます。

```
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();
}
```

オンラインでPing ICMPをむ https://riptutorial.com/ja/android/topic/9434/ping-icmp

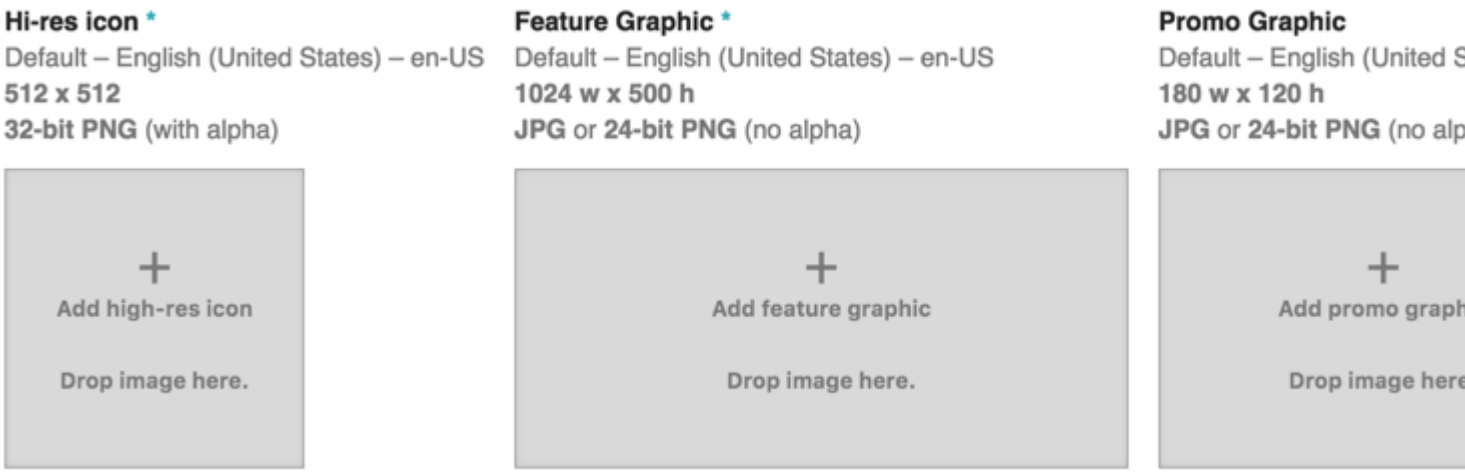
91: Playストアにする

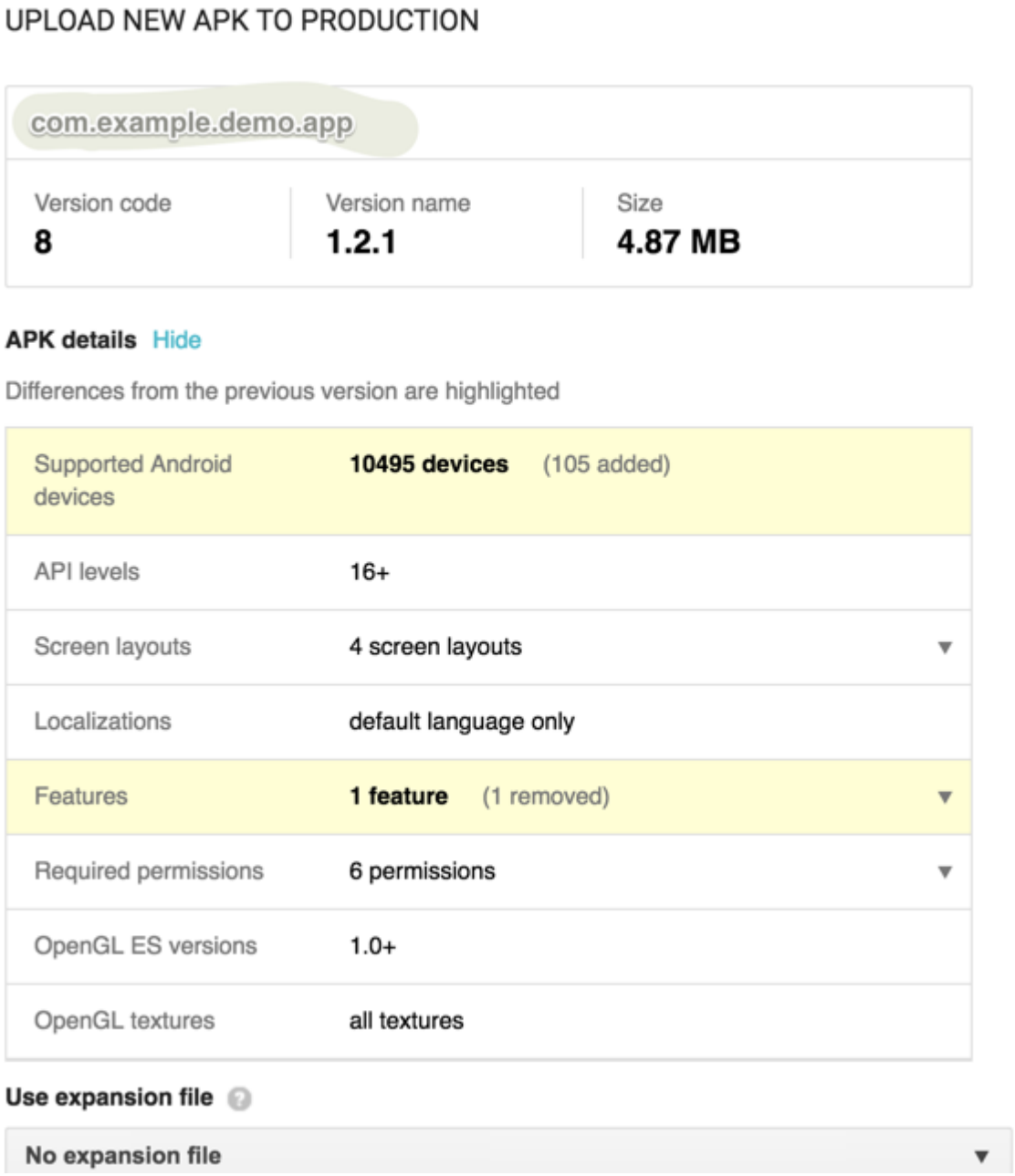
Examples

のアプリガイド

- アカウント
- すでにデバッグされていないapkをビルドしてしたapk
- アプリがないのアプリ
- Firebaseクラウドメッセージングまたはゲームサービスなし

1. https://play.google.com/apps/publish/にアクセスしてください
   1aアカウントがないはする
2. `Create new Application`を`Create new Application`ボタンをクリックします。
3. [APKを]をクリックします
4. Playストアにされるのアセットをめ、フォームのフィールドをすべてしてくださいのを
5. したら、 `Publish app`ボタンをしてください





のについては、のをしてください。

オンラインでPlayストアにするをむ https://riptutorial.com/ja/android/topic/5369/playストアにする

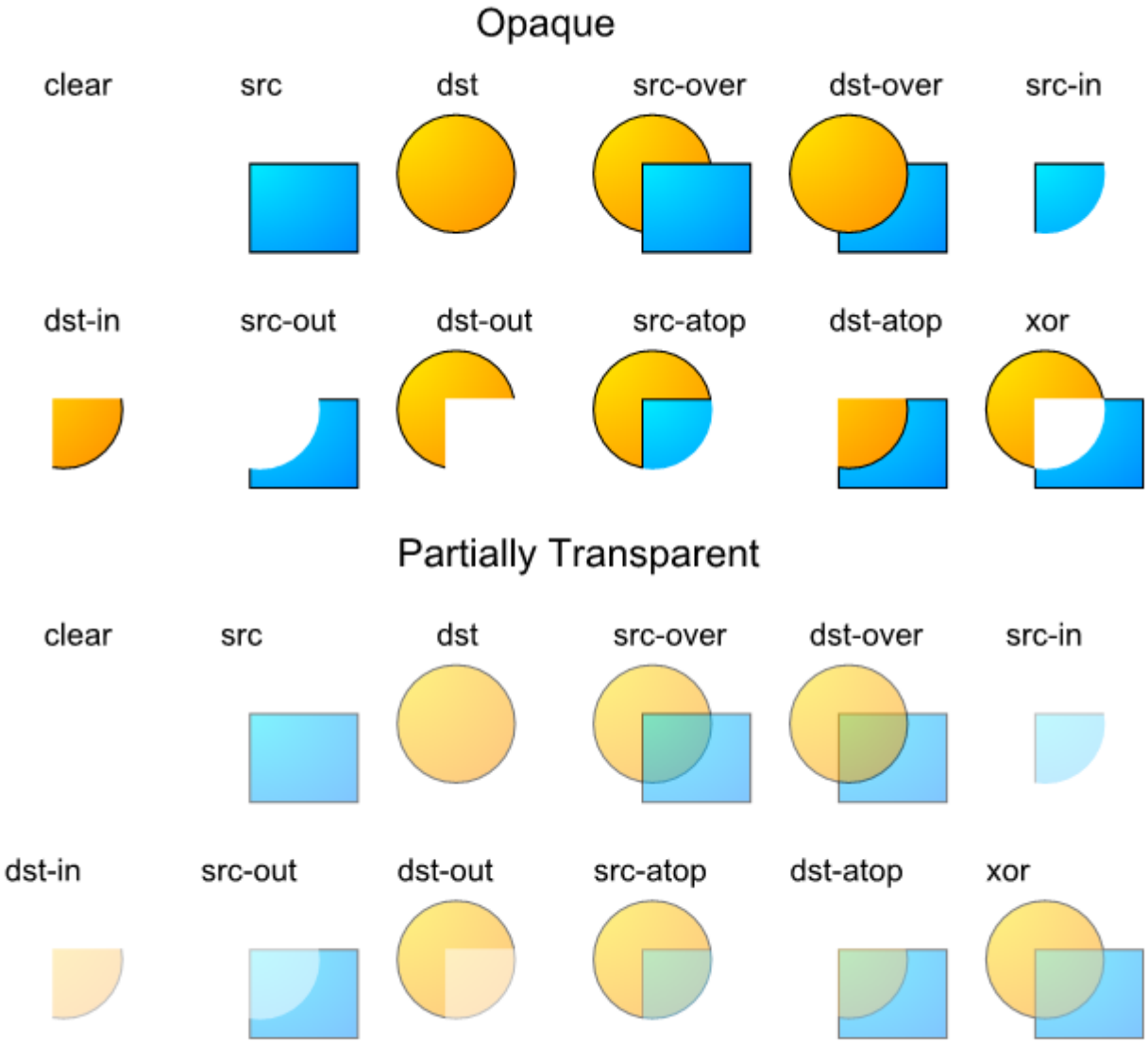
92: PorterDuffモード

き

PorterDuffは、イメージが、いにねわされた「なのボール」であるかのようにイメージをする、およびオーバーラップするをブレンドするためのスキームとしてされています

「Porter Duff」は、 Thomas PorterとTom Duffのにちなんでけられたアルファ です。

すると、このは、アルファチャネルをする2つのをり、2つののをみわせることによってをする。々なモードは、なるをもたらす。たとえば、のでは、の、のピクセルがなるモードでの、しいピクセルとみわされます。



Examples

PorterDuff ColorFilterの

PorterDuff.Modeは、 PorterDuffColorFilterをするためにされます。カラーフィルタは、ビジュアルリソースのピクセルのをします。

```
ColorFilter filter = new PorterDuffColorFilter(Color.BLUE, PorterDuff.Mode.SRC_IN);
```

のフィルタは、なピクセルをにします。

カラーフィルタはDrawableすることができます

```
drawable.setColorFilter(filter);
```

それはImageViewすることができます

```
imageView.setColorFilter(filter);
```

また、 Paintにすると、そのペイントをしてされるがフィルタによってされます。

```
paint.setColorFilter(filter);
```

PorterDuff XferModeの

Xfermode 「」モードとえるは、パイプラインのステップとしてします。 Xfermode Paintにすると、Paintでされたピクセルは、モードごとににあるピクセルにされているピクセルとされます。

```
paint.setColor(Color.BLUE);
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
```

はいいのがあります。されたシェイプは、シェイプのににするでないピクセルをくりつぶします。

PorterDuffXfermodeをしてのマスクビネットをビットマップにする

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

オンラインでPorterDuffモードをむ https://riptutorial.com/ja/android/topic/377/porterduffモード

93: ProGuard - コードのと

Examples

くわれているライブラリのいくつかのルール

、のライブラリのがまれています

1. バターナイフ
2. RxJava
3. Androidサポートライブラリ
4. Androidデザインサポートライブラリ
5.
6. GsonとJackson
7. オットー
8. Crashlitycs
9. ピカソ
10. ボレー
11. OkHttp3
12. パーセルブル

```
#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;
}
```

ビルドに**ProGuard**をにする

アプリケーションの`ProGuard`をにするには、モジュールレベルのグラデルファイルでにするがあります。 `minifyEnabled true`のを`minifyEnabled true`にするがあり`minifyEnabled true`。

また、にすることができ`shrinkResources true`リソースがされます`ProGuard`としてフラッグスを。

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

のコードは、リリースされたapkに`proguard-rules.pro` Eclipseの "proguard-project.txt"にまれるProGuardのをします。

でスタックトレースでがしたをできるようにするには、 "proguard-rules.pro"にのをめるがあります。

```
-renamesourcefileattribute SourceFile
-keepattributes SourceFile,LineNumberTable
```

EclipseでProguardをにするには、 `proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt`を「project.properties」にし`proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt`。

ビルドにトレースログおよびそののステートメントをする

のメソッドへのびしをしたい、voidをし、がないものとするとシステムの、、ななどはされませ

ん、ProGuardにビルドがしたにされます。

たとえば、これはデバッグにつデバッグ/ロギングステートメントをするのにですが、ではそれらのためのをするはありません。

```
# 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(...);
}
```

ProGuardので`-dontoptimize`をしてコードを/していないは、ステートメントがされません。 しかし、がのコードをしたくないでしょうか

2このびしはlogのびしをしますが、コードはされません。ストリングはにされたAPKにります。こののきをむ

ハッカーからコードをする

は、コードがハッカーによってされたにしにくくすることで、コードののとみなされます。

しかし、 `Log.x(..)`するとにハッカーがとするがされるとっているなら、なきがあります。

すべてのログをするには

```
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    ...etc
}
```

にログびしはされますが、はStringにはれません 。

たとえば、ログびしので、のようなのログメッセージをします`Log.d(MyTag,"Score="+score);`コンパイラは、Logびしので`+`を 'しいStringBuilder'にします。 ProGuardはこのしいオブジェクトをしません。

あなたのコンパイルされていないコードには、きき`"Score=" StringBuilder`があり、 `score`のされたバージョンがされていますこれを`b`にしたとしましょう。
、ハッカーは`b`がであるかをり、コードをします。

あなたのコードからこれらのをにするためのいは、にそれらをくフォーマッタをし、それらをするためのproguardルールをするか、 `Log`びしをのようにラップすることです。

```
    if (BuildConfig.DEBUG) {
        Log.d(TAG,".."+var);
    }
```

されたコーードがあなたでコンパイルされていることでどれくらいうまくされているかをテストしてください

1. dex2jar - apkをjar ファイルにする
2. jd - jar ファイルをデコンパイルし、GUIエディタできます

カスタムファイルでProGuardをにする

ProGuardは、がコーードをし、し、することをにします。

1こののステップは、ビルドに**proguard**をにすることです。

これは、あなたのむビルドで**'minifyEnabled'**コマンドを**true**にすることによってうことができます

22のステップは、されたビルドにしてどの**Proguard**ファイルをするかをすることです

これは、なファイルで **'proguardFiles'**をすることによってうことができます

```
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は、がむでプロゴアーードファイルをすることができます。

これはファイルをすることえば 'proguard-rules-tests.pro'とましいをえることによってうことができます。 のファイルは、プロガーードファイルのです

```
// 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
```

に、がしい.APKファイルをおよび/またはするたびに、カスタムプロアードがされ、をたします。

オンラインでProGuard - コードのとをむ https://riptutorial.com/ja/android/topic/4500/proguard----コードのと

94: ProGuardとはですか Androidでのはですか

き

Proguardは、のJavaクラスファイルツール、オプティマイザ、ツール、およびツールです。のクラス、フィールド、メソッド、およびをしてします。バイトコードをし、のをします。いなをして、りのクラス、フィールド、およびメソッドのをします。

Examples

プロガードでコードとリソースをする

APKファイルをなりさくするには、ビルドをにして、リリースビルドでのコードとリソースをするがあります。このページでは、そのと、ビルドにまたはするコードとリソースをするについてします。

Code Shrinkingは、パッケージされたアプリケーションからのクラス、フィールド、メソッド、およびをしてするProGuardでできます64KBのをするためのなツールになります。 ProGuardは、バイトコードをし、のコードをし、いのりのクラス、フィールド、メソッドをします。されたコードはAPKをリバースエンジニアリングするのをにします。これは、アプリでライセンスなどののいをするににちます。

リソースのは、GradleのAndroidプラグインでできます.Gradleは、コードライブラリのリソースをめ、パッケージされたアプリケーションからのリソースをします。のコードがされると、もはやされていないリソースもにできるように、コードとしてします。

あなたのコードをする

`ProGuard`してコードをするには、 `build.gradle`ファイルのなビルドタイプに`minifyEnabled`します。

コードをするとビルドがくなるので、であれば、デバッグビルドでしないでください。ただし、テストのなAPKでコードをできるようにすることがです。これは、するコードをにカスタマイズしていないとバグがするがあるためです。

たとえば、 `build.gradle`ファイルののスニペットでは、リリースビルドのコードをできます。

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

え`minifyEnabled`、 `proguardFiles`プロパティが`ProGuard rules`

getDefaultProguardFile 'proguard-android.txt'メソッドは、デフォルトのProGuardをAndroid SDK `tools/proguard/ folder`からし`tools/proguard/ folder`。ヒントさらにコードをするには、じにある`proguard-android-optimize.txt`ファイルをしてみてください。これにはじProGuardルールがまれていますが、バイトコードレベルメソッドおよびメソッドでをするそののをすることで、APKサイズをさらにし、よりくできます。 `proguard-rules.pro`ファイルでは、カスタムProGuardルールをできます。デフォルトでは、このファイルはモジュールのルートbuild.gradleファイルのにあります。ビルドバリアントにのProGuardルールをするには、する`productFlavor`ブロックにのproguardFilesプロパティをします。たとえば、のGradleファイルは、flavor2-rules.proをflavor2のフレーバにします。 flavor2は、リリースブロックのルールもされるため、3つのProGuardルールをすべてします。

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

オンラインでProGuardとはですか Androidでのはですかをむ
https://riptutorial.com/ja/android/topic/9205/proguardとはですか--androidでのはですか-

95: RecyclerView

き

RecyclerViewは、パフォーマンスがし、がされた、よりなリストビューです。

パラメーター

パラメーター	
アダプタ	データセットのをするビューをする、RecyclerView.Adapterのサブクラスです。
ポジション	アダプタのデータの
インデックス	getChildAtintのびしでされている、のビューのインデックスです。ポジションとの
バインディング	アダプターのにするデータをするビューをするプロセス
リサイクル	にじタイプのデータをでするために、のアダプタのデータをするためにされたビューをキャッシュにくことができます。これにより、レイアウトのインフレーションまたはをスキップしてパフォーマンスをにさせることができます
スクラップビュー	レイアウトににりされたにったビュー。ビューのRecyclerViewからにりされることなくスクラップビューがされるがあります。バインディングがないはされず、ビューがれているとされたはアダプターによってされます
ダーティービュー	されるにアダプターによってリバウンドされなければならないビュー

`RecyclerView`は、きなデータセットにられたウィンドウをするなビューです。

`RecyclerView`をするに、 `build.gradle`ファイルにサポートライブラリのをするがあります。

```
dependencies {
    // Match the version of your support library dependency
    compile 'com.android.support:recyclerview-v7:25.3.1'
}
```

サイトからバージョンのrecyclerviewをつけることができます。

そのトピック

`RecyclerView`コンポーネントについてするのトピックがあります。

- RecyclerView LayoutManager
- RecyclerViewアイテムのデコレーション
- RecyclerView onClickListener

http://developer.android.com/reference/android/support/v7/widget/RecyclerView.html

いバージョン

```
//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

RecyclerViewをする

Remarkセクションでされているようにをしてから、 `RecyclerView`をレイアウトにします。

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

レイアウトに`RecyclerView`ウィジェットをしたら、オブジェクトのハンドルをし、レイアウトマネ―ジャにし、するデ―タのアダプタをします。

```
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);
```

または、のをしてxmlからレイアウトマネ―ジャをするだけです

```
xmlns:app="http://schemas.android.com/apk/res-auto"
app:layoutManager="android.support.v7.widget.LinearLayoutManager"
```

あなたがのがすることをっている`RecyclerView`のレイアウトサイズされません`RecyclerView` 、コンポ―ネントのパフォ―マンスをするために、のコ―ドをします。 RecyclerViewのサイズがされている、RecyclerViewはのためにサイズされないので、レイアウトはびされません。それはをするだけです。がであれにするは、コ―ディネ―タ―、レイアウトなど。 `LayoutManager`と`Adapter`するでもこのメソッドをできます。

```
mRecyclerView.setHasFixedSize(true);
```

`RecyclerView`は、これらのビルトインレイアウトマネ―ジャをしてします。したがって、`RecyclerView`をして、リスト、グリッド、およびいいのグリッドをすることができます。

1. LinearLayoutManagerは、をまたはのスクロ―ルリストにします。
2. GridLayoutManagerはアイテムをグリッドにします。
3. StaggeredGridLayoutManagerは、をずらしたグリッドでします。

アイテムのスム―ズなみみ

`RecyclerView`のアイテムがネットワ―クからデ―タをみんだりはイメ―ジ、のをすると、がかかることがあります。これをけるには、の`LinearLayoutManager`をして、にされるにいくつかのをプリロ―ドすることができます。

```
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;
  }
}
```

ドラッグアンドドロップと**RecyclerView**でスワイプ

サードパーティのライブラリをしなくても、RecyclerViewでスワイプ/スワイプ/ドラッグアンドドロップをすることができます。
RecyclerViewサポートライブラリにまれている`ItemTouchHelper`クラスをして`ItemTouchHelper`。

`SimpleCallback`コールバックをして`ItemTouchHelper`をインスタンスし、サポートするにじて、`onMove(RecyclerView, ViewHolder, ViewHolder)`または`onSwiped(ViewHolder, int)`を`onSwiped(ViewHolder, int)`に`RecyclerView`にアタッチするがあります。

```
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);
```

`SimpleCallback`コンストラクタがじスワイピングを`RecyclerView`すべてのアイテムにすることにすることはです。どのようなでも、メソッド`getSwipeDirs(RecyclerView, ViewHolder)`にオーバーライドすることで、ののデフォルトスワイプをできます。

たとえば、 `RecyclerView`に`HeaderViewHolder`まれていて、らかにスワイプをしたくないとします。のように`getSwipeDirs`をオーバーライドするだけでです

```
@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);
}
```

RecyclerViewにヘッダー/フッターをする

これはサンプルのアダプター･コードです。

```
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
    }
}
}
```

ヘッダーとフッターをった`RecyclerView`のについて、 よくんでいます 。

のえがうまくいくに、 `NestedScrollView`をってリサイクルビューをってこのアプローチをうこともできます。のをってヘッダのレイアウトをすることができます

```
<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>
```

それともあなたはまた、すること`LinearLayout`あなたのにして`NestedScrollView`。

これは、 **23.2.0**よりの`RecyclerView`のみします

```
compile 'com.android.support:recyclerview-v7:23.2.0'
```

ItemViewTypeでのViewHoldersをする

RecyclerViewは、UIにされるリストにされるいくつかのタイプのビューをするがあり、ビューにはなるレイアウトxmlがです。

このでは、RecyclerView - `getItemViewType(int position)`なメソッドをして、のアダプタでなるViewHoldersをすることができます。

は、2つのViewHoldersのです。

1. リストをするためのViewHolder
2. のヘッダービューをするためのViewHolder

```
@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.
    }
}
```

RecyclerViewのアイテムをSearchViewでフィルタリングする

`RecyclerView.Adapter` `filter`メソッドを`filter`

```
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` `itemsCopy.addAll(items)`ように、アダプタのコンストラクタで`itemsCopy`がされます。

その、にびす`filter`から`OnQueryTextListener`から`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;
    }
});
```

recyclerViewのポップアップメニュー

このコードをViewHolderのにれてください

このコードでは、リスナーをitemViewオブジェクトにできる、 `recyclerview` `btnExpand` clickイベントにして、 `btnExpand` click-eventをしています。

```
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();
                }
            });

        }
    }
```

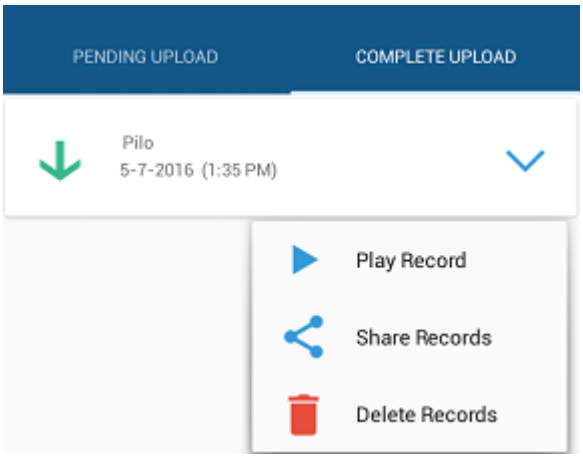
メニューにアイコンをするの

```
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) {

}
```

はのとおりです。



データをアニメーションする

`notifyDataSetChanged`をいて、いずれかの "notify"メソッドがされている、 `RecyclerView`はするアニメーションをします。これには、 `notifyItemChanged` 、 `notifyItemInserted` 、 `notifyItemMoved` 、 `notifyItemRemoved`などがまれます。

アダプターは`RecyclerView.Adapter`わりにこのクラスをするがあります。

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

`setModels`と`List`にはじ`List`をしないでください。

`models`をグローバルとしてします。 `DataModel`はダミークラスのみです。

```
private List<DataModel> models;
private YourAdapter adapter;
```

アダプタにすに`models`し`models`。 `YourAdapter`は`AnimatedRecyclerAdapter`のです。

```
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));
```

グローバル`models`をしたでこれをびします。

```
adapter.setModels(new ArrayList(models));
```

`equals`オーバーライドしない`equals` 、すべてのがによってされます。

SortedListをした

Androidは、 `RecyclerView`がされたに`SortedList`クラスをしました。このクラスは、`RecyclerView.Adapter`へのすべての 'notify'メソッドびしをして、なアニメーションをし、のをバッチできるようにして、アニメーションがジッタをこさないようにします。

```
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とデータバインディング

ここには、DataBindingレイアウトでできるのViewHolderクラスがあります。ここで、の`ViewDataBinding`クラスのインスタンスは、された`View`オブジェクトと`DataBindingUtil`ユーティリティクラスをしてされます。

```
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;
    }
}
```

このクラスをしたら、レイアウトファイルで`<layout>`して、のようにそのレイアウトのデータバインディングをにできます。

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>
```

ここにサンプルのdataModelがあります

```
public class ItemModel {
    public String itemLabel;
}
```

デフォルトでは、Android Data Bindingライブラリは、レイアウトファイルについて`ViewDataBinding`クラスをし、それをパスカルケースにし、に「Binding」をけます。このでは、それはされるだろう`MyItemBinding`レイアウトファイルのため`my_item.xml`。そのBindingクラスには、レイアウトファイルこのでは`ItemModel`のデータとしてされたオブジェクトをするためのsetterメソッドもあります。

これですべてのがしたので、のようにアダプタをできます

```
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();
    }
}
```

リサイクルビュ—のスクロ—ル。

ここでは、リサイクルビュ—でスクロ—ルをするためのコ—ドスニペットをしました。

ステップ**1**まずのようなRecycleviewアダプタでメソッドをします。

```
public abstract class ViewAllCategoryAdapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    public abstract void load();
}
```

ステップ**2**に、ViewAllCategoryAdapterクラスのonBindViewHolderおよび`getItemCount()`メソッドをオ—バ—ライドし、のように`Load()`メソッドをびします。

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, final int position) {
    if ((position >= getItemCount() - 1)) {
        load();
    }
}

@Override
public int getItemCount() {
    return YOURLIST.size();
}
```

ステップ**3**すべてのバックエンドのロジックがしました。このロジックをするです。アダプタのオブジェクトをするロ—ドメソッドをオ—バ—ライドするのはです。このメソッドは、リストのにユ—ザ—がしたときににびされます。

```
  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);
```

`load()`メソッドは、ユ—ザ—がリストのにスクロ—ルしながらにびすようになりました。

アイテムがロ—ドされるまで、またはデ—タができないときは、デフォルトビュ—をする

スクリ—ンショット



アダプタクラス

```
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>
```

はのためのIconicsライブラリとFontAwesomeをしています。これをアプリレベルのbuild.gradleファイルにします。

```
compile 'com.mikepenz:fontawesome-typeface:4.6.0.3@aar'
compile 'com.mikepenz:iconics-core:2.8.1@aar'
```

RecyclerViewアイテムにをする

これらのをにするだけです

```
RecyclerView mRecyclerView = (RecyclerView) view.findViewById(recyclerView);
mRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
mRecyclerView.addItemDecoration(new DividerItemDecoration(getActivity(),
DividerItemDecoration.VERTICAL));
```

`adapter`をし、 `.notifyDataSetChanged();`をびし`.notifyDataSetChanged();`いつものように
これはRecyclerviewのみみではありませんが、サポートライブラリにされています。だからあなたのアプリレベルbuild.gradleファイルにこれをめることをれないでください

```
compile "com.android.support:appcompat-v7:25.3.1"
compile "com.android.support:recyclerview-v7:25.3.1"
```

1つのRecyclerViewにのItemDecorationsをできます。

りのをする

itemDecorationのをするのはかなりです。

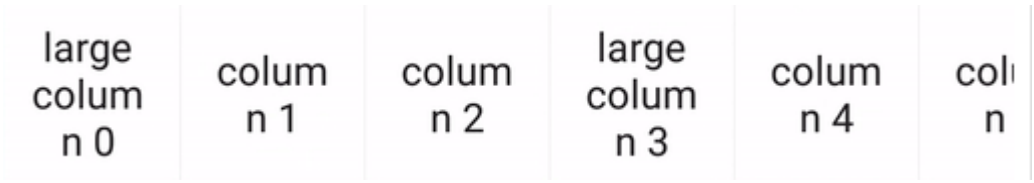
1. ステップはのとおりです。`divider.xml`にされているファイル`drawable`フォルダを

```
<?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. ステップはdrawableをする

```
    // 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);
```



オンラインでRecyclerViewをむ https://riptutorial.com/ja/android/topic/169/recyclerview

96: RecyclerView onClickListener

Examples

しい

```
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);
    }
}
```

KotlinとRxJavaの

のは、Kotlinでされ、RxJavaをしてきれいなをします。

```
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
        }
    }
}
```

いはとてもです。 RxJavaをしてのスレッドですることもです。

```
val adapter = SampleAdapter(arrayOf("Hello", "World"))
adapter.itemClickStream.subscribe { v ->
    if (v.id == R.id.textView) {
        // do something
    }
}
```

なOnLongClickとOnClickの

まず、ビュ―をします。

```
implements View.OnClickListener, View.OnLongClickListener
```

に、のようにリスナ―をします。

```
itemView.setOnClickListener(this);
itemView.setOnLongClickListener(this);
```

に、リスナ―をのようにオ―バ―ライドします。

```
@Override
public void onClick(View v) {
    onclicklistner.onItemClick(getAdapterPosition(), v);
}

@Override
public boolean onLongClick(View v) {
    onclicklistner.onItemLongClick(getAdapterPosition(), v);
    return true;
}
```

に、のコ―ドをします。

```
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);
}
```

アダプタのデモ

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

のコーードは、のコーードでびすことができます

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

アイテムクリックリスナー

アイテムのクリックリスナーおよび/またはアイテムのいクリックリスナをするには、アダプタでインターフェイスをします。

```
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;
    }

}
```

に、アダプターのインスタンスをしたに、クリックリスナーをできます。

```
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;
    }
});
```

Item Click Listenerをするの

のクリックリスナーをするもう1つのは、いくつかのメソッドをつインターフェイスをすることです。そのはクリックなビューのとしく、にすようにオーバーライドされたクリックリスナーをします。このは、クリックリスナを々のビューにでき、クリックロジックをににできるため、があります。

```
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);
        }
    }
}
```

また、アダプターのインスタンスがあるは、ビューをクリックしてリスニングするクリックリスナーをできます。

```
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クリッククリスナー

```
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);
    }
}
```

MainActivityで

```
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) {


   }
  }));
```

オンラインでRecyclerView onClickListenerをむ
https://riptutorial.com/ja/android/topic/96/recyclerview-onclicklistener

97: RecyclerViewおよびLayoutManager

Examples

スパンをつGridLayoutManager

グリッドレイアウトレイアウトマネージャをしてrecyclerviewをするときは、コンストラクタでスパンをするがあります。スパンはのをします。これはかなりぎこちなく、サイズやのきをしていません。 1つのアプローチは、さまざまなサイズののレイアウトをすることです。に、さらにもうつのダイナミックなアプローチをします。

に、のようにカスタムRecyclerViewクラスをします。

```
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);
        }
    }
```

```
}
```

このクラスは、recyclerviewにまるのをします。これをするには、のようにlayout.xmlにするがあります

```
<?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"
    />
```

columnWidthをすることにしてください。 recyclerviewは、なスペースにまるのをするためにそれをとします。

あなたのアクティビティ/フラグメントでは、recylerviewへのをし、アダプタおよびしたいアイテムやアニメーションをします。 レイアウトマネージャをしないでください

```
RecyclerView recyclerView = (RecyclerView) findViewById(R.id.auto_fit_recycler_view);
recyclerView.setAdapter(new MyAdapter());
```

MyAdapterはアダプタクラスです

あなたは、サイズにうようにスパンすなわちをするrecyclerviewをっています。に、columnsをrecyclerviewのにすることもできますデフォルトではlayout_startにえられています。AutofitRecyclerViewクラスをしすることで、これをうことができます。まず、recyclerviewにクラスをします。これは、GridLayoutManagerからするクラスになります。をにするには、にながされます。

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

に、AutofitRecyclerViewでLayoutManagerをすると、のようにCenteredGridLayoutManagerがされます。

```
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);
}
```

ですあなたはダイナミックなスパンマウント、centerlined gridlayoutmanagerベーのrecyclerviewをっています。

ソース

- Chiu-Ki ChanのSquare Islandブログ
- スタックオーバーフロー

Gridlayoutマネージャをしてrecyclerviewにヘッダビューをする

gridlayoutをつrecyclerviewにヘッダーをするには、にヘッダー·ビューがコンテンツにされるセルではなく、のであることをアダプターにえるがあります。に、レイアウトマネージャーは、のがリストの*スパンにしいスパンをつべきであることをえなければなりません。 *

のRecyclerView.Adapterクラスをし、のようにします。

```
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
    }
}
```

に、activity / fragment

```
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);
```

ヘッダーにえて、ヘッダーのわりにじフッターをすることもできます。

Chiu-Ki ChanのSquare Islandブログ

LinearLayoutManagerをしたシンプルなリスト

このでは、カスタム`Place`オブジェクトの`ArrayList`をデータセットとしてして、とのあるのリストをします。

アクティビティのレイアウト

アクティビティ/フラグメントのレイアウト、またはRecyclerViewがされるには、RecyclerViewがまれていなければなりません。 ScrollViewやのレイアウトはありません。

```
<?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>
```

データモデルをする

`int`、 `String`、 `float[]`、 `CustomObject`など、のクラスまたはプリミティブデータをモデルとしてできます。 RecyclerViewは、このオブジェクト/プリミティブの`List`をします。

リストがテキスト、、このではをむのようなるデータをするは、カスタムオブジェクトをすることをおめします。

```
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;
    }
}
```

リストアイテムのレイアウト

リストにされるxmlレイアウトファイルをするがあります。このでは、イメージには`ImageView`、には`TextView`がされています。 `LinearLayout`は`ImageView`をにし、 `TextView`はににします。

```
<?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>
```

RecyclerViewアダプタとViewHolderをする

に、あなたがするがあり`RecyclerView.Adapter`と`RecyclerView.ViewHolder`。 のクラスはのようになります。

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    public class ViewHolder extends RecyclerView.ViewHolder {
        // ...
    }
}
```

まず、 `ViewHolder`を し`ViewHolder`。 デフォルトコンストラクタをし、なビュ—をいくつかのフィ—ルドにします

```
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);
    }
}
```

アダプタのコンストラクタは、されるデータセットをします。

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    private List<Place> mPlaces;

    public PlaceListAdapter(List<Place> contacts) {
        mPlaces = contacts;
    }

    // ...
}
```

カスタムリストアイテムレイアウトをするには、 `onCreateViewHolder(...)`メソッドをオーバーライドします。このでは、レイアウトファイルの`place_list_item.xml`は`place_list_item.xml`です。

```
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);
    }

    // ...
}
```

`onBindViewHolder(...)`では、にビューのコンテンツをしています。されたの`List`でそれをつけて、`ViewHolder`のビューにイメージとをして、されたモデルをします。

```
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());
    }

    // ...
```

```
}
```

また、 `List`のサイズをす`getItemCount()`をするがあります。

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public int getItemCount() {
        return mPlaces.size();
    }

    // ...
}
```

ランダムデータをする

このでは、ランダムなをします。

```
@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;
}
```

RecyclerViewとPlaceListAdapterおよびデータセットをする

`RecyclerView`とアダプタのはにです。リストレイアウトをするには、レイアウトマネージャとして`LinearLayoutManager`をするがあります。

```
@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));
}
```

StaggeredGridLayoutManager

1. レイアウトXMLファイルにRecyclerViewをします

```
<android.support.v7.widget.RecyclerView
            android:id="@+id/recycleView"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
```

2. データをするModelクラスをします。

```
     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. RecyclerViewアイテムをするレイアウトファイルをします。

```
<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. RecyclerViewのアダプタクラスをします。

```
 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. アクティビティーまたはフラグメントでRecyclerViewをインスタンスする

```
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() );
```

build.gradleファイルにPicassoのをするのをれないでください

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

オンラインでRecyclerViewおよびLayoutManagerをむ
https://riptutorial.com/ja/android/topic/6772/recyclerviewおよびlayoutmanager

98: RecyclerViewデコレーション

- RecyclerView addItemDecorationRecyclerView.ItemDecoration
- RecyclerView addItemDecorationRecyclerView.ItemDecoration、intインデックス

パラメーター

パラメータ	
	`RecyclerView`にするアイテムの
	この`RecyclerView`ののリストのインデックスこれは、 `getItemOffset`と`onDraw`がびされるです。のびしでのびしをるがあります。

はです

はされるだけなので、クリックリスナーやそののUIをそれらにすることはできません。

の

`RecyclerView`のデコレーションをすることはによってはしますが、やのにのをするパブリックAPIはありません。ビューまたはビューをすることができます。は、されたすべてのオフセットのです。

そののトピック

RecyclerView
RecyclerView onClickListener

javadoc

https://developer.android.com/reference/android/support/v7/widget/RecyclerView.ItemDecoration.html

Examples

セパレータの

これは、すべてのビューのにをきますが、のはのセパレータとしてします。

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

ItemDecorationでのアイテムのマージン

RecyclerView.ItemDecorationをして、 RecyclerView.ItemDecorationのにをすることができます。これにより、アダプタとアイテムビューXMLのをクリーンアップすることができます。

```
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);
    }
}
```

デコレーションをにするには、にRecyclerViewにします。

```
// in your onCreate()
RecyclerView rv = (RecyclerView) findItemById(R.id.myList);
rv.addItemDecoration(new MyItemDecoration(context));
```

RecyclerViewにディバイダをする

まず、 RecyclerView.ItemDecorationするクラスをするがあります。

```
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);
    }
}

}
```

`divider_blue.xml`をあなたの`divider_blue.xml`フォルダにしてください

```
<?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>
```

に、のようにします。

```
recyclerView.addItemDecoration(new SimpleBlueDivider(context));
```

はのようになります


28% 14:20
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)

このは、をするにMaterial Designのにったければ、をどのようにさせるかのにすぎません。このリンクをてください とリンクをすることによって@Brenden Kromhoutにします。

とDividerItemDecorationをってりをする

DividerItemDecorationはアイテムのとしてできるRecyclerView.ItemDecorationです。

```
DividerItemDecoration mDividerItemDecoration = new DividerItemDecoration(context,
             mLayoutManager.getOrientation());
recyclerView.addItemDecoration(mDividerItemDecoration);
```

DividerItemDecoration.VERTICALとDividerItemDecoration.HORIZONTALをして
DividerItemDecoration.VERTICALをDividerItemDecoration.HORIZONTALます。

RecycleViewのGridLayoutManagerにするItemOffsetDecoration

のは、GridLayoutのアイテムにしいスペースをえるのにちます。

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;
    }
}
```

のコードのようにItemDecorationをびすことができます。

```
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);
```

アイテムオフセットの

```
<dimen name="item_offset">5dp</dimen>
```

オンラインでRecyclerViewデコレーションをむ
https://riptutorial.com/ja/android/topic/506/recyclerviewデコレーション

99: RecyclerViewのページネーション

き

ページネーションは、データのリストをするがあるくのモバイルアプリにとってのです。モバイルアプリケーションのほとんどは、スクロールがにしいコンテンツにみまれる「エンドレスページ」モデルをしめています。 CWACエンドレスアダプタは、Androidアプリケーションでこのパターンをするのがとてもです

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

パブリックインターフェイス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);
        }
    }

}
```

オンラインでRecyclerViewのページネーションをむ
https://riptutorial.com/ja/android/topic/9243/recyclerviewのページネーション

100: RenderScript

き

RenderScriptは、なグラフィックレンダリングとのコードをできるスクリプトです。これは、システムがでできるプロセッサのネイティブコードにコンパイルするためになコードをくをします。これは、CPU、マルチコアCPU、さらにはGPUでもかまいません。には、がすぐにできないくのにしますが、プラットフォームコンパイラがサポートするアーキテクチャにもします。

Examples

RenderScriptは、Androidでのをにするフレームワークです。あなたがいたスクリプトは、なてのプロセッサえば、CPU、GPUなどにわたってにされ、スケジューリングされされるではなく、したいタスクにすることができます。

スクリプトはC99ベースのでかれていますC99はCプログラミングのいバージョンです。スクリプトごとに、JavaコードのRenderScriptとにやりりできるJavaクラスがされます。

プロジェクトの

Androidフレームワークライブラリまたはサポートライブラリをして、アプリケーションのRenderScriptにアクセスするには2つのがあります。 APIレベル11よりにデバイスをターゲティングしたくないでも、さまざまなデバイスでデバイスのがされるため、サポートライブラリのをにするがあります。サポートライブラリのをするには、なくともビルドツールバージョン`18.1.0`をするがあります。

に、アプリケーションのbuild.gradleファイルをセットアップします。

```
android {
    compileSdkVersion 24
    buildToolsVersion '24.0.1'

    defaultConfig {
        minSdkVersion 8
        targetSdkVersion 24

        renderscriptTargetApi 18
        renderscriptSupportModeEnabled true
    }
}
```

- `renderscriptTargetApi` なすべてのRenderScriptをするもいAPIレベルにするがあります。
- `renderscriptSupportModeEnabled` サポートライブラリRenderScriptのをできるようにします。

RenderScriptのしくみ

なRenderScriptは、カ―ネルとという2つのでされています。はちょうどそのようなものです。はをけり、そのでかをい、をします。カ―ネルは、RenderScriptののパワ―がられるです。

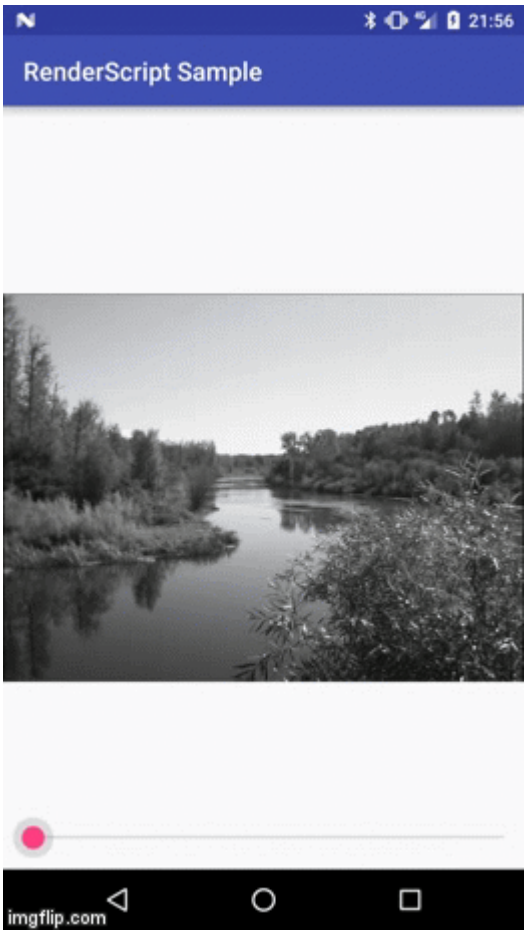
カ―ネルは、 `Allocation`のすべてのにしてされるです。 `Allocation`は、 `Bitmap`や`byte`のようなデ―タを`RenderScript`にすためにすることができ、カ―ネルからをるためにもされます。カ―ネルは、1つの`Allocation`をとして、もう1つをとして、または1つの`Allocation`のデ―タをすることができます。

あなたはつのカ―ネルをくことができますが、Gaussian Image Blurのようななをするためにできるくのみのカ―ネルもあります。

すべてのRenderScriptファイルについてにべたように、クラスはそれとするためにされます。これらのクラスはに`ScriptC_` RenderScriptファイルのでまります。たとえば、RenderScriptファイルがexampleとばれる`example` 、されたJavaクラスは`ScriptC_example`とばれ`ScriptC_example` 。 にされたすべてのスクリプトは`Script ScriptIntrinsicBlur`ます。たとえば、Gaussian Image Blur Scriptは`ScriptIntrinsicBlur`です。

あなたののRenderScriptをく

のは、GitHubのにづいています。これは、のをすることによって、なをします。 ここでソ―スコ―ドをつけて、それをでしてみたいとっているなら、それをチェックすることができます。はのようになります。



RenderScriptボイラープレート

RenderScriptファイルはプロジェクトの`src/main/rs`フォルダにあります。ファイルにはファイル`.rs`あり、に2つの`#pragma`がまれているがあります。

```
#pragma version(1)
#pragma rs java_package_name(your.package.name)
```

- `#pragma version(1)` しているRenderScriptのバージョンをできます。、バージョン1のみがします。
- `#pragma rs java_package_name(your.package.name)` このRenderScriptとするためにされたJavaクラスのパッケージをするためにできます。

の`#pragma`があります。、RenderScriptファイルにするがあり、をするためにされます。は3つのなるレベルにできます。

- `#pragma rs_fp_full` これはのもしいで、もしないのデフォルトです。いがなは、これをするがあります。
- `#pragma rs_fp_relaxed` これはいをするものではありませんが、のアーキテクチャではスクリプトをできるようながです。
- `#pragma rs_fp_imprecise` これにより、よりがくなり、がスクリプトににしないにするがあります。

ほとんどのスクリプトは`#pragma rs_fp_relaxed`うことができ`#pragma rs_fp_relaxed`ただし、がになをきます。

グロ―バル

Cコードのように、グロ―バルやをすることができます

```
const static float3 gMonoMult = {0.299f, 0.587f, 0.114f};

float saturationLevel = 0.0f;
```

`gMonoMult`のは`float3`です。 つまり、3つのからなるベクトルです。の`floatsaturationValue`はではないため、にきなにすることができます。あなたのカ―ネルやでこのようなをうことができるので、RenderScriptにをえたり、をけったりするのです。ではないのそれぞれにして、けられたJavaクラスでgetterおよびsetterメソッドがされます。

カ―ネル

しかし、ではカ―ネルをすることができます。このでは、イメ―ジのをするためにカ―ネルでされたについてするつもりはなく、カ―ネルをするとそのにをてます。こののでは、このカ―ネルのコ―ドがにをしているのかをにします。

なカ―ネル

にソ―スコ―ドをてみましょう

```
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);
}
```

ごのとおり、のCのようにえますが、りのとメソッドのには`__attribute__((kernel))`があります。これは、このメソッドがカ―ネルであることをRenderScriptにらせるものです。 あなたがくかもしれないのことは、このメソッドが`uchar4`パラメ―タをけれ、の`uchar4`をすことです。 `uchar4`は、のでした`float3`のように、ベクトルです。それは0から255までののバイトである4つの`uchar`をんで`uchar`ます。

これらの々のにはさまざまなでアクセスできます。たとえば、 `in.r`はピクセルのいチャネルにするバイトをします。それぞれのピクセルはのは`r` 、のは`g` 、のは`b` 、のは`a` 、アルファのは`b`の4つのでされているため、 `uchar4`をします。 RenderScriptでは、ベクトルからののをりして、のベクトルをすることもできます。たとえば、 `in.rgb`は、アルファのないピクセルの、、のだけをむ`uchar3`をします。

に、RenderScriptはイメ―ジのピクセルにしてこのKernelメソッドをびします。そのは、りとパ

ラメータがただ1つの`uchar4`にすぎない`uchar4`です。 RenderScriptは、なすべてのプロセッサでこれらのびしのくをにするため、RenderScriptはにです。これは、スレッディングやスレッドのについてするがないことをします。ピクセルにをしたいのかをするだけで、RenderScriptはりのをします。

Javaでカ―ネルをびすときは、2つの`Allocation`をします.1つはデ―タをみ、もう1つはをけります。あなたのカ―ネルメソッドは、`Allocation`にしてびされ、そのを`Allocation`きみます。

RenderScriptランタイムAPIメソッド

のカ―ネルでは、いくつかのがされています。 RenderScriptにはくのメソッドがされており、RenderScriptをしてするほとんどのものにとってです。それらのには、 `sin()`ようなをうメソッドと、のにって2つのをする`mix()`ようなヘルパ―メソッドがあります。しかし、ベクトル、、およびをうときには、よりなのためのメソッドもあります。

の**RenderScriptランタイムAPI**リファレンスは、のメソッドのをりたいや、のをするなどのなをするのメソッドをしているに、なリソ―スです。このドキュメントはここでつけることができます。

カ―ネルの

さて、このカ―ネルがをしているのかをてみましょう。カ―ネルののはのとおりです

```
float4 f4 = rsUnpackColor8888(in);
```

のは、 `uchar4`を`float4`にする`uchar4`メソッド**`rsUnpackColor8888()`**をびします。チャネルは、にさ`0.0f - 1.0f0.0f`バイトのにする`0`び`1.0f`に`255`。 これのなは、このカ―ネルのすべてのをもっとにすることです。

```
float3 dotVector = dot(f4.rgb, gMonoMult);
```

こののは、みみメソッド**`dot()`**をして、2つのベクトルのをします。 `gMonoMult`はのいくつかのをしたです。のベクトルはドットをするのにじさでなければならないので、ピクセルのアルファチャンネルではなくカラ―チャンネルにをえたいので、 `float3`ベクトルをるために`.rgb`をします。、およびのチャネルをむ。ドットプロダクトがどのようにするかをからえている々は、ドットプロダクトがベクトルではなく1つのをすがあることをすぐにします。のコ―ドでは、を`float3`ベクタにしています。これもRenderScriptのです。 1のをベクトルにすると、ベクトルのすべてのがこのにされます。たとえば、のスニペットは、 `float3`ベクトルの3つののそれぞれに`2.0f`をりてます。

```
float3 example = 2.0f;
```

のドットのは、の`float3`ベクトルのにりてられます。

は、グロ―バル`saturationLevel`をってイメ―ジの`saturationLevel`をするがあります

```
float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
```

これはみみのメソッド`mix()`をしてのとでしたドットベクトルをします。どのようにされるかは、グローバル`saturationLevel`によってまります。したがって、`saturationLevel`が`0.0f`、のカラーはのカラーのをたず、としてまたはグレーアウトされたイメージになる`dotVector`ののみでされます。`1.0f`のを`1.0f`すると、のはののでにされ、`1.0f`とのがえてるくやかになります。

```
return rsPackColorTo8888(newColor);
```

これはカーネルののです。`rsPackColorTo8888()`は、`float3`ベクトルを`uchar4`にしてします。のバイトは0255のにクランプされるため、`1.0f`よりきいは255のバイトとなり、`0.0`よりさいは`0`バイトになり`0`。

それがカーネルのです。ではっているは1つだけですJavaでカーネルをびす。

JavaでRenderScriptをびす

RenderScriptファイルについてにしたように、Javaクラスがされ、スクリプトとできます。これらのファイルには、`ScriptC_`というのにRenderScriptファイルのがきます。これらのクラスのインスタンスをするには、`RenderScript`クラスのインスタンスがです。

```
final RenderScript renderScript = RenderScript.create(context);
```

メソッド`create()`をして、`Context`から`RenderScript`インスタンスをできます。その、スクリプトにされたJavaクラスをインスタンスできます。RenderScriptファイル`saturation.rs`をびした、そのクラスは`ScriptC_saturation`とばれ`ScriptC_saturation`

```
final ScriptC_saturation script = new ScriptC_saturation(renderScript);
```

このクラスでは、レベルをしてカーネルをびすことができます。`saturationLevel`のためにされたsetterは、`set_`にのがきます。

```
script.set_saturationLevel(1.0f);
```

また、されているレベルをできる`get_`というがいたゲッターもあります。

```
float saturationLevel = script.get_saturationLevel();
```

RenderScriptでしたカーネルのには、`forEach_`カーネルメソッドのがきます。々がいたカーネルは、`Allocation`と`Allocation`をそのパラメータとしてしている

```
script.forEach_saturation(inputAllocation, outputAllocation);
```

`Allocation`はイメージがまれているがあり、`forEach_saturation`メソッドのにりてにされたイメー

ジデータがまれます。

Allocationインスタンスをすると、メソッドcopyFrom()およびcopyTo()をして、これらのAllocationsとのでデータをコピーできます。たとえば、しいを`Call by Allocation'というにコピーすることができます

```
inputAllocation.copyFrom(inputBitmap);
```

Allocation copyTo()をびすことによってイメージをするのとじです。

```
outputAllocation.copyTo(outputBitmap);
```

りてインスタンスの

Allocationをするはたくさんあります。あなたがっているAllocationあなたがそれらからとにしいデータをコピーすることができインスタンスをAllocationsとcopyTo()とcopyFrom()のようには、でしますが、にそれらをするために、あなたはにしているデータのをっているがあります。のAllocationからめましょう

メソッドcreateFromBitmap()をして、 Bitmapから Allocationをくすることができます。

```
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, image);
```

このでは、イメージはしてされないため、のAllocationするはありません。 saturationLevelがされてしいBitmapされるたびに、することができます。

Allocationはもうしです。まず、 Typeとばれるものをするがあります。 Typeは、どのようなのデータをっているかをAllocationためにされます。、 Type.Builderクラスをしてな Typeをくします。にコードをてみましょう

```
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
        .setX(inputBitmap.getWidth())
        .setY(inputBitmap.getHeight())
        .create();
```

たちは、の32ビットつまり、4バイトのBitmapを4つのカラーチャネルでしています。そのため、 Typeをするには Element.RGBA_8888をしています。その、々は、メソッドsetX()とsetY() 、とじサイズにのとさをします。に、 create()メソッドは、したパラメータで Typeをします。

々はしいたら Type 、たちは、をすることができます Allocationメソッドと createTyped()

```
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);
```

々はほぼしている。 Allocationからのデータをコピーできる Bitmapもです。これをうには、メソッドcreateBitmap()をして、 Bitmapとじサイズとでしいの Bitmapをします。

```
final Bitmap outputBitmap = Bitmap.createBitmap(
        inputBitmap.getWidth(),
        inputBitmap.getHeight(),
        inputBitmap.getConfig()
);
```

そしてそれでたちはRenderScriptをするためのすべてのパズルをっています。

な

ここでは、これらをまとめて1つのとしてします。

```
// 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);
```

このでは、なのためにのRenderScriptカ―ネルをするようにするがあります。しかし、あなたがえておかなければならないことがいくつかあります

- **RenderScript**はアプリケ―ションプロジェクトでのみします。、RenderScriptファイルはライブラリプロジェクトのにすることはできません。

- メモリにしてくださいRenderScriptはにですが、メモリをにすることもあります。いつでも`RenderScript`インスタンスは2つしてはいけません。また、できるだけするがあります。、`Allocation`インスタンスをするだけで、にそれらをすることができます。`Bitmaps`やスクリプトインスタンスもです。できるだけしてください。
- バックグラウンドでの やはりRenderScriptはにいですが、してにはできません。カ―ネル、になカ―ネルは、`AsyncTask`などのUIスレッドからするがあります。しかし、ほとんどの、メモリリ―クをするはありません。すべてのRenderScriptクラスはアプリケ―ション`Context`のみをするため、メモリリ―クはしません。しかし、あなたは、あなたがをしている`View`、`Activity`または`Context`インスタンスをらすようなのものについては、まだするがあります
- みみのものをう ぼかし、ブレンド、、サイズなどのタスクをするみのスクリプトがあります。また、カ―ネルをするのにつがたくさんあります。あなたがかしたいのであれば、あなたがやろうとしていることをにっているスクリプトやメソッドがあるがあります。をしないでください。

のコ―ドですぐにいめるには、このチュ―トリアルでしたをしたGitHubプロジェクトのをごください。 ここでプロジェクトをつけることができます。 RenderScriptをしんでください

をぼかす

このでは、Renderscript APIをしてをぼかしビットマップをするをします。このでは、ScriptInstrinsicBlurアンドロイドのrenderScript APIAPI> = 17によってされます。

```
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;
    }
}
```

スクリプトにはデ―タをするカ―ネルがあり、に`forEach`メソッドをしてびされます。

```
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
    }
}
```

これでここでのがづけられました。バックグラウンドスレッドですることをおめします。

ビュ—をぼかす

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;
    }
}
```

```
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);
```

オンラインでRenderScriptをむ https://riptutorial.com/ja/android/topic/5214/renderscript

101: RoboGuice

Examples

な

RoboGuiceは、GoogleのGuiceライブラリをして、Androidへののシンプルさとさをもたらすフレームワークです。

```
@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 );
    }
}
```

Gradleプロジェクトのインストール

あなたのgradleビルドファイルのセクションにのpomをしてください

```
project.dependencies {
    compile 'org.roboguice:roboguice:3.+'
    provided 'org.roboguice:roboblender:3.+'
}
```

@ContentViewアノテーション

@ContentViewアノテーションをすると、アクティビティのをさらにし、setContentViewをきえることができます。

```
@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アノテーション

のタイプのリソース、、アニメーション、なものなどをできます。

のリソースをアクティビティにするには、のがです。

- RoboActivityから
- @InjectResourceでリソースにをける

```
@InjectResource(R.string.app_name) String name;

@InjectResource(R.drawable.ic_launcher) Drawable icLauncher;

@InjectResource(R.anim.my_animation) Animation myAnimation;
```

@InjectViewアノテーション

@InjectViewアノテーションをしてのビューをすることができます

あなたはをうがあります

- RoboActivityから
- コンテンツビューをする
- @InjectViewでビューにをける

```
@InjectView(R.id.textView1) TextView textView1;

@InjectView(R.id.textView2) TextView textView2;

@InjectView(R.id.imageView1) ImageView imageView1;
```

RoboGuiceの

`RoboGuice`は、GoogleのGuiceライブラリをして、Androidへののシンプルさとさをもたらすフレームワークです。

RoboGuice 3はアプリケーションコードをスリムします。コードがなくても、バグのはなくなります。また、コードをにフォローすることができます。コードはもはやAndroidプラットフォームのみではなく、アプリケーションののロジックにをてることができます。

あなたにアイデアをえるために、なAndroid `Activity`このなをてみましょう

```
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 );
        }
    }
```

このは19のコードです。 `onCreate()` をもうとしているは、5ののをスキップして、なのものをつけなければなりません `name.setText()` 。そして、なアクティビティはこののコードのくをもたらすがあります。

これを`RoboGuice`をってかれたじアプリとする

```
 @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 );
        }
    }
```

RoboGuiceのは、Androidでにするがあるすべてのとライフサイクルコードについてではなく、あなたのコードをあなたのアプリにすることです。

@ContentView

@ContentViewアノテーションをすると、アクティビティのをさらにし、setContentViewをきえることができます。

```
@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アノテーション

まず、RoboActivityをするアクティビティがです。に、あなたのres / animフォルダにアニメーションmy_animation.xmlがあるとして、アノテーションをけてできます。

```
public class MyActivity extends RoboActivity {
```

```
    @InjectResource(R.anim.my_animation) Animation myAnimation;
    // the rest of your code
}
```

@

あなたのがRoboActivityからされていることをし、システムサービスのメンバーに@Injectをします。ロボグィスはりのことをします。

```
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();
```

ビュー、リソース、サービス、およびそののアンドロイドのものにえて、RoboGuiceはPlain Old Java Objectsをインジェクトすることができます。デフォルトでは、RoboguiceはあなたのPOJOでなしのコンストラクタをびします

```
class MyActivity extends RoboActivity {
    @Inject Foo foo; // this will basically call new Foo();
}
```

オンラインでRoboGuiceをむ https://riptutorial.com/ja/android/topic/2563/roboguice

102: RxJavaをったRetrofit2

Examples

RxJavaをったRetrofit2

まず、するをbuild.gradleファイルにします。

```
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'
    ....
}
```

に、したいモデルをします。

```
public class Server {
    public String name;
    public String url;
    public String apikey;
    public List<Site> siteList;
}
```

リモートサーバーとのデータにされるメソッドをむインターフェイスをします。

```
public interface ApiServerRequests {

    @GET("api/get-servers")
    public Observable<List<Server>> getServers();
}
```

に、 `Retrofit`インスタンスをします。

```
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;
}
```

に、コードのどこからでもメソッドをびします。

```
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
       }
   });
```

データをにするためのRxJavaによる

RxJavaのGitHubリポジトリから、 *RxJava*は、*Reactive Extensions*の*Java VM*です。これは、なシーケンスをしておよびイベントベースのプログラムをするためのライブラリです。これはオブザーバパターンをしてデータ/イベントのシーケンスをサポートし、レベルのスレッド、、スレッドセーフティ、およびデータなどのをしながらシーケンスをにみわせるをします。

Retrofitは、AndroidとJavaのためのタイプセーフなHTTPクライアントです。これをすると、はすべてのネットワークをよりにすることができます。として、いくつかのJSONをダウンロードし、それをRecyclerViewにリストとしてします。

アプリケーションレベルのbuild.gradleファイルにRxJava、RxAndroid、Retrofitのをします

```
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"
```

ApiClientと**ApiInterface**をしてサーバからデータをする

```
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;
}}
```

パブリックインターフェイスApiInterface {

```
@GET("repos/{org}/{repo}/issues")
Observable<List<Issue>> getIssues(@Path("org") String organisation,
                                  @Path("repo") String repositoryName,
                                  @Query("page") int pageNumber);}
```

getReposは、のリストだけでなくObservableをすることにしてください。

モデルをする

これのがされています。 JsonSchema2Pojoなどのサービスをできます。

```
public class Comment {

@SerializedName("url")
@Expose
private String url;
@SerializedName("html_url")
@Expose
private String htmlUrl;

//Getters and Setters
}
```

レトロフィットインスタンスの

```
ApiInterface apiService = ApiClient.getInstance().create(ApiInterface.class);
```

に、このインスタンスをしてサーバーからデータをします

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

これで、RetrofitとRxJavaをしてサーバーからデータをにできました。

ネストされたリクエストののリクエスト、の

1つのリクエスト `getAllPets` と1つのリソース `getSinglePet` のなデータをつのリクエストでオブジェクトメタデータをできるAPIがあるとします。どのようにしてそれらのすべてをのチェーンでクエリできますか

```
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;
    }
}
```

}

オンラインでRxJavaをったRetrofit2をむ https://riptutorial.com/ja/android/topic/7632/rxjavaをったretrofit2

103: SearchView

Examples

Appcompat SearchView with RxBindingsウォッチャー

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);
        }
    }
}
```

フラグメントきツールバーの**SearchView**

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);
    }
}
```

リファレンススクリーンショット



Hello Android

ツーールバーにされるスタイルににしていることをするがあります。テーマをするために、ツールバーはのをします。

`styles.xml`スタイルをする

```
<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>
```

スタイルをツールバーにします。

```
<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" />
```

これにより、ツールバーにするすべてのビューるボタン、メニューアイコン、およびSearchViewにましいがえられます。

オンラインでSearchViewをむ https://riptutorial.com/ja/android/topic/4786/searchview

104: SensorManager

Examples

センサイベントの

オンボードセンサーからのセンサーの

```
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
    }
}
```

へのセンサ

Androidからされるセンサーは、のにしていますたとえば、+ Yがのにかっている。これらのセンサーを、センサー・マネージャーのをしてワールドにすることができますえば、の+ Y、にをける+ Y

まず、データがされる/をしてするがありますこれは`onCreate`メソッドなどでうことができます。

```
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];
```

に、センサのをし、するにしで/のでしたい、とワールドへのをするがあります。

```
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];

}
```

をして、デバイスがかどうかをする

`onCreate()` / `onResume()`メソッドにのコードをします。

```
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);
```

で`movementThreshold`をして、をするがあるかもしれません。に、 `onSensorChanged()`メソッドをのようにオーバーライドします。

```
@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);
    }
}
```

をたないデバイスにアプリがインストールされないようにするには、マニフェストにのをするがあります

```
<uses-feature android:name="android.hardware.sensor.accelerometer" />
```

オンラインでSensorManagerをむ https://riptutorial.com/ja/android/topic/3344/sensormanager

105: SharedPreferences

き

SharedPreferencesは、データをキーとのペアのでディスクにするをします 。

- コンテキスト メソッド
  - public SharedPreferences getSharedPreferences、intモード
- アクティビティ メソッド
  - public SharedPreferences getPreferences
- **SharedPreferences** メソッド
  - public SharedPreferences.Editor edit
  - public boolean contains
  - public Map <String、> getAll
  - public boolean getBooleanString key、boolean defValue
  - public float getFloatString key、float defValue
  - public int getIntString key、int defValueされたをします。
  - public long getLongString key、long defValue
  - public String getStringString key、String defValue
  - public Set getStringSetString key、Set defValues
  - public void registerOnSharedPreferenceChangeListener SharedPreferences.OnSharedPreferenceChangeListener listener
  - public void unregisterOnSharedPreferenceChangeListener SharedPreferences.OnSharedPreferenceChangeListener listener
- **SharedPreferences.Editor** メソッド
  - public void apply
  - public boolean commit
  - public SharedPreferences.Editor clear
  - public SharedPreferences.Editor putBooleanString key、boolean value
  - パブリックSharedPreferences.Editor putFloatString key、float value
  - パブリックSharedPreferences.Editor putIntString key、int value
  - パブリックSharedPreferences.Editor putLongString key、long value
  - パブリックSharedPreferences.Editor putStringString key、String value
  - public SharedPreferences.Editor putStringSetキー、の
  - パブリックSharedPreferences.Editor removeString key

パラ メーター

パラメータ	
キー	パラメータをするnullの`String`それは、またはをむことができます。これはアプリケーションおよびXMLファイルでのみされるため、にするはありませんが、ソースコードでとしてすることをおめします。それをローカライズしないでください。
defValue	すべてのgetはをとり、されたキーが`SharedPreferences`しないにされます。キーがしていてもがっているはされません。そのは`ClassCastException`され`ClassCastException`。

- `SharedPreferences`はのデータのにはしないでください。そのようなのためには、`SQLiteDatabase`をうほうがずっとい`SQLiteDatabase`。
- モード`MODE_MULTI_PROCESS`をしないり、`SharedPreferences`はプロセスのみです。したがって、アプリにのプロセスがある、のプロセスでメインプロセスの`SharedPreferences`をみることができなくなります。このようなは、プロセスでデータをするためにのメカニズムをするがありますが、`MODE_MULTI_PROCESS`をしないでください。
- `Singleton`クラスの`SharedPreferences`インスタンスをして、アプリケーション`context`にアクセスするがよいでしょう。のアクティビティにしてのみするは、`getPreferences()`ます。
- `SharedPreferences`をしているは、みやすいのでをでしないでください。

https://developer.android.com/reference/android/content/SharedPreferences.html

Examples

SharedPreferencesへののみりときみ

```
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()`は、 `Activity`がする`Context`クラスのメソッドです。あなたがアクセスするがあるは`getSharedPreferences()`のクラスからメソッドを、あなたがすることができます`context.getSharedPreferences()`で`Context`からのオブジェクト`Activity` 、 `View` 、または`Application` 。

キーをする

```
private static final String MY_PREF = "MyPref";

// ...

SharedPreferences prefs = ...;

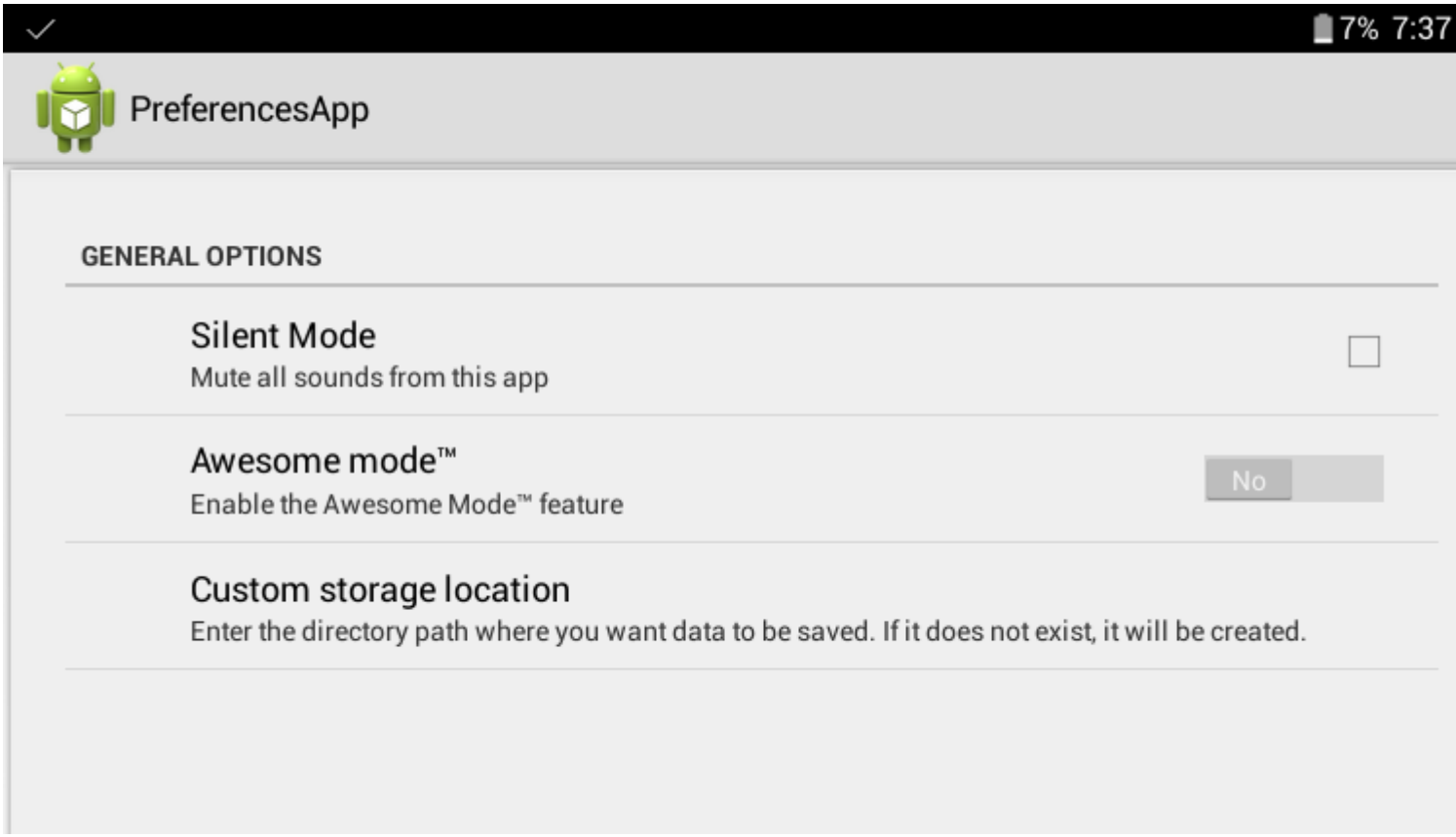
// ...

SharedPreferences.Editor editor = prefs.edit();
editor.putString(MY_PREF, "value");
editor.remove(MY_PREF);
editor.apply();
```

`apply()`に、 `prefs`はにまれているものにえて、 "key" - > "value"がまれます。「キー」をしてしたようにえますが、がににわれます。 `Editor`のはすべて、したではなく、にされます。すべてのは、すべてのにわれます。

SharedPreferencesをしたの

`SharedPreferences` 1つのは、ユーザーがの/オプションをできる「」をアプリケーションにすることです。このような



PreferenceScreenは、 `SharedPreferences`ユーザーをします。 PreferenceScreenをするには、いくつかのことがです。

なオプションをする**XML**ファイル

これは`/res/xml/preferences.xml`にあり、のではのようになります。

```
<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>
```

これは、でなオプションをします。のくのがあり`Preference`のAndroidのドキュメントにされているプリファレンスクラス 。

に、 **Preferences**のユーザーインターフェイスをホストするアクティビティがです。この、かなりく、のようになります。

```
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);
    }
}
```

これは、 `PreferenceActivity`し、のユーザーインターフェイスをします。この、のアクティビティとに、のようにできます。

```
Intent i = new Intent(this, PreferencesActivity.class);
startActivity(i);
```

あなたの`AndroidManifest.xml` `PreferencesActivity`をすることをれないでください。

あなたのアプリケーションののをすることはにです `setDefaultValues()`まずびして、XMLでされているデフォルトをし、デフォルトの`SharedPreferences`してください。

```
//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", "");
```

の**SharedPreferences**ファイルからすべてのされたエントリをする

`getAll()`メソッドは、すべてのをからします。たとえば、 `SharedPreferences`のコンテンツをログに

することができます。

```
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);
    }
}
```

`getAll`された`Collection`にするがにされます

> このメソッドからされたコレクションをしたり、そのをしたりしないでください。したデ―タのはされません。

SharedPreferencesのリスニングの

```
SharedPreferences sharedPreferences = ...;
sharedPreferences.registerOnSharedPreferenceChangeListener(mOnSharedPreferenceChangeListener);


private final SharedPreferences.OnSharedPreferenceChangeListener
mOnSharedPreferenceChangeListener = new SharedPreferences.OnSharedPreferenceChangeListener() {
    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
        //TODO
    }
}
```

ごください

- リスナ―は、valueがまたはされたにのみされます。じをすると、それがびされません。
- ので、リスナ―は、のクラスでメンバと**NOT**にするが`registerOnSharedPreferenceChangeListener`してするので、ごみをすることができるであろう。
- メンバをするわりに、クラスによってされ、`registerOnSharedPreferenceChangeListener(this);`びすこともでき`registerOnSharedPreferenceChangeListener(this);`
- それはもうとしないときに、リスナ―のをすることをれないでください`unregisterOnSharedPreferenceChangeListener`。

シングルトンをって**SharedPreferences**にデ―タをみきする

SharedPreferencesすべてのタイプのデ―タをみきするためのマネ―ジャシングルトンクラス。

```
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);
    }
```

```
}
```

プロガードのをけるためにGsonにくクラスによってされるModel interface。

```
public interface Model {

}
```

Modelインターフェイスのproguardルール

```
-keep interface com.example.app.Model
-keep class * implements com.example.app.Model { *;}
```

SharedPreferencesのオブジェクトをインスタンスするさまざまな

SharedPreferencesには、いくつかのでアクセスできます。

デフォルトのSharedPreferencesファイルをする

```
import android.preference.PreferenceManager;
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
```

のSharedPreferencesファイルをする

```
public static final String PREF_FILE_NAME = "PrefFile";
SharedPreferences prefs = getSharedPreferences(PREF_FILE_NAME, MODE_PRIVATE);
```

のアプリからSharedPreferencesをする

```
// 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);
```

getPreferencesintVS getSharedPreferencesString、int

`getPreferences(int)`

ドキュメントでされているように、 `Activity's class name`によってされたプリファレンスをします。

> このアクティビティのにアクセスするためのSharedPreferencesオブジェクトをします。これはに、このアクティビティのクラスをプリファレンスとしてすことによって、となるgetSharedPreferencesString、intメソッドをびします。

getSharedPreferencesString name、int modeメソッドをすると、されたnameされたprefsがされます。ドキュメントのように

プリファレンスファイル 'name'のをしてし、そのをおよびできるSharedPreferences をします。

したがって、 `SharedPreferences`にされているをアプリでするがあるは、 `getSharedPreferences (String name, int mode)`をでするがあります。 As、 `getPreferences(int)`をすると、それをびす `Activity`するプリファレンスがされ/されます。

コミットと

`editor.apply()`メソッドはですが 、 `editor.commit()`はです。

らかに、 `apply()`または`commit()`いずれかをびすがあります。

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()`が2.3API 9でされた`apply()` 、またはをすブールをさずにコミットします。

`commit()`は、がするはtrueをし、そうでないはfalseをします。

Androidデベロッパーチームがほとんどもりにづいていないことにいたので`apply()`がされたので 、はであるためくなります。

`apply()`は、そのをなストレージににきす`commit()`とはなり、そのをメモリ`SharedPreferences`ただちにコミットしますが、ディスクへのコミットをします。この`SharedPreferences`のエディタが `apply()`がであるにの`commit()`を`commit()`と、すべてのコミットがし、ののコミットがするまで `commit()`がブロックされます。

SharedPreferencesでサポートされるデータ

`SharedPreferences`すると、データ `boolean` 、 `float` 、 `long` 、 `int` 、 `String` 、および`string set` のみをできます。よりなオブジェクトを`SharedPreferences`することはできません。にはユーザーなどをするであるため、ユーザーデータをするデータベースではありませんユーザーがしたToDoリストのなど。

`SharedPreferences`かをするには、KeyとValueをします。キーは、でしたものとするデータをするです。

```
    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();
```

SharedPreferencesからデ―タを、、、およびする

SharedPreferencesをするBuyyaPref

```
SharedPreferences pref = getApplicationContext().getSharedPreferences("BuyyaPref",
MODE_PRIVATE);
Editor editor = pref.edit();
```

KEY / VALUEペアとしてデ―タをする

```
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
```

SharedPreferencesデ―タをする

keyのがしないは、2のparamをしますこのはnull、これはデフォルトのようです

```
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
```

SharedPreferencesからのキ―の

```
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
```

SharedPreferencesからすべてのデ―タをする

```
 editor.clear();
 editor.commit(); // commit changes
```

StringSetをってpre-Honeycombをサポートする

ユーティリティクラスはのとおりです。

```
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() {}
}
```

StringSetデータとしてをするはのとおりです。

```
Set<String> sets = new HashSet<>();
sets.add("John");
sets.add("Nicko");
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
SharedPreferencesCompat.putStringSet(preferences.edit(), "pref_people", sets);
```

それらをりすには

```
Set<String> people = SharedPreferencesCompat.getStringSet(preferences, "pref_people", new
HashSet<String>());
```

リファレンス Androidサポートの

EditTextPreferenceのフィルタをする

このクラスをします。

```
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;
    }
}
```

つかいます

```
EditText compressPic = ((EditTextPreference)
findPreference(getString("pref_key_compress_pic"))).getEditText();
compressPic.setFilters(new InputFilter[]{ new InputFilterMinMax(1, 100) });
```

オンラインでSharedPreferencesをむ
https://riptutorial.com/ja/android/topic/119/sharedpreferences

106: SpannableString

- `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 removeSpanObject what
- `void setSpan (Object what, int start, int end, int flags)`
- `CharSequence subSequence (int start, int end)`
- `String toString ()`
- `SpannableString valueOf (CharSequence source)`

Examples

TextViewにスタイルをする

のでは、のTextViewをするActivityをします。

TextViewはそのコンテンツとして`SpannableString`をします。これはなスタイルのいくつかをします。

ここでは、テキストでをするつもりです

- それをきくする
- な
- アンダーライン
- イタリック
- ストライクスルー
- つき
- された
- きとして
- きとして
- リンクとして
- クリックにします。

```
@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);

}
```

はのようになります。



マルチストリング、マルチカラー

メソッド **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;
}
```

```
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`
```

オンラインでSpannableStringをむ https://riptutorial.com/ja/android/topic/10553/spannablestring

107: SparseArrayのいき

SparseArrayはMapです。 Mapはそのキーがオブジェクトであることをとします。ボクシングのは、プリミティブなintをキーとしてするにします。コンパイラは、プリミティブをボックスされたににします intからInteger。メモリフットプリントのいはつintは4バイト、 Integerは16バイトをする。 SparseArrayはintをキーとしてします。

- メモリがないプリミティブキーのため
- オートボクシングはありません。

- SparseArrayはOlog nにバイナリをするため、のHashMapをでするがある、にならないがあります。

-SparseArray <Integer、Long> -LongSparseArray <Long、Object> -LongSparseLongArray <Long、Long>このようなファミリには、のようながあります。 >

SparseArray

- の - putint、xされたキーからされたへのマッピングをし、されたキーからののマッピングをきえます。 - appendint、xキー/のペアをにし、キーがののキーよりもきいにします。パフォーマンスをするには、キーのはappendをするがあります。それのはputはありません。

- remove element - deleteintされたキーがあるは、そのキーからマッピングをします。 removeAtintされたインデックスのマッピングをします。 removeAtRangeint、intのマッピングをバッチとしてします。

- アクセスしている - getintされたキーからマッピングされたintをします。マッピングがわれていないは0をします。 - getint、Eされたキーからマッピングされたintまたはそのようなマッピングがされていないはされたをします。 valueAtint0 ... size - 1のにあるインデックスをすると、このSparseIntArrayがするindex_key-valueマッピングのをします。インデックスはにべられます。

- index / key search - keyAtint0 ... size - 1ののインデックスをすると、このSparseIntArrayがするインデックスキーとのマッピングからキーをします。インデックスはにべられます。 valueAtint0 ... size - 1のにあるインデックスをすると、このSparseIntArrayがするindex_key-valueマッピングのをします。インデックスはにべられます。 - indexOfKeyintkeyAtintがされたキーをすインデックスをします。されたキーがマップされていないはのをします。 - indexOfValueEvalueAtintがされたキーをすインデックスをします。キーがされたにマップされていないはのをします。これは、キーによるルックアップとはなり、であり、のキーをじにマップすることができ、そのうちの1つだけをつけることにしてください。メモリののいはつintは4バイトをし、Integerは16バイトをします.parseArrayはintをキーとしてします。

Examples

SparseArrayをしたな

```
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);
```

YouTubeでのチュートリアル

オンラインでSparseArrayのいをむ https://riptutorial.com/ja/android/topic/8824/sparsearrayのい

108: SQLite

き

SQLiteはCでかれたリレーショナルデータベースシステムです。 AndroidフレームワークでSQLiteデータベースをするには、 SQLiteOpenHelperをするクラスをし、にじてカスタマイズします。

SQLiteOpenHelperクラスは、static onCreate()およびonUpgrade()メソッドをします。これらのメソッドは、のテーブルでカスタマイズするSQLiteOpenHelperサブクラスのするメソッドでびされます。

Examples

SQLiteOpenHelperクラスの

```
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;
            }
        }
    }
}
```

データをデータベースにする

```
// 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
);
```

onUpgradeメソッド

SQLiteOpenHelperは、データベースのとバージョンをするヘルパークラスです。

このクラスでは、スキーマをするときにonUpgrade()メソッドがデータベースのアップグレードをいます。これは、データベースファイルがすでにするが、そのバージョンがアプリケーションののバージョンでされているバージョンよりもいにびされます。データベースのバージョンごとに、をするがあります。

```
@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;
        }
    }
}
```

カーソルからのデータのみり

に、SQLiteOpenHelperサブクラスにするメソッドのをします。これは、searchTerm Stringをしてをフィルタリングし、Cursorのをし、そのをProduct ObjectsのListにします。

まず、データベースからしたのコンテナになるProduct POJOクラスをします。

```
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;
  }
}
```

に、データベースをするメソッドをし、Product ObjectsのListをします。

```
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();
    }
}
```

AndroidでSQLiteの、ヘルパー、プロバイダをする

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;
    }

}
```

い

```
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;
}
```

テーブルのの

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

トランザクションの

トランザクションをして、データベースにしてのをアトミックにうことができます。のトランザクションはこのパターンにいます。

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

アクティブなトランザクションで`beginTransaction()`びしてもはありません。

テーブルからをする

テーブルからすべてのをするには

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

db.delete(TABLE_NAME, null, null);
db.close();
```

テーブルからすべてのをし、されたのカウントをりでするには

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();
```

```
int numRowsDeleted = db.delete(TABLE_NAME, String.valueOf(1), null);
db.close();
```

WHEREでをするには

```
//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();
```

SQLiteにをする

データベースの

```
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);
    }
}
```

データベースにする

```
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 );
}
```

データの

```
 byte[] image = cursor.getBlob(1);
```

1. データベースにするに、ビットマップイメージをバイトにしてから、データベースクエリをしてするがあります。
2. データベースからするときは、かにバイトのイメージがあります。バイトをのイメージにすことがです。したがって、デコードするにはBitmapFactoryをするがあります。

はがあなたにつことをむユーティリティクラスです

```
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);
    }
}
```

フォルダからデータベースをする

dbname.sqliteまたはdbname.dbファイルをプロジェクトのassetsフォルダにします。

```
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) {

        }
```

```
    }
```

ここであなたのにデ―タベ―スオブジェクトにアクセスするです。

```
// Create Databasehelper class object in your activity.
   private Databasehelper db;
```

にonCreateメソッドでそれをし、のようにcreateDatabaseメソッドをびします。

```
db = new Databasehelper(MainActivity.this);
        try {
            db.createDataBase();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

のように、、、、およびをすべてします。

```
String query = "select Max(Id) as Id from " + TABLE_NAME;
        db.openDataBase();
        int count = db.getId(query);
        db.close();
```

デ―タベ―スのエクスポ―トとインポ―ト

たとえば、bacukupsのためにデ―タベ―スをインポ―トおよびエクスポ―トすることができます。についてれてはいけません。

```
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();
    }
}
```

のデ―タをにするをします。するすべてのデ―タは、ContentValuesのにめられます。

```
@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;
}
```

に、そののをします。

```
ContentResolver resolver = mContext.getContentResolver();
ContentValues[] valueList = new ContentValues[object.size()];
//add whatever you like to the valueList
resolver.bulkInsert(IChatContract.MessageColumns.CONTENT_URI, valueList);
```

オンラインでSQLiteをむ https://riptutorial.com/ja/android/topic/871/sqlite

109: SurfaceViewをしたキャンバス

サーフェスビューのをしてからすることがです。

- これはにのウィンドウのです
- ネイティブUIをそのにすることができます
- はのUIスレッドをしてわれます
- はハードウェアアクセラレーションではありません
- 2つのバッファをします.1つはされ、もう1つはにされます。
- `unlockCanvasAndPost()`はバッファをします。

`lockCanvas()`および`unlockCanvasAndPost()`メソッドがしいでびされないと、デッドロックがしやすくなります。

Examples

スレッドをえたSurfaceView

このでは、のスレッドをしてSurfaceViewをするについてします。このでは、CPUのをするために、ののやスレッドのなどのエッジ・ケースもします。

```
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");
    }
}
```

このレイアウトには、カスタムSurfaceViewのみがまれ、サイズにわせてされます。

```
<?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>
```

SurfaceViewをするアクティビティは、スレッドのとをいます。このは、アクティビティがバックグラウンドでされるとすぐにがされるのでバッテリをします。

```
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();
    }
}
```

オンラインでSurfaceViewをしたキャンバスをむ
https://riptutorial.com/ja/android/topic/3754/surfaceviewをしたキャンバス

110: TabLayout

Examples

ViewPagerなしでのTabLayoutの

たいていの、 `TabLayout`は`TabLayout`とにされ、 のスワイプをします。

することがである`TabLayout`せずに`ViewPager`して`TabLayout.OnTabSelectedListener`。

まず、アクティビティのXMLファイルに`TabLayout`をします。

```
<android.support.design.widget.TabLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:id="@+id/tabLayout" />
```

`Activity`をするには、したタブについてUIをでします。

```
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) {

    }
});
```

オンラインでTabLayoutをむ https://riptutorial.com/ja/android/topic/7601/tablayout

111: TensorFlow

き

TensorFlowは、モバイルおよびみみプラットフォームをしてされています。サンプルコードをしており、これらのプラットフォームですぐすことができます。

Android iOS Raspberry Pi

MindRocksののち

Examples

い

ここからBazelをインストールしてください。 Bazelは、TensorFlowのシステムです。さて、WORKSPACEをすると、にクローンしたTensorFlowのルートディレクトリにあるWORKSPACEファイルをつけることができます。

```
# 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)
```

のようにたちのsdkとndkのパス

```
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)
```

オンラインでTensorFlowをむ https://riptutorial.com/ja/android/topic/9991/tensorflow

112: TextInputLayout

き

TextInputLayoutは、EditTextにラベルをするためにされました。フロー―ティングラベルをするには、EditTextをTextInputLayoutでラップするがあります。

`TextInputLayout`は、ユーザ―がテキストをしたためにヒントがになっているときに、 `EditText` またはをラップしてフロー―ティングラベルをするレイアウトです。 `TextInputLayout` `EditText`すると、 `EditText`にエラー―メッセ―ジをすることができます。

のでアプリケ―ションの`build.gradle`ファイルにのがされていることをしてください。

```
compile 'com.android.support:design:25.3.1'
```

Examples

な

これは`TextInputLayout`ないです。
のでされているように、 `build.gradle`ファイルにをしてください。

```
<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>
```

エラ―の

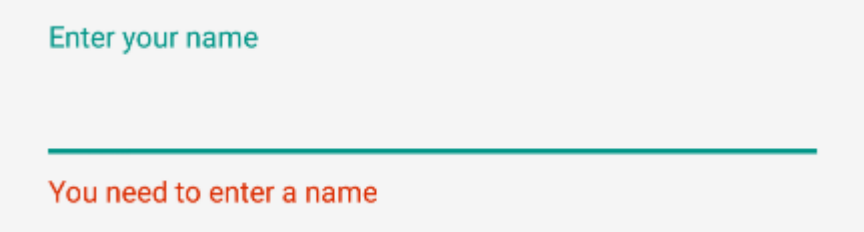
あなたはすることができます`TextInputLayout`にって、エラ―メッセ―ジをするには、 ガイドライン`setError`と`setErrorEnabled`を。

EditTextのにエラ―をするには

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setErrorEnabled(true);
til.setError("You need to enter a name");
```

`TextInputLayout`エラ―をにするには、xmlまたは`til.setErrorEnabled(true);`
`app:errorEnabled="true"`をでき`til.setErrorEnabled(true);`のように。

あなたはをるでしょう



カウントの

TextInputLayoutには、そのにされたEditTextのカウンターがあります。
カウンタはEditTextのにされます。

`setCounterEnabled()`メソッドと`setCounterMaxLength`メソッドをして`setCounterEnabled()`

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setCounterEnabled(true);
til.setCounterMaxLength(15);
```

または`app:counterEnabled`と`app:counterMaxLength` 、XMLの。

```
<android.support.design.widget.TextInputLayout
    app:counterEnabled="true"
    app:counterMaxLength="15">

    <EditText/>

</android.support.design.widget.TextInputLayout>
```

パスワードのりえ

パスワードタイプでは、 `passwordToggleEnabled`をしてテキストをまたはにできるアイコンをにすることもできます。

のをしてじデフォルトをカスタマイズすることもできます。

- `passwordToggleDrawable` デフォルトののアイコンをする
- `passwordToggleTint` パスワードトグルなにいをします。
- `passwordToggleTintMode` いをするためにされるブレンドモードをします。

```
<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はEditTextであり、「」モードのときにIMEにヒントをするためののがえられています 。

Extractモードは、スペースがさすぎるときにEditTextをクリックしたときにキーボードエディタがりわるモードですたとえば、スマートフォンのランドスケープ。
この、テキストのにEditTextをすると、IMEがのもののヒントをえることはありません

TextInputEditTextこのをし、ユーザーのデバイスのIMEがモードになっているにヒントテキストをします。

```
<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>
```

TextInputLayoutののカスタマイズ

あなたはのをカスタマイズすることができTextInputLayoutとそのみみEditTextあなたにカスタムスタイルをすることでstyles.xml。 されたスタイルは、スタイルまたはテーマとしてTextInputLayoutできます。

ヒントのをカスタマイズする

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>
```

スタイルをするにはTextInputLayoutとEditTextをのようにします

```
<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>
```

`TextInputLayout`アクセントカラーをカスタマイズするアクセントのは、 `EditText`のベースラインのとヒントテキストのテキストのにします。

styles.xml

```
<style name="TextInputLayoutWithPrimaryColor" parent="Widget.Design.TextInputLayout">
    <item name="colorAccent">@color/primary</item>
</style>
```

レイアウトファイル

```
<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>
```

オンラインでTextInputLayoutをむ https://riptutorial.com/ja/android/topic/5652/textinputlayout

113: TextView

き

Android SDKのTextViewカスタマイズにするすべて

- TextViewコンテキストコンテキスト
- TextViewfindViewByIdint id
- void setTextint resid
- void setTextCharSequence text//をとしてできます

xmlデザインでするか、プログラムでしてみてください。

Examples

なるテキストサイズのテキストビュー

あなたはスパンでTextviewのなるTextsizeをアーカイブすることができます

```
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)
```

TextViewのカスタマイズ

```
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);
        }
    }
}
```

```
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);
    }
}
```

レイアウト

```
<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>
```

アタレス

```
<?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>
```

プログラムによる

```
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`をAndroidですると、1つの`TextView`ウィジェットでなる、スタイル、サイズ、またはクリックイベントでテキストののをできます。

のように`TextView`をしたとします。

```
TextView textview=findViewById(R.id.textview);
```

に、のようにさまざまなをできます。

- **Spannable**テキストのあるになるをするために、 `ForegroundColorSpan`のにすように、することができます。

  ```
  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 );
  ```

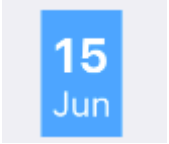
  のコードでされた

- スパニング なフォントテキストのになるフォントサイズをするには、ののように`RelativeSizeSpan`をできます。

  ```
  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 );
  ```

  のコードでされた

- スパニング なテキストのになるフォントをするには、のにすようにカスタムの`TypefaceSpan`

スパンをできます。

```
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 );
```

しかし、のコードをさせるためには、 `CustomTypefaceSpan`クラスを`TypefaceSpan`クラスからさせなければなりません。これはのようにしてうことができます。

```
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);
    }
}
```

イメージき**TextView**

Androidでは、プログラマは`TextView`すべてにをできます。たとえば、 `TextView`をしてフィールドをしているときに、そのフィールドがであることをするとに、はそのフィールドのくにアイコン

をします。 Androidでは、 `TextView` ドロアブルとばれるいオプションがされています。

```
<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" />
```

のように、 `TextView`のにできます。

```
android:drawableLeft="@drawable/edit"
android:drawableRight="@drawable/edit"
android:drawableTop="@drawable/edit"
android:drawableBottom="@drawable/edit"
```

drawableのは、のでプログラムでうこともできます。

```
yourTextView.setCompoundDrawables(leftDrawable, rightDrawable, topDrawable, bottomDrawable);
```

`setCompoundDrawables()`にきされるパラメータを`null`にすると、 `TextView`するサイドからアイコンがされます。

ったTextView

テキストのり

```
String sampleText = "This is a test strike";
textView.setPaintFlags(tv.getPaintFlags()| Paint.STRIKE_THRU_TEXT_FLAG);
textView.setText(sampleText);
```

~~これはテストストライクです~~

テキストのれすぎる

```
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);
```

~~これ~~はテストストライクです

テーマとスタイルのカスタマイズ

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>
```

RelativeSizeSpanをにえる

RelativeSizeSpanえるには、カスタムクラスをSuperscriptSpanクラスからさせることができます。のでは、クラスのはTopAlignSuperscriptSpanです。

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();

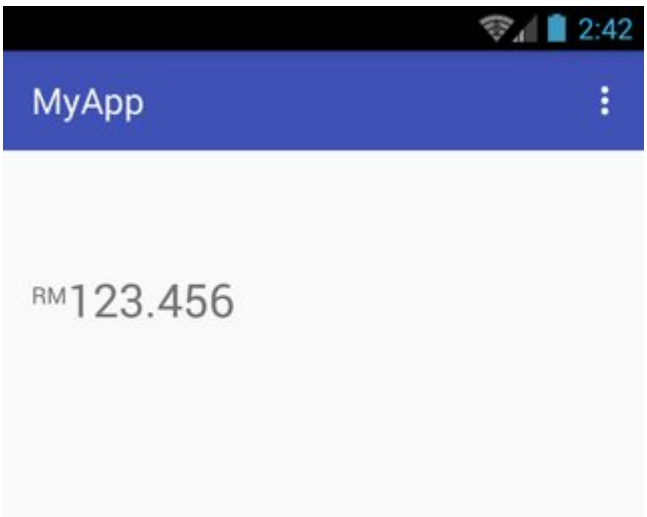
        //scale down the font
        tp.setTextSize( tp.getTextSize() / fontScale );

        //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 );
    }
}
```

リファレンススクリーンショット



TextViewのPinchzoom

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;
    }
}
```

2つのなるをつのTextView

きテキストは、テキストとフォントをのにすことでできます。

```
private String getColoredSpanned(String text, String color) {
    String input = "<font color=" + color + ">" + text + "</font>";
    return input;
}
```

に、のサンプルコードをして、きテキストを`TextView`または`Button`、 `EditText`などにすることができます。

まず、のように`TextView`をします。

```
TextView txtView = (TextView)findViewById(R.id.txtView);
```

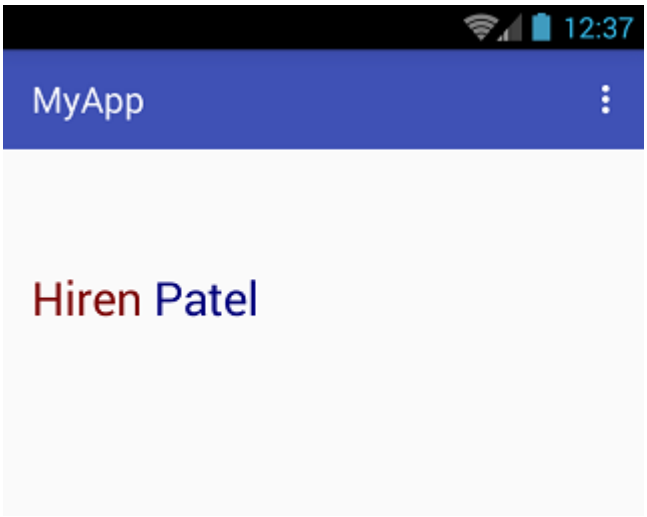
に、なるのテキストをし、にりてます。

```
String name = getColoredSpanned("Hiren", "#800000");
String surName = getColoredSpanned("Patel","#000080");
```

に、のなる2つのを`TextView`にします。

```
txtView.setText(Html.fromHtml(name+" "+surName));
```

リファレンススクリーンショット



オンラインでTextViewをむ https://riptutorial.com/ja/android/topic/4212/textview

114: TransitionDrawable

Examples

2つのにトランジションまたはクロスフェードをします。

ステップ1XMLでをする

プロジェクトの`res/drawable`フォルダに`transition.xml`ファイルをします。

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/image1"/>
    <item android:drawable="@drawable/image2"/>
</transition>
```

image1とimage2は、したい2つので、 `res/drawable`フォルダにもくがあります。

ステップ2あなたのXMLレイアウトにImageViewのコードをしてのオブジェクトをします。

```
<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>
```

ステップ3あなたのアクティビティのonCreateメソッドでXMLトランジションドロワブルにアクセスし、onClickイベントでトランジションをします。

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

TransitionDrawableをしてスイッチカラーをアニメーション

```
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);
}
```

オンラインでTransitionDrawableをむ
https://riptutorial.com/ja/android/topic/6088/transitiondrawable

115: Twitter API

Examples

Twitterボタンでログインをし、それにコールバックをする

1. レイアウトに、のコードをむログインボタンをします。

```
<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. ボタンをするアクティビティまたはフラグメントでは、のようにログインボタンにコールバックをしてするがあります。

```
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. アクティビティのをボタンにす

```
@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);
}
```

、フラグメントのTwitterLoginButtonをしているは、わりにのをします。

```
@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. **build.gradle**のにのをします。

```
apply plugin: 'io.fabric'

repositories {
    maven { url 'https://maven.fabric.io/public' }
}

compile('com.twitter.sdk.android:twitter:1.14.1@aar') {
    transitive = true;
}
```

オンラインでTwitter APIをむ https://riptutorial.com/ja/android/topic/4801/twitter-api

116: Typedef@IntDef、@StringDef

アノテーションパッケージには、のコードをることができるなメタデータアノテーションがまれており、バグをえるのにちます。

`build.gradle`ファイルにをするだけです。

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

Examples

IntDefアノテーション

このアノテーションは、するなのみがされることをします。
のは、をするをしています。

```
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;
    }
}
```

また、コードにより、されたをにすることもできます。
このコードをすると、typeパラメータがされたのいずれかをしないにがされます。

フラグとの

`IntDef#flag()`を`true`にすると、のをみわせることができます。

このトピックのじをする

```
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)

    .....

}
```

ユーザーは、されたをフラグ`|`、`&`、`^`とみわせることができます。

オンラインでTypedef@IntDef、@StringDefをむ
https://riptutorial.com/ja/android/topic/4505/typedef--intdef--stringdef

117: UIAutomatorによるアプリUIテスト

- Instrumentation getInstrumentation
- UIDevice UiDevice.getInstance
- boolean UIDevice.pressHome
- boolean UIDevice.pressBack
- boolean UIDevice.pressRecentApps
- void UIDevice.wakeUp
- boolean UIDevice.swipeint startX、int startY、int endX、int endY、int steps
- ブールUIDevice.dragint startX、int startY、int endX、int endY、int steps
- UIDbject2 UIDevice.findObjectBy.descString contentDesc
- boolean UIObject2.click

UIAutomatorは、にユーザーストーリーのテストにしています。ビューにの*resource-id*も*content-desc*もないは、がします。ほとんどの、とにかくテストをするがありますが、それにはくのがかかります。あなたのアプリのコードにをえることができれば、UIAutomatorがあなたのテストツールかもしれません。

Examples

あなたのプロジェクトをし、のUIAutomatorテストをく

Androidモジュールのbuild.gradleのセクションになライブラリをします

```
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'
}
```

versionsもちろん、バージョンはがなるがあります。

このがされた。

に、androidTestフォルダにしいJavaクラスをします。

```
public class InterAppTest extends InstrumentationTestCase {
```

```
  private UiDevice device;

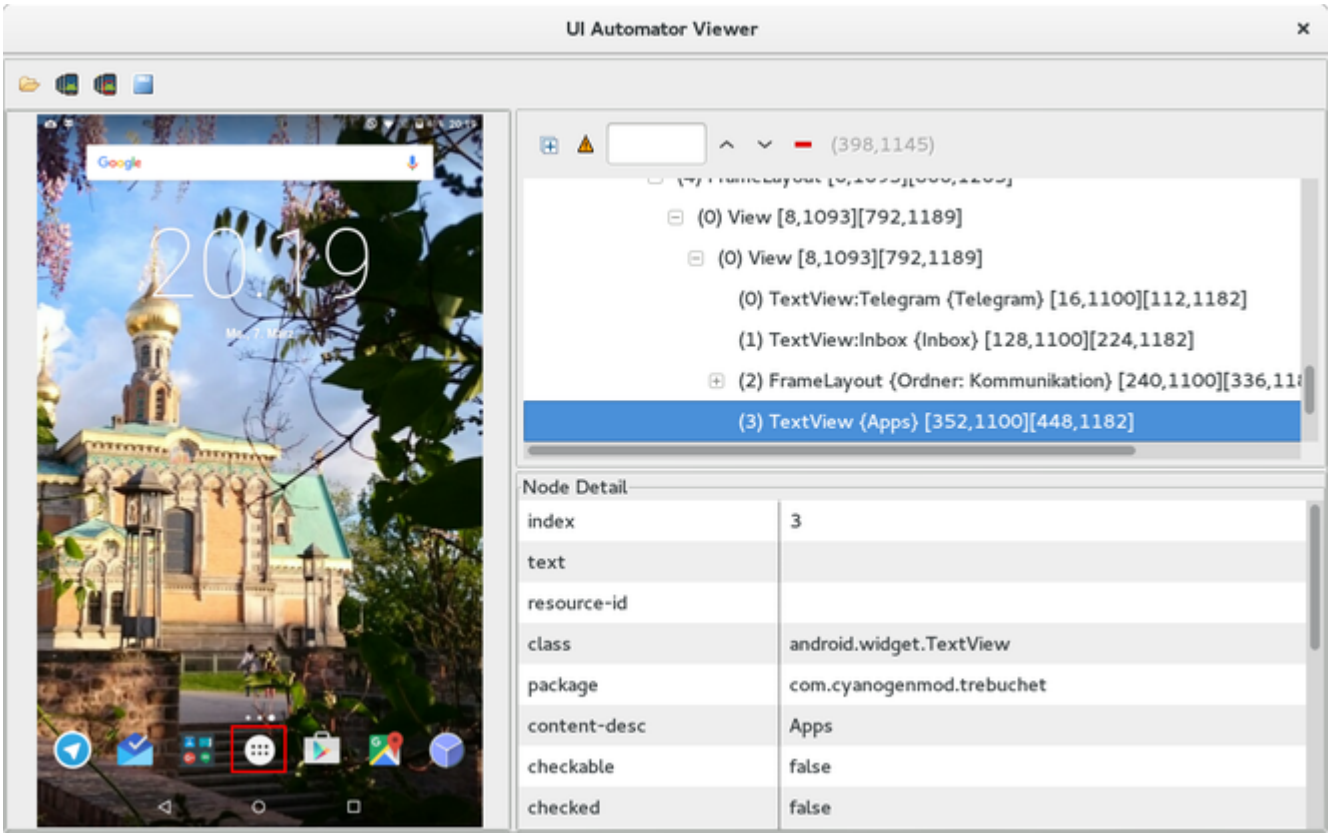
  @Override
  public void setUp() throws Exception {
     device = UiDevice.getInstance(getInstrumentation());
  }

  public void testPressHome() throws Exception {
    device.pressHome();
  }
}
```

クラスタブをクリックして "InterAppTestを"すると、このテストがされます。

UIAutomatorViewerをしたよりなテストの

よりなUIテストをできるようにするには、 *UIAutomatorViewer*がです。 */ tools /*にあるツールは、されているビューのレイアウトをむフルスクリーンのスクリーンショットをします。されるをするには、のをしてください。



UIテストでは、 *resource-id* 、 *content-desc*、またはビューをしてテストでするためのそののものをしています。

*uiautomatorviewer*は、ターミナルでされます。

たとえば、アプリケーションボタンをクリックしてアプリをいてスワイプしたいは、テストメソッドがのようになります。

```
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)
}
```

UIAutomatorテストのテストスイートをする

UIAutomatorテストをテストスイートにまとめるのはすばらしいことです

```
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 {}
```

クリックしてスイートをして、1のテストとにします。

オンラインでUIAutomatorによるアプリUIテストをむ
https://riptutorial.com/ja/android/topic/6249/uiautomatorによるアプリuiテスト

118: UIテストの - Android

き

このドキュメントのは、アンドロイドのUIとテストをくとをすることです。 エスプレッソとUIAutomatorはGoogleによってされるので、これらのツールとそれぞれのラッパーAppium、Spoonなどにをてるがあります。

- アイドリングリソース
- String getName - アイドルのリソースのをしますロギングおよびのにされます。
- boolean isIdleNow - リソースがアイドルのにtrueをします。
- void registerIdleTransitionCallbackIdlingResource.ResourceCallback callback - されたIdlingResource.ResourceCallbackをリソースにします。

JUnitのルール

MockWebServerのとActivityTestRuleでることができるように、これらはすべてJUnitルールのカテゴリにされます。これらのルールは、でし、そのをするテストごとにするがあります。@see https : //github.com/junit-team/junit4/ウィキ/ルール

アピウム

パラメータ

ドキュメントのバグがされるまで、パラメータにはいくつかがあります。

パラメータ	
クラス activityClass	するアクティビティ
initialTouchMode	にアクティビティをタッチモードにするがあります https : //android-developers.blogspot.de/2008/12/touch-mode.html
launchActivity	アクティビティをテストメソッドごとに1するがあるはtrue。のBeforeメソッドのにされ、のAfterメソッドのにします。

Examples

MockWebServerの

あなたの、、およびUIにバックグラウンドがなは、MockWebServerをAndroidデバイスでロ―カルにすることで、UIをじてテストなにすることができます。

https://github.com/square/okhttp/tree/master/mockwebserver

のステップは、gradleをむ

```
testCompile 'com.squareup.okhttp3:mockwebserver:(insert latest version)'
```

モックサ―バ―のとのはのとおりです。

- サ―バオブジェクトをする
- のアドレスとポ―トはlocalhostportnumberでします。
- のにするをエンキュ―する
- テストをする

これはモックウェブサ―バのgithubペ―ジでうまくされていますが、たちの、すべてのテストでよりいものとなものがほしいとっています.JUnitのル―ルはうまくいきます

```
/**
 *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”);
        }
    }
}
```

ここで、ののようなくじアクティビティがあるとしてみましょう。この、ボタンアプリケーションをプッシュすると、ネットワークからかをします https : //someapi.com/name

これは、あなたがしたNAME +テキストのように、スナックバーのテキストにされるテキストをします。

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

ディスパッチャーをらかのビルダーでラップすることをおめします。これにより、にしいをにさせてすることができます。えば

```
 return newDispatcherBuilder()
            .withSerializedJSONBody("/authenticate", Mocks.getAuthenticationResponse())
            .withSerializedJSONBody("/getUserInfo", Mocks.getUserInfo())
            .withSerializedJSONBody("/checkNotBot", Mocks.checkNotBot());
```

IdlingResource

アイドリングリソースのは、アプリケーションのネットワーキング、、アニメーションなどが`sleep()`でするのをつがなくなり、フレークさをもたらしたりテストをしたりすることにあります。はここにあります 。

`IdlingResource`インターフェイスをするには、の3つのことがです。

- **`getName()`** - アイドルのリソースのをします。
- **`isIdleNow()`** - あなたのxyzオブジェクト、などがアイドルかどうかをチェックします。
- **`registerIdleTransitionCallback`** `IdlingResource.ResourceCallback` callback - オブジェクトがアイドルにしたときにびすコールバックをします。

は、のロジックをして、アプリがアイドルであるかどうかをするがあります。これはアプリにし

ているためです。は、なをしています。オンラインののもありますが、のアプリのは、のアイドリングリソ―スのをもたらします。

ノ―ト

- アプリのコ―ドに`IdlingResources`をくいくつかのGoogleのがあります。 こんなことしないで。らはおそらくらがくをすためにそこにそれをしました。
- コ―ドをきれいにち、ののをすることはあなたです

なことをするアクティビティがあり、がみまれるまでにがかかり、フラグメントからリソ―スをつけることができないためにエスプレッソテストがするアクティビティのをするがあるそれをスピ―ドアップするため。しかし、どのようなでもシンプルにするために、のは、どのようにえるかをしています。

のアイドルのリソ―スには、2つのオブジェクトがあります。

- あなたがつけてにするのをつがあるのタグ。
- フラグメントをつけるためにされる**FragmentManager**オブジェクト。

```
/**
 * 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;
```

```
    }
}
```

あなたの`IdlingResource`かれたので、あなたはそれをどこかでうがありますか

テストクラスのをスキップし、テストケースがどのようにえるかをてみましょう

```
@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);
}
```

JUnitルールとのみわせ

これはしいことではありません。 JUnitテストルールのでアイドリングリソースをすることもできます。たとえば、VolleyがっているSDKをいくつかっていて、EspressoがそのSDKをつようにしたいとしましょう。テストケースをべたり、セットアップでそれをしたりするわりに、JUnitルールをしてのようにくことができます

```
@Rule
public final SDKIdlingRule mSdkIdlingRule = new
SDKIdlingRule(SDKInstanceHolder.getInstance());
```

これがであるので、それをのものとしてはいけません。ここのコードはすべてのものであり、デモでのみされます。

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

オンラインでUIテストの - Androidをむ https://riptutorial.com/ja/android/topic/3530/uiテストの---android

119: UIライフサイクル

Examples

メモリトリミングにデータをする

```
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);
    }
}
```

では、ここでがこっているのですか

Androidシステムは、できるだけくのメモリをクリアするようにします。したがって、あなたのがバックグラウンドに`onTrimMemory()`、のフォアグラウンドがそのをしている、Androidシステムはあなたのについて`onTrimMemory()`をびします。

しかし、それはすべてのあなたのがえなければならないというわけではありません。あなたはBundleオブジェクトにそれらをするがあります。バンドルオブジェクトは、メモリをにするがはるかにれています。バンドルでは、すべてのオブジェクトがのテキストシーケンスでされ`EXAMPLE_ARG`。のでは、`mArg`は`EXAMPLE_ARG`にされてい`EXAMPLE_ARG`。アクティビティがされると、Bundleオブジェクトからいをするのではなく、からします

オンラインでUIライフサイクルをむ https://riptutorial.com/ja/android/topic/3440/uiライフサイクル

120: VectorDrawableおよびAnimatedVectorDrawable

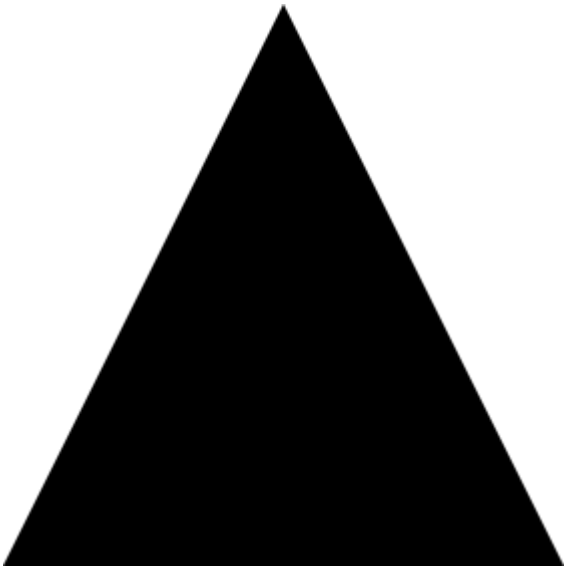
Examples

ベーシックVectorDrawable

`VectorDrawable`は、シェイプをするなくとも1つの`<path>`タグでするがあります

```
<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>
```

これはいをします



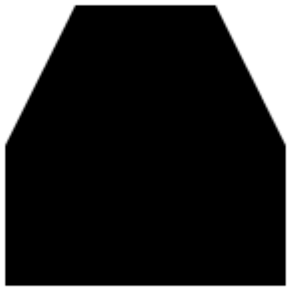
`<clip-path>`はウィンドウとしてするシェイプをし、 `<path>` `<clip-path>`シェイプにあり、りをりっているにのみ`<path>`がされるようにします。

```
<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>
```

この、 `<path>`はいをし`<clip-path>`が、 `<clip-path>`はよりさなをし、ののみがされるようにします。



タグ

`<group>`タグをすると、 `VectorDrawable`の1つまたはののスケーリング、、およびをできます。

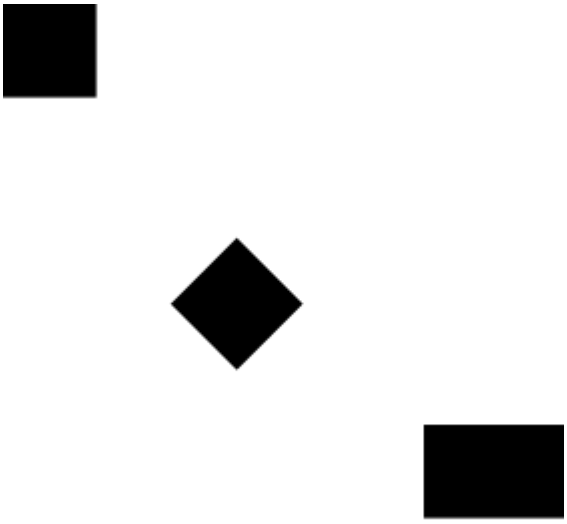
```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
android:width="24dp"
android:height="24dp"
android:viewportWidth="24.0"
android:viewportHeight="24.0">
    <path
        android:pathData="M0,0 h4 v4 h-4 z"
        android:fillColor="#FF000000"/>

    <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>
```

ののコードには、3つのの<path>タグがまれています。すべてがのです。のはされていません。 2のは<group>タグでまれており、それをして45°します。 3のは<group>タグでまれており、タグをさせてに50ばします。はのとおりです。



<group>タグにはの<path> <clip-path>タグと<clip-path>タグをめることができます。の<group>めることもできます。

Basic AnimatedVectorDrawable

AnimatedVectorDrawableはなくとも3つのコンポーネントがです。

- されるVectorDrawable
- どのプロパティをするかをするobjectAnimator
- AnimatedVectorDrawableふobjectAnimatorするVectorDrawableアニメーションをします

は、をからにえるをします。

VectorDrawable 、 ファイル 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` 、 ファイル `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"/>
```

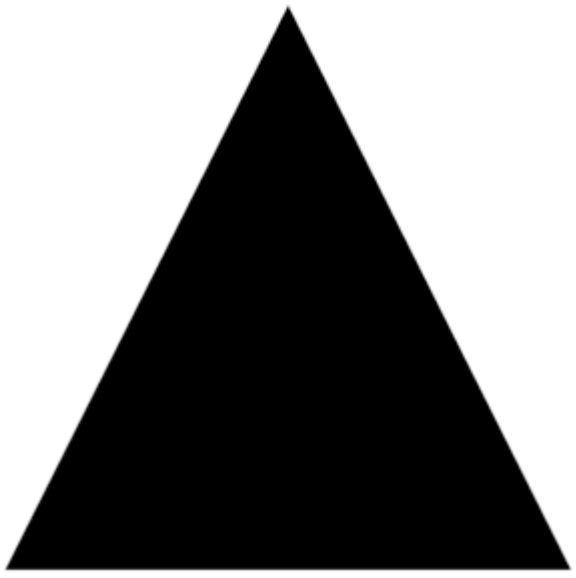
`AnimatedVectorDrawable` 、 ファイル `triangle_animated_vector.xml`

```
<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>
```

`<target>` `<path>`は、 `VectorDrawable` `<path>`とする`android:name="triangle"`がされてい`VectorDrawable` 。 `VectorDrawable`はのがまれ、 `android:name`プロパティはどのがとなっているかをするためにされます。



ストロークの

SVGストロークをすると、 Material Design Guidelinesにって、されたストロークのVector Drawableをにできます。

> したストロークのみは、システムのアイコンファミリをするためのです。、、のストロークをむ、すべてのストロークインスタンスにして2dpのをします。

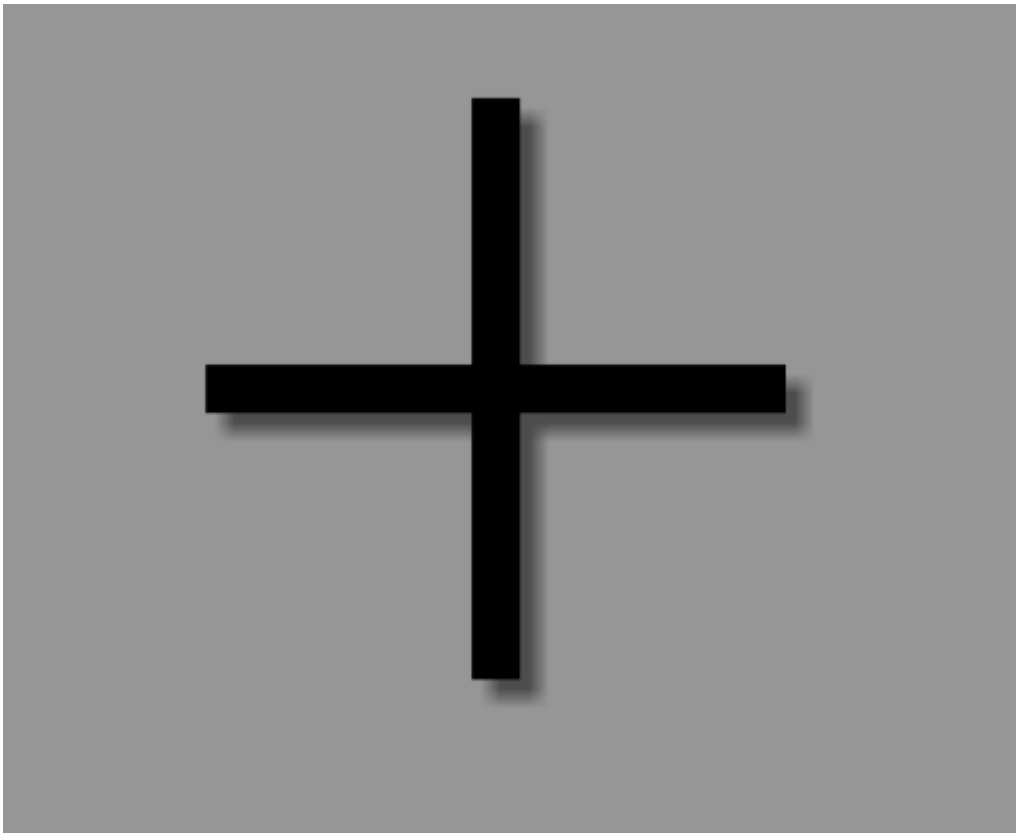
たとえば、これはストロークをして「プラス」をするです。

```
<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`はストロークのをします。
- `strokeWidth`は、ストロークのdpをしますこの、ガイドラインでされているように2dp。
- `pathData`はSVGイメージをするです
- `M12,0`は「カーソル」を12,0にする
- `V24`は`V24`をします。

のパスコマンドのについては、 SVGドキュメントとw3schoolsのこのな「SVG Path」チュートリアルをしてください 。

その、たちはこのフリルプラスサインをました



これは、なパスではなく、されたさのストロークでするため、 `AnimatedVectorDrawable`をするににです 。

AppCompatによるベクター—の

VectorDrawableのAPI 7とAnimatedVectorDrawablesのAPI 13ではいくつかのがありますまで、ベクター—の`build.gradle`にはいくつかのがあります。

```
//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'
}
```

あなたの`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" />
```

オンラインでVectorDrawableおよびAnimatedVectorDrawableをむ
https://riptutorial.com/ja/android/topic/1627/vectordrawableおよびanimatedvectordrawable

121: VideoView

Examples

VideoView Create

アクティビティでVideoViewをつけ、ビデオをします。

```
VideoView videoView = (VideoView) .findViewById(R.id.videoView);
videoView.setVideoPath(pathToVideo);
```

ビデオのをします。

```
videoView.start();
```

VideoViewをXMLレイアウトファイルでします。

```
<VideoView
    android:id="@+id/videoView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center" />
```

VideoViewをしてURLからビデオをする

```
 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;
      }
});
```

オンラインでVideoViewをむ https://riptutorial.com/ja/android/topic/8962/videoview

122: ViewFlipper

き

`ViewFlipper`は、された2つのビューをりえる`ViewAnimator`です。に1のしかされません。リクエストされた、 `ViewFlipper`はに`ViewFlipper`をので`ViewFlipper`できます。

Examples

をスライドさせる`ViewFlipper`

XMLファイル

```
<ViewFlipper
    android:id="@+id/viewflip"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:layout_weight="1"
    />
```

JAVAコード

```
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);
    }
}
```

オンラインでViewFlipperをむ https://riptutorial.com/ja/android/topic/9032/viewflipper

123: ViewPager

き

ViewPagerはレイアウトマネージャーで、ユーザーがデータのページをにめくることができます。これは、ほとんどの、ページのライフサイクルをしするためのなであるフラグメントとみわせてされます。

ViewPagerのいについてするなは、 `FragmentPagerAdapter`と`FragmentStatePagerAdapter` 2つのなるバージョンがあることです。

FragmentPagerAdapterまたはFragmentStatePagerAdapterをつ`android.app.Fragment`ネイティブフラグメントをしているは、アダプタのv13サポートライブラリバージョンつまり、`android.support.v13.app.FragmentStatePagerAdapter`をするがあります。

FragmentPagerAdapterまたはFragmentStatePagerAdapterをつ`android.support.v4.app.Fragment`サポートライブラリのフラグメントをしているは、アダプタのv4サポートライブラリバージョンつまり、 `android.support.v4.app.FragmentStatePagerAdapter`をするがあります。

Examples

フラグメントによるViewPagerの

`ViewPager`では、アクティビティにのフラグメントをすることができます。これらのフラグメントは、またはにフリップすることでナビゲートできます。 `ViewPager`は、 `PagerAdapter`をしてビューまたはフラグメントのフィードであるがあります。

ただし、 `FragmentPagerAdapter`と`FragmentStatePagerAdapter` Fragmentをするにもな2つのながあります。フラグメントをめてインスタンスするがある、インスタンスがなにして`getItem(position)`がびされます。 `getCount()`メソッドはページをします。そのため、 `ViewPager`するがあるフラグメントのをします。

`FragmentPagerAdapter`と`FragmentStatePagerAdapter`は、 `ViewPager`がするがあるフラグメントのキャッシュをします。デフォルトでは、 `ViewPager`はされているFragmentにするFragmentを3つ、とのFragmentをで3つしようとします。また、 `FragmentStatePagerAdapter`はフラグメントのをします。

のは、あなたがることができるようにあなたがそれらのなをつののリストをわりにつので、もしあなたのフラグメントは、そのをするとしていることにしてください`getItem()`メソッドは、サブクラスをするがあります`PagerAdapter`なくとも`instantiateItem()` 、 `destroyItem()`および`getItemPosition()`メソッドをオーバーライドします。

なでしたように、レイアウトにViewPagerをするだけです

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>
    <android.support.v4.view.ViewPager
        android:id="@+id/vpPager">
    </android.support.v4.view.ViewPager>
</LinearLayout>
```

に、アダプターのページにするページとフラグメントをするアダプターをします。

```
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

`android.app.Fragment`をしている、このをするがあります

```
compile 'com.android.support:support-v13:25.3.1'
```

`android.support.v4.app.Fragment`をしている、このをするがあります

```
compile 'com.android.support:support-fragment:25.3.1'
```

ViewPagerとTabLayout

`TabLayout`をしてにナビゲーションを`TabLayout`ことができます。
`TabLayout.newTab()`メソッドをしてアダプタのフラグメントのタブをできますが、このタスクには`TabLayout.setupWithViewPager()`というもう1つのでなメソッドがあります。

このメソッドは、`ViewPager`たびに`ViewPager`けられているアダプタのにってタブをおよびすることによってされます。
また、ユーザーがページをめくるたびに、するタブがされるようにコールバックをします。

レイアウトをするだけです

```
<?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>
```

に、`FragmentPagerAdapter`をし、それを`ViewPager`します。

```
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;
        }

    }
}
```

PreferenceFragmentをつViewPager

まで、 `android.support.v4.app.FragmentPagerAdapter`をすると、FragmentPagerAdapterでされる

Fragmentの1つとして`PreferenceFragment`のがげられます。

サポートv7ライブラリのバージョンには、ViewPagerとv4バージョンのFragmentPagerAdapterでする`PreferenceFragmentCompat`クラスがされました。

`PreferenceFragmentCompat`をするフラグメントの

```
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) {

    }
}
```

このフラグメントは、 `android.support.v4.app.FragmentPagerAdapter`サブクラスでできます。

```
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;
        }
    }

    // .......

}
```

ViewPagerをする

のでアプリケーションの`build.gradle`ファイルにのがされていることをしてください。

```
compile 'com.android.support:support-core-ui:25.3.0'
```

に、 `ViewPager`をアクティビティレイアウトにします。

```
<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
```

に、 `PagerAdapter`します。

```
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;
    }
}
```

に、アクティビティで`ViewPager`をします

```
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));
    }
}
```

ドットインジケータきのViewPager



たちがとするのは、 ViewPager 、 TabLayout 、およびされたデフォルトドットと2つのドロウアブルです。

まず、レイアウトに`TabLayout`をし、それを`ViewPager`するが`ViewPager`ます。たちは2つのでこれをうことができます

ViewPagerのネストされた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:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</android.support.v4.view.ViewPager>
```

この、 `TabLayout`はに`ViewPager`にされ`ViewPager`が、 `TabLayout`は`ViewPager`に`ViewPager` 、にはされません。

々の**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"/>
```

この、 `TabLayout`どこにでもくことができますが、 `TabLayout`とプログラムで`ViewPager`をする`TabLayout`が`ViewPager`ます

```
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);
```

レイアウトをしたら、たちはをするがあります。そこで、 `selected_dot.xml` 、 `default_dot.xml` 、および`tab_selector.xml` 3つのファイルをします。

---

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>
```

これで、xmlレイアウトの`TabLayout`にコードを3するだけでみます。

```
app:tabBackground="@drawable/tab_selector"
app:tabGravity="center"
app:tabIndicatorHeight="0dp"
```

セットアップ**OnPageChangeListener**

したページのをリッスンするがあるは、 `ViewPager.OnPageChangeListener`リスナーをできます。

```
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
```

```
    }
});
```

オンラインでViewPagerをむ https://riptutorial.com/ja/android/topic/692/viewpager

124: WebView

き

WebViewは、アプリケーションにWebページをするビューです。これにより、のURLをできます。

Androidマニフェストファイルにをすることをれないでください

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

Examples

WebViewのJavaScriptアラートダイアログ - させる

デフォルトでは、WebViewはJavaScriptアラートダイアログをしていません。 `alert()`はもしません。まずJavaScriptをにしてらかに..、 `WebChromeClient`をして、ページからのダイアログのリクエストをするようにするには、のようにします。

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

ここでは、 `onJsAlert`をオーバーライドしてから、スーパーインプリメンテーションをびして、のAndroidダイアログをします。カスタムスタイルのダイアログをするやログにするなど、メッセージやURLをでることもできます。

JavascriptからJavaへのAndroid

Androidのアクティビティ

```
package com.example.myapp;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class WebViewActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        WebView webView = new WebView(this);
        setContentView(webView);

       /*
        *   Note the label Android, this is used in the Javascript side of things
        *   You can of course change this.
        */
        webView.addJavascriptInterface(new JavascriptHandler(), "Android");

        webView.loadUrl("http://example.com");
    }
}
```

Java Javascript Handler

```
import android.webkit.JavascriptInterface;

public class JavascriptHandler {

    /**
     *  Key point here is the annotation @JavascriptInterface
     *
     */
    @JavascriptInterface
    public void jsCallback() {
       // Do something
    }

    @JavascriptInterface
    public void jsCallbackTwo(String dummyData) {
       // Do something
    }
}
```

ウェブペ―ジ、**Javascript**コ―ル

```
<script>
...
Android.jsCallback();
...
Android.jsCallback('hello test');
...
</script>
```

なヒント

なデ―タをす、なはJSONをすることです。

```
Android.jsCallback('{ "fake-var" : "fake-value", "fake-array" : [0,1,2] }');
```

Androidでは、おにりのJSONパ―サ―をしてくださいJSONObject

Javaから**Javascript**への

な

```
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!')");
   }
}
```

ダイヤラのをく

Webペーージにがまれているは、のダイヤラをしてをかけることができます。このコードは、telでまるURLをチェックしてから、ダイヤラをくであり、クリックしたにをかけることができます。

```
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;
}
```

コンソールメッセージのまたはリモートデバッグによる**WebView**のトラブルシューティング

webviewコンソールメッセージをlogcatにする

するにはconsole Webページからのメッセージを、あなたはきすることができonConsoleMessageで
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;
    }
}
```

あなたのやにそれをしてください

```
webView.setWebChromeClient(new ChromeClient());
```

したがって、このサンプルページ

```
<html>
<head>
    <script type="text/javascript">
        console.log('test message');
    </script>
</head>
<body>
</body>
</html>
```

logcatにログ 'test message'をきみます

    WebViewテストメッセージsample.html4

console.info()、 console.warn()、 console.error()もchrome-clientでサポートされています。

ChromeでAndroidデバッグをリモートデバッグする

あなたはデスクトップChromeからリモートデバッグWebviewベースのアプリケーションをできます。

AndroidでUSBデバッグをにする

Androidで、[]をき、[オプション]セクションをつけて、USBデバッグをにします。

Androidをしてする

Chromeページののページをく chrome// inspect /devices

Inspect Devicesダイアログでデバイスをし、 **inspect**をします。 ChromeのDevToolsのしいインスタンスがマシンにされます。

よりなガイドラインとデベロッパーツールのは、でつけることができますdevelopers.google.com

ローカルファイルをく/ Webviewでコンテンツをする

Layout.xml

```
            <WebView
            android:id="@+id/WebViewToDisplay"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:fadeScrollbars="false" />
```

WebViewToDisplayにデータをロードする

```
        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");
```

オンラインでWebViewをむ https://riptutorial.com/ja/android/topic/153/webview

125: Wi-Fi

Examples

WEPとする

このでは、SSIDとパスワードをして、WEPをしてWi-Fiアクセスポイントにします。

```
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;
    }
}
```

WPA2でする

このでは、WPA2をしてWi-Fiアクセスポイントにします。

```
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;
  }
}
```

アクセスポイントをスキャンする

このでは、なアクセスポイントとアドホックネットワークをスキャンします。 `btnScan`は、`WifiManager.startScan()`メソッドによってされたスキャンをアクティブにします。スキャン、`WifiManager`は`SCAN_RESULTS_AVAILABLE_ACTION`インテントをびし、 `WifiScanReceiver`クラスがスキャンをします。は`TextView`されます。

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

アクセス

*AndroidManifest.xml*でのをするがあります。

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

`WifiManager.getScanResults()`をびすには`android.permission.ACCESS_WIFI_STATE`がです。`android.permission.CHANGE_WIFI_STATE`をしないと、 `WifiManager.startScan()`スキャンをできません。

プロジェクトをAPIレベル23Android 6.0にコンパイルするは、`android.permission.ACCESS_FINE_LOCATION`または`android.permission.ACCESS_COARSE_LOCATION`するがあ

ります。さらに、あなたのメインアクティビティの`onCreate`メソッドのように、アクセスをする
があります。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    String[] PERMS_INITIAL={
            Manifest.permission.ACCESS_FINE_LOCATION,
    };
    ActivityCompat.requestPermissions(this, PERMS_INITIAL, 127);
}
```

オンラインでWi-Fiをむ https://riptutorial.com/ja/android/topic/3288/wi-fi

126: XMPPログインとチャットのな

Examples

XMPPログインとチャットのな

あなたのシステムやサーバにopenfireやチャットサーバをインストールしてください。はこちらをクリックしてください。

androidプロジェクトをし、これらのライブラリをgradleにします

```
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'
```

に、xmppから1つのxmppクラスをします。

```
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();
    }
}

}
```

に、これをしてください

```
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();
    }

}
```

マニフェストファイルにインター—ネットがされていることをしてください。

オンラインでXMPPログインとチャットのなをむ
https://riptutorial.com/ja/android/topic/6747/xmppログインとチャットのな

127: Xposed

Examples

Xposedモジュールの

Xposedは、のアプリケーションのメソッドびしをフックできるフレームワークです。 APKをコンパイルしてすると、なにコマンドを**/**できます。ただし、で***APK***をコンパイル**/**するがあり、パッケージをすることしかできません。 Xposedをすると、メソッドのにのコードをしたり、メソッドをにきえることができます。ながら、Xootは***rooted***デバイスにのみインストールできます。のアプリやコア***Android***システムのをしたいや、APKのコンパイル、コンパイル、のをきたいは、ず***Xposed***をしてください。

まず、Android Studioにアクティビティのないアプリをします。

に、 *build.gradle*にのコードをするがあります。

```
repositories {
    jcenter();
}
```

その、のをします。

```
provided 'de.robv.android.xposed:api:82'
provided 'de.robv.android.xposed:api:82:sources'
```

これで、 *AndroidManifest.xml*にあるアプリケーションタグのにこれらのタグをして、 *Xposed*があなたのモジュールをできるようにしなければなりません

```
<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" />
```

に**82**をのXposedバージョンにきえてください。

メソッドをフックする

`IXposedHookLoadPackage`をするしいクラスをし、 `handleLoadPackage`メソッドをし`handleLoadPackage`。

```
public class MultiPatcher implements IXposedHookLoadPackage
{
    @Override
    public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
    {

    }
}
```

このメソッドので、フックするアプリケーションのパッケージの`loadPackageParam.packageName`をします。

```
@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }
}
```

コードがされる、またはに、メソッドをフックしてすることができます

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

オンラインでXposedをむ https://riptutorial.com/ja/android/topic/4627/xposed

128: Youtube-API

1. まず、のリンクからのjarをダウンロードするがありま す。 https://developers.google.com/youtube/android/player/downloads/
2. このjarをプロジェクトにめるがあります。このjarファイルをlibsフォルダにコピーペーストして、それをgradleファイルのにすることをれないでください。{コンパイルファイル 'libs / YouTubeAndroidPlayerApi.jar'}
3. あなたは、YouTube APIにアクセスするためのAPIキーがです。 https://developers.google.com/youtube/android/player/registerにアクセスしてAPIキーをしてください。
4. プロジェクトをしてビルドします。 YoutubeAndroidPlayerApiをするができましたyoutube ビデオをするには、YouTubeでできるように、それにするビデオIDがです。たとえば https : //www.youtube.com/watch?v= B08iLAtS3AQの、B08iLAtS3AQはyoutubeでするがあるIDです。

Examples

StandAlonePlayerActivityをする

1. スタンドアロンプレーヤーの

```
     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);
```

YouTubeBaseActivityをするアクティビティ

```
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());
    }

}
```

ポートレートの**YoutubePlayerFragment Activty**

のコードは、なYoutubePlayerFragmentをしています。アクティビティのレイアウトはポートレートモードでロックされ、がわるか、ユーザがYoutubePlayerでをクリックすると、YoutubePlayerがをたしてlansscapeにわります。 YoutubePlayerFragmentは、Youtubeライブラ

リによってされるアクティビティをするはありません。 YoutubePlayerをするには、YouTubePlayer.OnInitializedListenerをするがあります。たちのアクティビティのクラスはのりです

```
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をアクティビティのレイアウトxamlにすることができます。

```
<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>
```

に、マニフェストファイルにのをアクティビティのタグにするがあります

```
android:configChanges="keyboardHidden|orientation|screenSize"
android:screenOrientation="portrait"
```

YouTube Player API

Android APIキーをする

まず、java keytoolをしてマシンにSHA-1をするがあります。 cmd / terminalでのコマンドをして

、SHA-1をします。

```
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);
    }
}
```

`Config.java`ファイルをします。このファイルには、GoogleコンソールAPIのキーとYouTubeのIDがまれています

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ファイル

```
<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" />
```

AndroidでのYouTubeデータAPIの

このでは、AndroidのYouTube Data APIをしてプレイリストデータをするをします。

SHA-1

まず、あなたのマシンにSHA-1をするがあります。それをするためのさまざまながあります。 このQAにされているをすることができます。

AndroidGoogle APIコンソールと**YouTube**キー

SHA-1のをしたら、Google APIコンソールをいてプロジェクトをします。 このページにし、そのSHA-1キーをしてプロジェクトをし、YouTube Data APIをにします。はをるでしょう。このキーは、Androidからのリクエストのやデータのにされます。

け

YouTube Data APIのは、のをGradleファイルにするがあります。

```
compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
```

YouTubeのネイティブクライアントをしてリクエストをするには、Gradleにのをするがあります。

```
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:+'
```

をけるために、のもGradleにするがあります。

```
configurations.all {
    resolutionStrategy.force 'com.google.code.findbugs:jsr305:3.0.2'
}
```

に、 *gradle.build*がにどのようにえるかがされています。

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:+'
}
```

、Javaのがます。々がするのでHttpTransportネットワーキングとするためにGsonFactory POJOにJSONにするために、々はすべてのをするためにのライブラリをとしません。

ここで、プレイリストIDをしてYouTube APIでプレイリストをするをしたいといます。このタスクでは、 AsyncTaskをしAsyncTask。パラメータをリクエストするとフローをするをするには、YouTube Data APIをください。

```
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;
    }
}
```

のタスクは、YouTube SDKのビルドインクラスであるPlaylistListResponseインスタンスをします。フィールドがすべてあるので、POJOをでするはありません。

に、 MainActivityではのことをうがあります。

```
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);
    }
}
```

オンラインでYoutube-APIをむ https://riptutorial.com/ja/android/topic/7587/youtube-api

129: アイコンフォントをしてAndroidのパフォーマンスをさせる

アイコンフォントは、のわりにシンボルをつのフォントタイプとています。ほとんどの、アプリケーションでできます。

らです

- フレキシブル
- スケーラブル
- ベクトル
-
-
- アクセスな

サイズへの

アンドロイドデバイスにさまざまなサイズのをきすと、アプリケーションごとにがかかり、あたり30kBのアセットサイズがされます。フォントファイル.ttfに36のアイコンをすると、わずか9kBのコストがかかります。さまざまなの36のファイルをするは、1000kBになるでしょう。これは、アイコンフォントをすることでできるスペースです。

アイコンフォントの。

- アイコンフォントはナビゲーションドロワーでできます。タイトルをせずにメニューファイルをすることはできないため、ナビゲーションビューでメニューのアイコンとしてすることはできません。したがって、これらのアイコンのリソースとしてsvgファイルをすることをおめします。
- アクションボタンではアイコンフォントをできません。 `setText()` をたないためです。
- xmlからフォントをすることはできません。それらはjavaファイルをしてするがあります。それのは、なビューをし、このポストでされたビューをするがあります

Examples

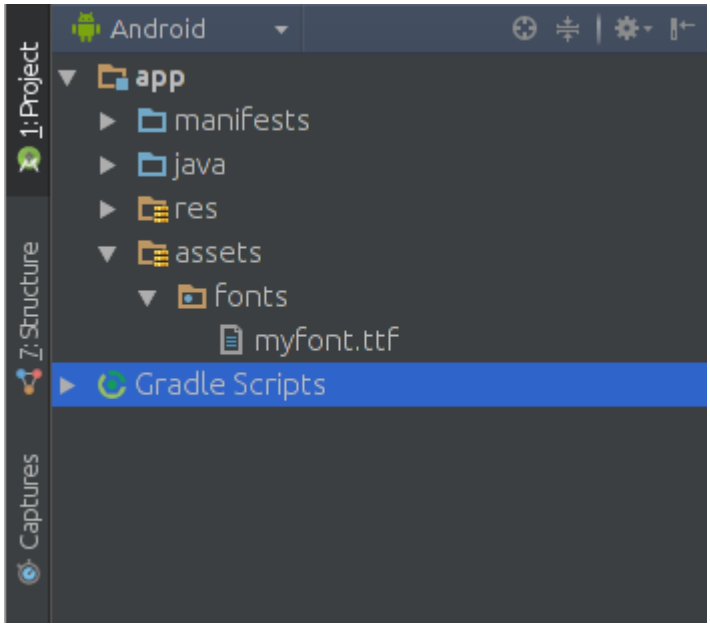
アイコンフォントをする

アイコンフォントをするには、のにってください

- プロジェクトにフォントファイルをする

  icomoonなどのオンラインWebサイトからフォントアイコンファイルをして、なアイコンのSVGファイルをアップロードし、したアイコンフォントをダウンロードすることができま

す。に、 *.ttf*フォントファイルをassetsフォルダに*fonts*というのフォルダにれます。



- ヘルパークラスをする

  に、フォントのコードをりさないように、のヘルパークラスをします。

```
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);
  }
}
```

  アセットからフォントをするために`Typeface`クラスをすることができます。このようにして、タイプフェイスをさまざまなビューボタンなどにできます。

```
Button button=(Button) findViewById(R.id.button);
Typeface iconFont=FontManager.getTypeface(getApplicationContext());
button.setTypeface(iconFont);
```

  、ボタンのがしくされたアイコンフォントにされました。

- あなたがむアイコンをぶ

  アイコンフォントにされている*styles.css*ファイルをきます。そこにあなたのアイコンのUnicodeでスタイルをつけることができます

```
.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";
}
```

このリソースファイルはとしてし、のアイコンにけられたUnicodeをがめるにマップします。に、リソースをのようにします。

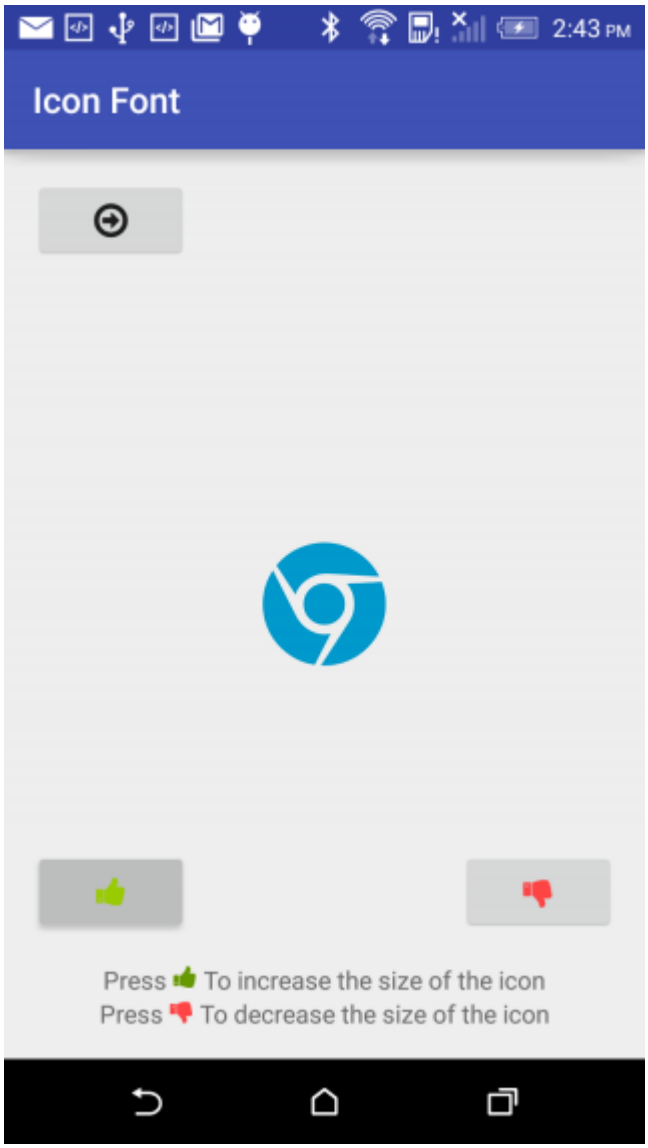
```
<resources>
  <! – Icon Fonts -->
  <string name="icon_arrow_circle_down">&#xe001; </string>
  <string name="icon_arrow_circle_left">&#xe002; </string>
  <string name="icon_arrow_circle-o_down">&#xe003; </string>
  <string name="icon_arrow_circle_o_left">&#xe004; </string>
</resources>
```

- コードのアイコンをする

  さて、のように、あなたのフォントをさまざまなビューでうことができます

  ```
  button.setText(getString(R.string.icon_arrow_circle_left))
  ```

  アイコンフォントをしてボタンテキストビューをすることもできます。



アイコンフォントきTabLayout

```
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;
}

}
```

- このでは、myfont.ttfはAssetsフォルダにあります。 アセットのフォルダ
- あなたのアクティビティクラス

```
//..
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);
    //..
```

オンラインでアイコンフォントをしてAndroidのパフォーマンスをさせるをむ
https://riptutorial.com/ja/android/topic/3642/アイコンフォントをしてandroidのパフォーマンスをさせる

130: アカウントとAccountManager

Examples

カスタムアカウント/について

のは、コンセプトとなのをでカバーしています。

1. ユーザーからのをしますはしたログインから
2. サーバーでをするカスタムをする
3. デバイスにをします。

AbstractAccountAuthenticatorをする にのとにされる

```
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.
  }
}
```

サービスをします *Account Manager*フレームワークは、サービスインタフェースをして *AbstractAccountAuthenticator*にします

```
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();
    }
}
```

オーセンティケータ**XML** アカウントマネージャフレームワークがです。これは*Android*の*[] - > [*アカウント*]*にされます

```
<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" />
```

AndroidManifest.xmlの のすべてのをまとめて*AccountManager*をしてプログラムでできるようにする

```
<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>
```

のでは、このをするについてします。

オンラインでアカウントとAccountManagerをむ https://riptutorial.com/ja/android/topic/7003/アカウントとaccountmanager

131: アクティビティ

き

アクティビティは、ユーザーインターフェイスUIをつのをします。 Android Appには、のアクティビティがあるがあります。たとえば、メールアプリケーションには、すべてのメールをするアクティビティ、メールコンテンツをするのアクティビティ、さらにはしいメールをするのアクティビティがあります。 Appのすべてのアクティビティがして、なユーザーエクスペリエンスをします。

- void onCreateBundle savedInstanceState//アクティビティのにびされます。
- void onPostCreateBundle savedInstanceState//アクティビティのがしたときonStartおよびonRestoreInstanceStateBundleがびされたにびされます。
- void onStart// onCreateBundleのにびされるか、またはonRestartのにアクティビティがしたときにびされますが、ユーザにびされます。
- void onResume//あなたのアクティビティがユーザとやりりをするために、onRestoreInstanceStateBundle、onRestart、またはonPauseのにびされます。
- void onPostResume//アクティビティのレジュームがしたときonResumeがびされたにびされます。
- void onRestart//のアクティビティがユーザにされているユーザがそれまでナビゲートしているときにonStopのにびされます。
- void onPause//アクティビティがバックグラウンドにっても、まだまだされていないときに、アクティビティのライフサイクルのとしてびされます。
- void onStop//ユーザーがえなくなったときにびされます。
- void onDestroy//アクティビティがされるになクリーンアップをします。
- void onNewIntentIntent intent//パッケージのlaunchModeを "singleTop"にするアクティビティ、またはstartActivityIntentをびすときにクライアントがFLAG_ACTIVITY_SINGLE_TOPフラグをしたにびされます。
- void onSaveInstanceStateBundle outState// onCreateBundleまたはonRestoreInstanceStateBundleでをできるように、アクティビティからインスタンスごとのをするためにびされますこのメソッドによってされたBundleは、 。
- void onRestoreInstanceStateBundle savedInstanceState//このメソッドは、アクティビティがにされたからされているときにonStartのにびされます。

パラメーター

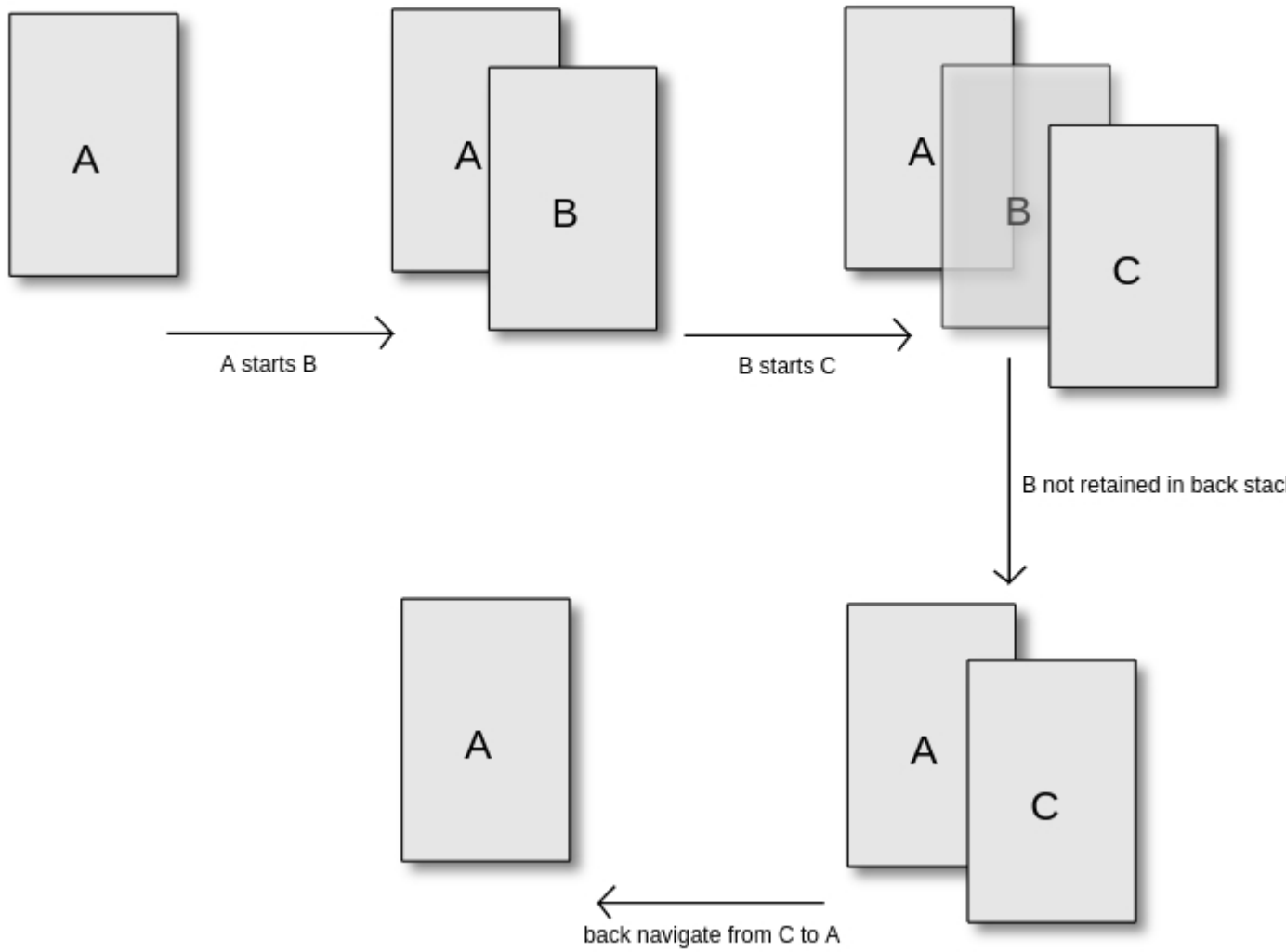
パラメータ	
インテント	startActivityとにしてアクティビティをすることができます。
バンドル	Stringキーからさまざまなへのマッピング。
コンテキスト	アプリケーションにするグローバルなへのインタフェース。

アクティビティとは、をかける、をる、メールをする、をするなど、かをするためにユーザーができるをするアプリケーションコンポーネントです。アクティビティには、そのユーザーインターフェイスをするウィンドウがされます。ウィンドウは、をりつぶしますが、よりもさく、のウィンドウのにいているがあります。

Examples

バックスタックからアクティビティをする

オープンできるアクティビティBがあり、さらにくのアクティビティをすることができます。しかし、ユーザは、タスクアクティビティをナビゲートするときにそれにするべきではありません。



もなは、 `AndroidManifest.xml` `<activity>`タグにして`noHistory`を`true`にすることです。

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

のアクティビティをするに`B`が`finish()`びすも、このじがコードからです。

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

`noHistory`フラグのなは、"スプラッシュスクリーン"またはログインアクティビティです。

Androidのアクティビティライフサイクルの

MainActivityをつアプリケーションが、ボタンクリックをしてのアクティビティをびすことができるとします。

```
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);
    } }
```

そして

```
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");
    } }
```

アプリがにされるとき
D / MainActivityMainActivityからonCreateをびす
D / MainActivityMainActivityからonStartをびす
D / MainActivityMainActivityからonResumeをびす
びされる

がスリ―プすると
081103.142 D / MainActivityMainActivityからonPauseをびす
081103.192 D / MainActivityMainActivityからonStopをびす
びされます。そしてもうがめると
081155.922 D / MainActivityMainActivityからonRestartをびす
081155.962 D / MainActivityMainActivityからonStartをびす
081155.962 D / MainActivityMainActivityからonResumeをびす
びされる

ケ―ス1メインアクティビティからアクティビティがびされたとき
D / MainActivityのアクティビティをびす
D / MainActivityMainActivityからonPauseをびす
D / NextActivityのアクティビティからonCreateをびす

D / NextActivityのアクティビティからonStartをびす
D / NextActivityのアクティビティからonResumeをびす
D / MainActivityMainActivityからonStopをびす

るボタンをしてのアクティビティからメインアクティビティにるとき
D / NextActivityのアクティビティからonPauseをびす
D / MainActivityMainActivityからonRestartをびす
D / MainActivityMainActivityからonStartをびす
D / MainActivityMainActivityからonResumeをびす
D / NextActivityのアクティビティからonStopをびす
D / NextActivityのアクティビティからonDestroyをびす

ケース2アクティビティがになオーバービューボタンがされたまたはアプリがバックグラウンドにし、のアプリがにそれをす
D / MainActivityMainActivityからonPauseをびす
D / MainActivityMainActivityからonStopをびす
アプリがフォアグラウンドにってユーザーのをけれるができたら、
D / MainActivityMainActivityからonRestartをびす
D / MainActivityMainActivityからonStartをびす
D / MainActivityMainActivityからonResumeをびす
びされる

ケース3のインテントをするためにアクティビティがびされ、ユーザーがをったとき。たとえば、ボタンがされ、ユーザーがされたアプリケーションのリストからアプリをするがある
D / MainActivityMainActivityからonPauseをびす

アクティビティはされますが、アクティブではありません。がし、アプリがアクティブな
D / MainActivityMainActivityからonResumeをびす
とばれる

ケース4
アプリケーションがバックグラウンドでのフォアグラウンドアプリのリソースをするために されると、アプリケーションがするに*onPause* ハニカムデバイスまたは*onStop* ハニカムデバイスがにびされます。

onCreateとonDestroyは、アプリケーションがされるたびにびされます。しかし、onPause、onStop、onRestart、onStart、onResumeは、ライフサイクルにもびされることがあります。

アクティビティ**launchMode**

モードは、タスクのまたはのアクティビティのをします。
なモードはのとおりです。

-
- シングルトップ

- シングルタスク
- singleInstance

これは、 `<activity/>`のandroid manifestで`android:launchMode`としてするがあります。

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

デフォルト。このモードがされている、にしいごとにしいアクティビティがされます。したがって、じタイプのくのアクティビティをすることはです。しいアクティビティがタスクのにされます。なるアンドロイドバージョンにはいくつかのいがあります。アクティビティがのアプリケーションからする、アンドロイド<= 4.4ではアプリケーションとじタスクにされますが、> = 5.0のしいタスクがされます。

シングルトップ

このモードは、 `standard`とほぼじです。 singleTopアクティビティのくのインスタンスをできました。いは、のスタックのにアクティビティのインスタンスがすでにする、しいインスタンスをするわりに`onNewIntent()`がびされるです。

SingleTask

このモードのアクティビティーは、システムに 1つのインスタンスのみをつことができます。アクティビティーのタスクがされますしない。それのは、アクティビティのあるタスクがにし、`onNewIntent`がびされます。

SingleInstance

このモードは`singleTask`とてい`singleTask`。 いは、 `singleInstance`アクティビティをするタスクはこのアクティビティのみを`singleInstance`ことができ、それのアクティビティをたないことです。`singleInstance`アクティビティがのアクティビティをすると、そのアクティビティをするためのしいタスクがされます。

setContentViewをしたUIの

アクティビティクラスは、 `setContentView`てUIをできるウィンドウをします。
3つの`setContentView`メソッドがあります。

- `setContentView(int layoutResID)` - レイアウトリソースからアクティビティコンテンツをしま

す。

- `setContentView(View view)` - アクティビティのをなビューにします。
- `setContentView(View view, ViewGroup.LayoutParams params)` - された`setContentView(View view, ViewGroup.LayoutParams params)`、アクティビティのをなビューにします。

`setContentView`がびされると、このビューはアクティビティのビューにされます。それがなビューになるがあります。

リソースファイルからコンテンツをする

ビューをつリソースファイルこのではmain.xmlをします。

```
<?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>
```

アクティビティのコンテンツとしてします。

```
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);
    }
}
```

コンテンツをなビューにする

```
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);
    }
}
```

のアクティビティスタックをクリアし、しいアクティビティをする

のアクティビティスタックをクリアしてしいアクティビティをするたとえば、アプリからログアウトしてアクティビティでログをする、2つのアプロ―チがあるようです。

1. タ―ゲット**API> = 16**

アクティビティから`finishAffinity()`をびす

2. タ―ゲット**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();
```

リクル―トからされたアプリケ―ションの

まずAndroidManifest.xmlにExitActivityをします

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

その、ExitActivityクラス

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

のためののナビゲーション

のナビゲーションは、Manifest.xmlの`android:parentActivityName=""`をアクティビティタグにすることで、アンドロイドでされます。にこのタグをすると、アクティビティのアクティビティについてシステムにします。

どうしたの

```
<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>
```

HomeActivityのツールバーのをクリックすると、アクティビティにります。

Javaコード

ここでは、このになJavaコードをします。

```
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.
}
}
```

このコードをすると、るボタンをすとMainActivityにります。のナビゲーションのをさらにめるために、はドキュメントをむことをおめします

オーバーライドすることで、にじてこのをカスタマイズすることができます

```
@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);
}
```

シンプルハック

これはなハックで、がバックスタックにある、にアクティビティーにナビゲートするためにされます。 idが`android.R.id.home`としいに`onBackPressed()`びす

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

オンラインでアクティビティをむ https://riptutorial.com/ja/android/topic/1481/アクティビティ

132: アニメーションのAlertDialogボックスき

Animated Alert Dialog Fadein、Slideleft、Slidetop、SlideBottom、Sliderright、Fall、Newspager、Fliph、Flipv、RotateBottom、RotateLeft、Slit、Shake、Sidefillのようなダイアログボックスでアニメーションをします。アプリケーションな..

Examples

アニメーションダイアログのにコードをく...

```
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();
    }
}
```

のをbuild.gradleにして、NifyBuilderCustomView

build.gradle

```
 dependencies {

     compile 'com.nineoldandroids:library:2.4.0'

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

 }
```

リンク https : //github.com/sd6352051/NiftyDialogEffects

オンラインでアニメーションのAlertDialogボックスをむ
https://riptutorial.com/ja/android/topic/10654/アニメーションのalertdialogボックス

133: アニメーター

Examples

ImageViewのアニメーションをる

resフォルダのに "anim"というのしいフォルダをしてアニメーションリソースをし、そのフォルダにします。

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" />
```

Landingとばれるのアクティビティをする

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>
```

Landing.javaでビューをアニメーションする

```
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);
    }
```

フェードイン/アウトアニメーション

ビューがゆっくりとフェードインまたはビューになるようにするには、 `ObjectAnimator`し `ObjectAnimator`。 のコードにすように、 `.setDuration(millis)`ミリをしてをします。ここで、 `millis`パラメータはアニメーションのミリです。のコードでは、ビューは500ミリ、すなわち1/2にわたってフェードイン/フェードします。 `ObjectAnimator`のアニメーションをするには、 `.start()`びします。 アニメーションがすると、 `onAnimationEnd(Animator animation)`がびされます。ビューのを`View.GONE`または`View.VISIBLE`にするには、 `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();
}
```

TransitionDrawableアニメーション

このでは、2つののみをむビューのトランザクションをしますループごとにトランザクションのに1と2のレイヤーのにのをにできます

- `res/values/arrays.xml`をする

```
<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`は、 `int` 、 `float`などののののをアニメーションするなをします。

それをするのはのとおりです。

1. を`min`から`max`アニメーションする`ValueAnimator`をします。
2. されたアニメーションをする`UpdateListener`をしますこれは`getAnimatedValue()`できます。

ValueAnimatorをするは2つあります。

サンプルコードは、アニメーションfloatから20fに40fで250ms

1. xmlから /res/animator/く

```
<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. コードから

```
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のサブクラスであるValueAnimatorのプロパティにしたをするをつtarget View。

ちょうどのようにValueAnimator 、あなたがすることができ、2つのがありますObjectAnimator

サンプルコードは、アニメーションalphaのViewから0.4fする0.2fで250ms

1. xmlからそれを/res/animatorれて

```
<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. コードから

```
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`は、 `View`プロパティをアニメーションするためのされたされたです。

すべてののView`は、 `animate()`メソッドでな`ViewPropertyAnimator`オブジェクトがあります。これをして、なびしでのプロパティをにアニメーションできます。 `ViewPropertyAnimator`すべてののメソッドは、 `ViewPropertyAnimator`がアニメーションするのパラメータのターゲットをします。

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

`ViewPropertyAnimator`オブジェクトで`start()`を`ViewPropertyAnimator`ことはではありません。そうしないと、なのアニメーションパスでアニメーションのをするだけのプラットフォームになります。にcall `start()`びして`start()`う、アニメーションがすぐにされることをしています。

ビューのアニメーションのとりたたみ

```
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);
        }
    }
```

オンラインでアニメーターをむ https://riptutorial.com/ja/android/topic/1829/アニメーター

134: アプリ

Examples

なアプリ

は、ゲーム、ゲームの、パワーアップなどできるです。

このでは、4つのなる `"item1", "item2", "item3", "item4"`ます。

のステップ

1. アプリライブラリをプロジェクトにしますAIDLファイル。
2. `AndroidManifest.xml`ファイルになをします。
3. されたapkをGoogle Developers Consoleにデプロイします。
4. あなたのをしてください。
5. コードをします。
6. アプリのテストオプション。

ステップ1

まず、AIDLファイルをGoogleドキュメントのここでにされているようにプロジェクトにするがありますます 。

`IInAppBillingService.aidl`は、In-app Billing Version 3サービスへのインターフェイスをする Android Interface Definition Language `IInAppBillingService.aidl` ファイルです。このインタフェースをして、IPCメソッドびしをびしてをいます。

ステップ2

AIDLファイルをした、 `AndroidManifest.xml` BILLINGをします。

```
<!-- Required permission for implementing In-app Billing -->
<uses-permission android:name="com.android.vending.BILLING" />
```

ステップ3

きapkをし、Google Developers Consoleにアップロードします。これは、そこにあるアプリのをするためにです。

ステップ4

なるIDをつすべてのをし、それぞれのIDにをします。には2ありますとサブスクリプション。にべたように、々は4つのなる `"item1", "item2", "item3", "item4"` です。

ステップ5

のすべてのをしたら、コードをのアクティビティでするがいました。

な

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

ステップ6

コードをしたら、apkをベータ/アルファチャンネルにデプロイしてテストし、のユーザーがコードをテストできるようにします。ただし、テストモードでは、のアプリはできません。すべてのがにになるように、まずアプリ/ゲームをPlayストアにするがあります。

アプリのテストにするはこちらをごください 。

In-App v3ライブラリ

ステップ1まず、アプリのをするには、の2つのにいます。

1.をしてライブラリをします。

```
repositories {
           mavenCentral()
       }
       dependencies {
          compile 'com.anjlab.android.iab.v3:library:1.0.+'
       }
```

2.マニフェストファイルにをします。

```
<uses-permission android:name="com.android.vending.BILLING" />
```

ステップ2プロセッサをする

```
BillingProcessor bp = new BillingProcessor(this, "YOUR LICENSE KEY FROM GOOGLE PLAY CONSOLE
HERE", this);
```

Billing HandlerBillingProcessor.IBillingHandlerをし、4つのメソッドがまれています。 onBillingInitialized; b。 onProductPurchasedString productId、TransactionDetails detailsこれは、がしたにされるアクションをするがあるです。 onBillingErrorint errorCode、Throwable errorプロセスにしたエラーをします。d。 onPurchaseHistoryRestoredアプリののリストア

ステップ3をする。

のをするには

```
bp.purchase(YOUR_ACTIVITY, "YOUR PRODUCT ID FROM GOOGLE PLAY CONSOLE HERE");
```

をするには

```
bp.subscribe(YOUR_ACTIVITY, "YOUR SUBSCRIPTION ID FROM GOOGLE PLAY CONSOLE HERE");
```

ステップ4をする。

をするには、にconsumePurchaseメソッドをびします。

bp.consumePurchase "Google Playのコンソーシアムからのあなたの ID";

アプリのgithubにするのメソッドについては

オンラインでアプリをむ https://riptutorial.com/ja/android/topic/2843/アプリ

135: アラートダイアログの

き

このトピックでは、 `AlertDialog`のについて`AlertDialog`します。

Examples

クリックなリンクをむダイアログ

クリックしてくことができるリンクをむダイアログをするには、のコードをします。

```
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
```

オンラインでアラートダイアログのをむ https://riptutorial.com/ja/android/topic/10163/アラートダイアログの

136: アンドロイドのGoogleサインイン

き

このトピックは、Googleログインをする、アンドロイドアプリをするについています

Examples

あなたのプロジェクトにGoogle Authの。 ファイルをする

まずサインインのファイルをします

のリンクをく

[https://developers.google.com/identity/sign-in/android/start-integrating] [1]

getをクリックすると、ファイルがされます。

- アプリケーションとパッケージをし、サービスのとをクリックします
- SHA1 Google SIGNINをにしてファイルをする

ファイルをダウンロードし、プロジェクトのapp /フォルダにします

1. をプロジェクトレベルのbuild.gradleにします。

   classpath 'com.google.gmsgoogle-services3.0.0'

2. アプリレベルのbuild.gradleにプラグインをします

   apply plugin 'com.google.gms.google-services'

3. このをアプリケーションのgradleファイルにする

   {compile 'com.google.android.gmsplay-services-auth9.8.0'}

コードGoogle SignIn

- ログインアクティビティのonCreateメソッドで、アプリでなユーザーデータをリクエストするようにGoogle Sign-Inをします。

```
 GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestEmail()
        .build();
```

- Google Sign-In APIとしたオプションにアクセスできるGoogleApiClientオブジェクトをしま

す。

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
        .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
        .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
        .build();
```

- ユーザーがGoogleのサインインボタンをクリックすると、このがびされます。

```
 private void signIn() {
 Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
 startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

- OnActivityResultをしてをします。

```
 @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);
    }
}
```

- のステップをしてユーザーデータをする

```
 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);
 }
}
```

オンラインでアンドロイドのGoogleサインインをむ https://riptutorial.com/ja/android/topic/9960/アンドロイドのgoogleサインイン

137: アンドロイドのORMLite

Examples

Android OrmLite over SQLiteの

ORMLiteは、よりくのORMパッケージのさとオーバーヘッドをしながら、Javaオブジェクトを SQLデータベースにするためのシンプルでなをするオブジェクトリレーショナルマッピングパッケージです。

AndroidのOrmLiteは、すぐにできるサポートされているSQLiteデータベースでされています。これは、SQLiteにアクセスするためのAPIへのびしをいます。

グレイドセットアップ

めるには、パッケージをビルドgradleにめるがあります。

```
 // https://mvnrepository.com/artifact/com.j256.ormlite/ormlite-android
compile group: 'com.j256.ormlite', name: 'ormlite-android', version: '5.0'
POJO configuration
```

に、POJOをデータベースにするようにするがあります。にするがあります。

- @DatabaseTableをクラスのにします。 @Entityをすることもできます。
- するフィールドのに@DatabaseFieldアノテーションをします。 @Columnなどをすることもできます。
- なくともパッケージのをつクラスに、のないコンストラクタをします。

```
@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;
    }
}
```

のでは、4つのフィールドをつ1つのテーブルform_modelがあります。

idフィールドはインデックスです。

usernameはデータベースのインデックスです。

のについては、 ドキュメントをしてください 。

データベースヘルパー

するには、OrmLiteSqliteOpenHelperクラスをするデータベースヘルパークラスをするがあります。

このクラスは、アプリケーションのインストールにデータベースをおよびアップグレードし、のクラスでされるDAOクラスをすることもできます。

DAOはデータアクセスオブジェクトので、すべてのスクラムをし、のクラスのにしています。

ヘルパークラスは、の2つのメソッドをするがあります。

- onCreateSQLiteDatabase sqliteDatabase、ConnectionSource connectionSource;

  onCreateは、アプリケーションがにインストールされたときにデータベースをします。

- onUpgradeSQLiteDatabaseデータベース、ConnectionSourceソース、int oldVersion、int newVersion;

  onUpgradeは、アプリケーションをしいバージョンにアップグレードするときにデータベーステーブルのアップグレードをします

データベースヘルパークラスの

```
  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;
        }
    }
```

オブジェクトをSQLiteにする

に、オブジェクトをデータベースにするクラス。

```
     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;
    }
}
```

のクラスのコンストラクタでのDOAのアクセサは、のようにされます。

```
 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;
}
```

オンラインでアンドロイドのORMLiteをむ https://riptutorial.com/ja/android/topic/7571/アンドロイドのormlite

138: アンドロイドのZipファイル

Examples

アンドロイドのZipファイル

```
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();
        }

      }
```

```
    }
```

オンラインでアンドロイドのZipファイルをむ https://riptutorial.com/ja/android/topic/8137/アンドロイドのzipファイル

139: アンドロイドのためのGenymotion

き

Genymotionは、デフォルトのAndroidエミュレータのわりにできるのサードパーティエミュレータです。によっては、のデバイスですることかそれのがあります。

Examples

Genymotionのインストール、

ステップ1 - VirtualBox インストール

オペレーティングシステムにじてVirtualBoxをダウンロードしてインストールします。 Genymotion をするがあります。

ステップ2 - Genymotion ダウンロードGenymotion

Genymotionのダウンロードページに Genymotionし、オペレーティングシステムにじてGenymotion をダウンロードしてください 。

しいアカウントをするか、アカウントでログインするがあります。

ステップ3 - Genymotion インストール

Linuxは、このえをして.binファイルをインストールしてしてください。

ステップ4 - Genymotionのエミュレータをインストールする

- Genymotionする
- ボタンバーのをします。
- あなたのアカウントでログインすれば、なエミュレータをすることができます。
- なものをしてインストールします。

ステップ5 -genymotion とAndroid Studio

Genymotionは、プラグインでAndroid Studioとすることができます。 ここではAndroid Studioにイン

ストールするをします

- ファイル/WindowsおよびLinuxのまたはAndroid Studio /Mac OS Xのにします。
- プラグインをし、リポジトリのをクリックします。
- Genymotionをクリックし、Download and installをクリックします。

これで、プラグインアイコンをることができます。このをてください。

>ツールバーをクリックしてツールバーをしたいがあります。

ステップ6 - Android Studioから Genymotionをする

- ファイル/WindowsおよびLinuxのまたはAndroid Studio /Mac OS Xのにします。
- そのの/ Genymotionにし、 `Genymotion's`フォルダのパスをしてをします。

これで、プラグインアイコンをして、インストールされているエミュレータをし、スタートボタンをすことで、 `Genymotion's`エミュレータをできるはずです。

GenymotionにするGoogleのフレームワーク

がGoogle MapsやGmail、Youtube、Googleドライブなどののサービスをテストしたいは、まずGoogleフレームワークをGenymotionにインストールするがあります。ここにはステップがあります

4.4 Kitkat
5.0 Lollipop
5.1ロリポップ
6.0マシュマロウ
7.0ヌーガット
7.1ヌーガットwebviewパッチ

1. のリンクからダウンロードする
2. ダウンロードしたzipファイルをgenymotionにドラッグドロップするだけでできます
3. Googleアカウントをして「Google Playミュージック」をダウンロードしてします。

-
このトピックにするスタックオーバーフローにする

オンラインでアンドロイドのためのGenymotionをむ https://riptutorial.com/ja/android/topic/9245/アンドロイドのためのgenymotion

140: アンドロイドのAPI

してください

Githubサンプルプロジェクト

Androidのブログスポット

Androidデベロッパーサイト

Examples

Androidアプリケーションでのスキャナの

AndroidはAndroid 6.0MarshmallowSDK 23のAPIをサポートしています

アプリでこのをするには、まずマニフェストにUSE_FINGERPRINTをします。

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

ここにう

まず、ユーザーがでしてKeyGenParameterSpecをしたにのみできるKeyGeneratorをして、Android Key Storeにキーをするがあります。

```
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();
```

KeyGenParameterSpec.Builder.setUserAuthenticationRequiredをtrueにすると、ユーザーのでされたをめ、ユーザーがしたでのみキーのをできます。

```
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);
```

その、FingerprintManager.authenticateをびし、センサーのをきめると、されたでされたがされます。あるいは、としてサーバーでされたパスワードにすこともできます。

`fingerprintManger.class`から`FingerprintManger`してする

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

するには、 `FingerprintManger` APIをし、

`FingerprintManager.AuthenticationCallback`メソッドをオーバーライドします。

```
onAuthenticationError
onAuthenticationHelp
onAuthenticationSucceeded
onAuthenticationFailed
```

めること

fingerPrintイベント・コールをするには、メソッドをします。

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

キャンセル

スキャナコールのリスンをする

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

またはパスワードがされると、FingerprintManager.AuthenticationCallback onAuthenticationSucceededコールバックがびされます。

```
@Override

public void onAuthenticationSucceeded(AuthenticationResult result) {

            }
```

Android Fingerprint APIをしてユーザーのパスワードをする

このヘルパークラスのでは、プリントマネージャとし、パスワードのとをします。こののにされるはAESです。これはのではなく、 のもします。このでは、データはのようにされ、されます。

1. ユーザーはヘルパーにされていないパスワードをえます。
2. ユーザーはをするがあります。
3. されると、ヘルパーは`KeyStore`からキーをし、 `Cipher`をしてパスワードをします。
4. パスワードとIVIVはすべてのでされ、されませんはプリファレンスにされ、プロセスのでされます。

1. ユーザーのパスワード。
2. ユーザーはをするがあります。
3. ヘルパーはIVをして`Cipher`をし、ユーザーがされると、KeyStoreはキー`KeyStore`からキーをし、パスワードをします。

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

のこのアクティビティは、ユーザーがパスワードをしてヘルパーとするのになです。

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

オンラインでアンドロイドのAPIをむ https://riptutorial.com/ja/android/topic/7523/アンドロイドのapi

141: アンドロイドプロジェクトでRetrolambdaをするためのい。

き

Retrolambdaは、Java 7、6または5でJava 8ラムダ、メソッド、およびtry-with-resourcesステートメントをできるライブラリです。

Gradle Retrolambdaプラグインをすると、RetrolambdaをGradleベースのビルドにできます。これにより、なAndroidのはJava 8をサポートしていないため、たとえば、Androidアプリケーションでこれらのをすることができます。

Examples

セットアップと

セットアップ

1. jdk8をダウンロードしてインストールします。
2. プロジェクトのメインbuild.gradleにをします

```
buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
        classpath 'me.tatarka:gradle-retrolambda:3.2.3'
    }
}
```

3. これをアプリケーションモジュールのbuild.gradleにします

```
apply plugin: 'com.android.application' // or apply plugin: 'java'
apply plugin: 'me.tatarka.retrolambda'
```

4. これらのをアプリケーションモジュールのbuild.gradleにして、IDEにレベルをします。

```
android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

だからこのようなもの

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        log("Clicked");
    }
});
```

これになる

```
button.setOnClickListener(v -> log("Clicked"));
```

オンラインでアンドロイドプロジェクトでRetrolambdaをするためのい。をむ
https://riptutorial.com/ja/android/topic/8822/アンドロイドプロジェクトでretrolambdaをするためのい-

142: インターネットをする

き

このメソッドは、WI-Fiがされているかどうかをするためにされます。

- isNetworkAvailableデバイスでインターネットがかどうかをする

パラメーター

パラメータ	
コンテキスト	アクティビティコンテキストのリファレンス

インターネットにされている、メソッドはtrueまたはfalseをします。

Examples

デバイスにインターネットがあるかどうかをする

アプリケーションマニフェストファイルになネットワークアクセスをします。

```
<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;
}
```

どのようにアンドロイドでネットワークのさをするには

```
        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
            }
        }
```

ネットワークのさをチェックする

なをデシベルでするには、

```
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
        }
    }
```

ネットワークタイプをチェックするには、この**Class-**

```
 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 "";
        }
    }
```

```
}
```

オンラインでインターネットをするをむ https://riptutorial.com/ja/android/topic/3918/インターネットをする

143: インターフェイス

Examples

カスタムリスナー

インタフェースをする

```
//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);
}
```

リスナーをする

のステップでは、 `MyCustomListener`でコールバックをするオブジェクトにインスタンスをするがあります。リスナーにセッターをします。

```
public class SampleClassB {
    private MyCustomListener listener;

    public void setMyCustomListener(MyCustomListener listener) {
        this.listener = listener;
    }
}
```

リスナーをする

さて、のクラスでは、 `SampleClassB`インスタンスをできます。

```
public class SomeActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
    }
}
```

の2つのでリスナーを`sampleClass`できます。

クラスの`MyCustomListener`をしています

```
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) {

    }
}
```

またはのクラスをインスタンスするだけです

```
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) {

            }
        });
    }
}
```

トリガーリスナー

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

なリスナー

「リスナー」パターンまたは「オブザーバー」パターンは、Androidでコールバックをするためのもなです。

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

あなたのについて

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

オンラインでインターフェイスをむ https://riptutorial.com/ja/android/topic/1785/インターフェイス

144: インテント

き

インテントは、Androidシステムのりをったさなメッセ―ジです。このメッセ―ジには、たちののにするがっているかもしれません。

には、されるアクションのなをするパッシブなデ―タです。

- インテント
- インテントインテントインテント
- インテントアクション
- インテントインテントストリングアクション、Uri uri
- インテントインテントContext packageContext、Class <> cls
- インテントインテントStringアクション、Uri uri、コンテキストpackageContext、クラス<> cls
- void startActivityインテントの
- void startActivityインテント、バンドルオプション
- void startActivityForResultインテントインテント、int requestCode
- void startActivityForResultintent intent、int requestCode、バンドルオプション
- Intent putExtraString name、double [] value
- Intent putExtraString name、int value
- Intent putExtraString name、CharSequence value
- Intent putExtra、char
- Intent putExtra、バンドル
- Intent putExtraString name、Parcelable [] value
- Intent putExtraString name、Serializable value
- Intent putExtraString name、int [] value
- Intent putExtraString name、float value
- Intent putExtraString name、byte [] value
- Intent putExtraString name、long [] value
- Intent putExtraString name、Parcelable value
- Intent putExtraString name、float [] value
- Intent putExtra、long
- Intent putExtraString name、String [] value
- Intent putExtraString name、boolean value
- Intent putExtraString name、boolean [] value
- Intent putExtra、short
- Intent putExtraString name、double value
- Intent putExtra、short []
- Intent putExtra、

- Intent putExtra、バイト
- Intent putExtraString name、char [] value
- Intent putExtraString name、CharSequence [] value

パラ メーター

パラ メータ	
	する
リクエストコード	をするの
オプション	アクティビティのにするオプション
	なデータの
	なデータの
CHOOSE_CONTACT_REQUEST_CODE	`onActivityResult`メソッドでそれをするリクエストのコード
アクション	このインテントでするアクション、Intent.ACTION_VIEW
ウリ	されたアクションをするでされるデータuri
packageContext	インテントをするためにするコンテキスト
CLS	このインテントでされるクラス

のインテントをするの

のインテントをびすときは、システムがそれをすることがかどうかをすることがにちます。

これは`PackageManager.queryIntentActivities(Intent intent, int flags)`をしてチェックすることによってできます。

```
PackageManager pm = getActivity().getPackageManager();
if (intent.resolveActivity(pm) != null) {
    //intent can be handled
    startActivity(intent);
} else {
     //intent can not be handled
}
```

singleTask

または`singleTop`あるアクティビティ

アクティビティのモードが`singleTask`または`singleTop`、アクティビティがデータnullでされるとすぐに`onActivityResult`がびされます。これをぐには、`Intent.setFlags(0)`をしてデフォルトのフラグをリセットします。

Examples

アクティビティをする

このでは、`OriginActivity`から`DestinationActivity`をし`OriginActivity`。

ここで、`Intent`コンストラクタは2つのパラメータをとります

1. のパラメータとしてのContextこれは、Activityクラスが`Context`のサブクラスであるためにされます
2. インテントをするのあるアプリケーションコンポーネントのクラスこのは、するがあるアクティビティ

```
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.
    }
}
```

するためののとして`Intent`くため`DestinationActivity`のデフォルトのコンストラクタをすることで`Intent`、そして`setClass()`オープンしているにそれをえるためのを

```
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
```

アクティビティでデータをす

このでは、`"Some data!"`の`String`をしてい`"Some data!"` `OriginActivity`から`DestinationActivity`へ

これは、2つのアクティビティでデータをするもなです。よりなには、スターターパターンのをしてください。

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);
    }
}
```

のprimitiveデータ、 arrays 、 BundleとParcelableデータもすことができます。 Serializableすこともですが、 Parcelableより3れているのでけてParcelable。

シリアライズなのは、のJava interfaceです。 Serializable interfaceをすることで、クラスをSerializableとマークするだけで、Javaはなでに Serializable interface。

Parcelableは、のデータつまりのオブジェクト/ POJOオブジェクトでできるAndroidのinterfaceです。オブジェクトをフラットにして、もするのないデスティネーションなしでできます。 オブジェクトをにするためのドキュメンテーションがあります 。

parcelableなオブジェクトをparcelableたら、それをインテントオブジェクトとともにプリミティ

ブのようにることができます

```
intent.putExtra(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

または、バンドル/フラグメントのとして

```
bundle.putParcelable(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

getParcelableExtraをしてのインテントからもみることができます。

```
final MyParcelableType data = intent.getParcelableExtra(EXTRA_DATA);
```

または、バンドルからフラグメントをみむとき

```
final MyParcelableType data = bundle.getParcelable(EXTRA_DATA);
```

`Serializable`オブジェクトを`Serializable`たら、それをインテントオブジェクトにれることができます

```
bundle.putSerializable(DestinationActivity.EXTRA_DATA, mySerializableObject);
```

のようにのインテントオブジェクトからそれをみみます。

```
final SerializableType data = (SerializableType)bundle.getSerializable(EXTRA_DATA);
```

メールをする

```
// 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
}
```

これは、ユーザーがしたメールアプリケーションにメールをあらかじめします。

ファイルをするがあるは、 `Intent.ACTION_SEND`わりに`Intent.ACTION_SENDTO`できます。のファイルの、 `ACTION_SEND_MULTIPLE`をできます

のはいないすべてのデバイスは、プロバイダのためのっているACTION_SENDTOびすと、startActivity()でチェックせずにresolveActivity()ActivityNotFoundExceptionをスローすることがあります。

のアクティビティからをする

startActivityForResult(Intent intent, int requestCode)をすると、のActivityをし、そのActivityからonActivityResult(int requestCode, int resultCode, Intent data)メソッドのをけることができます。はIntentとしてされます。インテントには、バンドルをしてデータをめることができます

このでは、 MainActivityはMainActivityをし、 DetailActivityからをします。タイプは、それのつべきint オーバーライドになるように、コードをonActivityResult(int requestCode, int resultCode, Intent data)におけるMainActivity 、それのとすることによってするためにどのをすることができるrequestCodeとREQUEST_CODE_EXAMPLEこのでもたとえば、1つしかありません。

な

```
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();
    }
}
```

あなたがづくがあるいくつかの

- データは、 `finish()` をびしたにのみされます。 `finish()` びすに `setResult()` をびすがあります。そうしないと、はされません。

- あなたのことをしてください`Activity`されていない`android:launchMode="singleTask"`、またはそれがとなり`Activity`のタスクでするように、したがって、あなたはそれからをしません。あなたの`Activity`がモードとして`singleTask`をするは、コード`Activity.RESULT_CANCELED`すぐに`onActivityResult()`をびします。

- `android:launchMode="singleInstance"`をするときはしてください。 LollipopAndroid 5.0、API レベル21のでは、アクティビティからがされません。

- `startActivityForResult()`をびすときは、またはなインテントをできます。をけるためにのの1つをするときは、なをして、されるをにけるべきです。な`intent`は、それにがまれていてもにターゲットにされます。 `filter`はされません。しかし、のインテントは、コンポーネントのフィルタの1つをできるにのみ、コンポーネントにされます。

ブラウザでURLをく

デフォルトのブラウザでく

このでは、アプリケーションではなくみみのWebブラウザでプログラムでURLをくをします。これにより、マニフェストファイルに`INTERNET`アクセスをめるなく、アプリがウェブページをくことができます。

```
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);
    }
}
```

ユーザーにブラウザのをす

このでは、 `Intent.createChooser()`メソッドをしている`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"));

}
```

によっては、URLは「www」でまるがあります。その、このがします

`android.content.ActivityNotFoundException` Intentをするアクティビティがつかりません

URLはに「http*//*」または「https*//*」でまるがあります。したがって、のコーードスニペットにすように、コーードでチェックするがあります。

```
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);
}
```

ベストプラクティス

のインテントをけることができるアプリがデバイスにないかどうかをします。それのは、`startActivity()`びすとアプリケーションがクラッシュします。インテントをけるアプリケーションがすることをにするには、Intentオブジェクトにして`resolveActivity()`をびします。がnullでないは、インテントをできるアプリがなくとも1つあり、 `startActivity()`をびしてもです。がnullのは、インテントをしないでください。であればインテントをびすをにするがあります。

アクティビティスタックをクリアする

バック･スタックからのアクティビティをしているに、しいアクティビティをしたいがあります。そのとして、ログインアクティビティでアプリケーションをし、アプリケーションのメインアクティビティをすることがありますが、ログアウトしても、ログインすることなくってくることができます。そのような、インテントの`FLAG_ACTIVITY_CLEAR_TOP`フラグをできます。つまり、のアクティビティがすでにのタスクLoginActivityでされているは、そのアクティビティのしいインスタンスをするわりに、それがされ、このインテントはしいインテントとしてのいアクティビティにされます。

```
Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
```

これは、フラグをすることもです`FLAG_ACTIVITY_NEW_TASK`とに`FLAG_ACTIVITY_CLEAR_TASK`あなたがって、スタックのすべてのをクリアしたいは

```
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

このでは、ブラウザからインテントをするをします。

```
<a href="intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end">Start
intent</a>
```

このインテントは、 `com.sample.test`パッケージでアプリをするか、このパッケージでgoogle playをきます。

また、このはjavascriptでめることができます

```
var intent = "intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end";
window.location.replace(intent)
```

アクティビティでは、このホストとパスはインテントデータからできます。

```
@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    Uri data = getIntent().getData(); // returns host.com/path
}
```

インテントURI

```
HOST/URI-path // Optional host
#Intent;
    package=[string];
    action=[string];
    category=[string];
    component=[string];
    scheme=[string];
end;
```

のコンポーネントへのメッセージのブロードキャスト

インテントは、アプリケーションののコンポーネントのバックグラウンドサービスなどまたはAndroidシステムにメッセージをブロードキャストするためにできます。

アプリケーションでブロードキャストをするには、 `LocalBroadcastManager`クラスをします。

```
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);
```

アプリケーションののコンポーネントにブロードキャストをするには、 `Context`オブジェクトで`sendBroadcast()`メソッドをします。

```
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);
```

ブロードキャストのにするは、 ブロードキャストレシーバー

CustomTabsIntent for Chromeのカスタムタブ

4.0.3

`CustomTabsIntent`をすると、アプリからいたブラウザでなUIコンポーネントをカスタマイズするためにChromeのカスタムタブをできるようになりました。

これは、によってはWebViewをするいです。インテントをつWebページをみむことができ、ブラウザにアプリのルック･アンド･フィールをあるできるようになりました。

`CustomTabsIntent`をしてURLをくのをにします

```
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));
```

カスタムタブをするには、このをbuild.gradleにするがあります

```
compile 'com.android.support:customtabs:24.1.1'
```

インテントによるのファイルの

`share()`メソッドへのパラメータとしてされるString Listは、 `share()`たいすべてのファイルのパスをんでいます。

にはパスをループしてUriにし、このタイプのファイルをけれることができるアクティビティをします。

```
  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);
    }
```

スター―ター―パター―ン

このパター―ンは、 `Activity`をするためのよりなアプロ―チです。そのは、コ―ドのをさせるとに、コ―ドのさ、コスト、およびコンポ―ネントのをらすことです。

のでは、、 `Activity`のメソッドとしてされるスター―ター―パター―ンをしています。このメソッドは、なすべてのパラメ―タをけり、そのデ―タからな`Intent`をし、 `Activity`します。

`Intent`は、2つのアクティビティなど、々のコンポ―ネントでバインディングをするオブジェクトです。インテントは、アプリの「かをする」をします。インテントはさまざまなタスクにできますが、ここではのアクティビティがされます。

```
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);
    }
}
```

このパター―ンでは、のデ―タをインテントとともにすこともできます。

に、 `ExampleActivity`をこのようにすることができます。ここで、 `context`はアクティビティコンテキストです。

```
ExampleActivity.start(context, "Some data!");
```

インテントをしてアンバウンドサ―ビスをする

サ―ビスは、ユ―ザ―とすることなく、UIスレッドでバックグラウンドでされるコンポ―ネントです。バインドされていないサ―ビスはされたばかりであり、アクティビティのライフサイクルにされていません。

サービスをするには、ののようにします。

```
// 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);
```

`onStartCommand()`オーバーライドをすると、インテントからのエキストラをできます。

```
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;
    }

}
```

の

なるアプリとなをする。

```
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)));
```

なるアプリとをする。

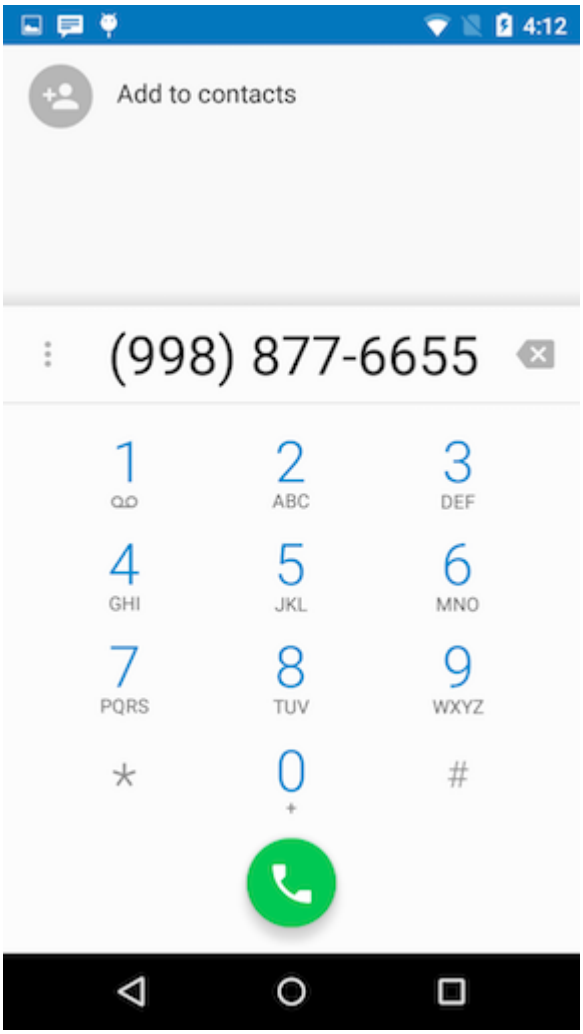
```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uriToImage);
shareIntent.setType("image/jpeg");
startActivity(Intent.createChooser(shareIntent, getResources().getText(R.string.send_to)));
```

ダイヤラをする

のは、されたをして、デフォルトのダイヤラのびしをうアプリケーションをくをしています。

```
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);
```

のコードをした



された、でGoogleマップをきます

Intentをしてアプリから Googleマップに、をすことができます

```
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);
```

アクティビティのインテントをじてなるデータをす

1.データをす

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.ダブルデ―タをす

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.デ―タのけし

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");
```

または

```
Intent mIntent = getIntent();
String stringValue = mIntent.getStringExtra("stringVariableName");
```

4. ArrayListデ―タをす

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putStringArrayListExtra("arrayListVariableName", arrayList);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
arrayList = mIntent.getStringArrayListExtra("arrayListVariableName");
```

5.オブジェクトデ―タのけし

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("ObjectVariableName", yourObject);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
yourObj = mIntent.getSerializableExtra("ObjectVariableName");
```

カスタムクラスは`Serializable`インター―フェイスをするがあることにしてください。

6. HashMap <String、String>デ―タをす

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.ビットマップデ―タをす

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("image",bitmap);
startActivity(mIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
Bitmap bitmap = mIntent.getParcelableExtra("image");
```

ファイルセレクタアクティビティの

```
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);
}
```

をむ

```
@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;
}
```

アクティビティでカスタムオブジェクトをす

`Bundle`クラスをしてカスタムオブジェクトをのアクティビティにすこともできます。

2つのがあります。

- JavaおよびAndroidの`Serializable`インターフェイス
- `Parcelable`インターフェース - メモリはで、Androidのみ

パーセルブル

なはシリアライズなよりもはるかにです。この1つは、リフレクションをしてするわりに、シリアライゼーションプロセスについてにしていることです。また、こののためにコードがにされたもあります。

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

アクティビティコードの

```
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);
```

アクティビティでオブジェクトをします。

```
//Getting MyObjects
Intent mIntent = getIntent();
MyObjects workorder = (MyObjects) mIntent.getParcelable("UniqueKey");
```

あなたはのように**Paraceble**オブジェクトの**Arraylist**をすことができます

```
//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");
```

このようなAndroid Studioプラグインは、パーセルなコードをするためにです

シリアライザブル

アクティビティコードの

```
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);
```

アクティビティでオブジェクトをします。

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Bundle bundle = this.getIntent().getExtras();
    Product product = null;
    if (bundle != null) {
        product = (Product) bundle.getSerializable("product");
    }
```

オブジェクトの**`Arraylist`** オブジェクトしとじ

カスタムオブジェクトは、 `Serializable`インターフェイスをするがあります。

のをする

のアクティビティからをするは 、 `Fragment`のメソッド`startActivityForResult(Intent intent, int requestCode)`をびすがあります。 `getActivity().startActivityForResult()`をびさないでください `getActivity().startActivityForResult()`これは、を`Fragment`の`Activity`すためです。

をけるには、 `Fragment`のメソッド`onActivityResult()`ます。フラグメントのアクティビティが `onActivityResult()`をオーバーライド`onActivityResult()` 、それが`super`インプリメンテーションであることをするがあります。

のでは、 `ActivityOne`に`FragmentOne`がまれています`FragmentOne`は`ActivityTwo`をし、そのをします。

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
        }
    }
}
```

アクティビティ2

```
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();
    }
}
```

オンラインでインテントをむ https://riptutorial.com/ja/android/topic/103/インテント

145: ウィジェット

SDv

Examples

マニフェスト -

アプリケーションの`AndroidManifest.xml`ファイルで`AppWidgetProvider`クラスをします。えば

```
<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>
```

メタデータ

`res/xml` AppWidgetProviderInfoメタデータをします。

```
<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>
```

AppWidgetProviderクラス

もな`AppWidgetProvider`コールバックは、 **`onUpdate()`**です。これは、appwidgetがされるたびにびされます。

```
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()`は、ウィジェットのまたはサイズにびされます。

ウィジェットがされると`onDeleted(Context, int[])`がびされます。

なるレ イアウトをつ2つのウィジェット

1. マニフェストファイルに2つのをする

```
<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. 2つのレイアウトをする
    - `@xml/widget_1x1`
    - `@xml/widget_2x2`
3. サブクラス`UVMateWidget2x2`をの`UVMateWidget`クラスからします。

```
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) {

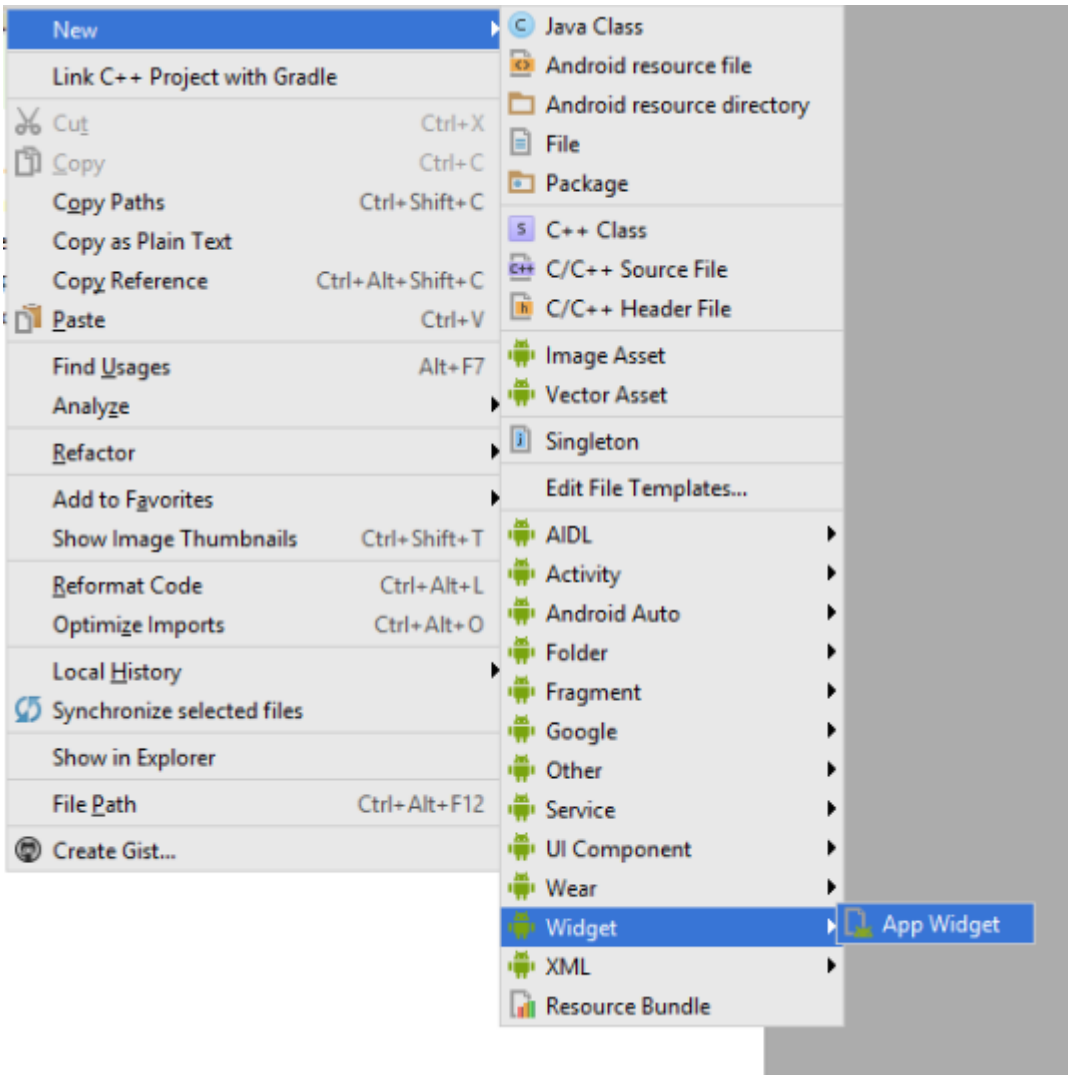
        mUVMateHelper.saveWidgetSize(mContext.getString(R.string.app_ws_2x2));
        return new RemoteViews(context.getPackageName(), R.layout.widget_2x2);
    }
}
```

Android Studioをしたウィジェットの/

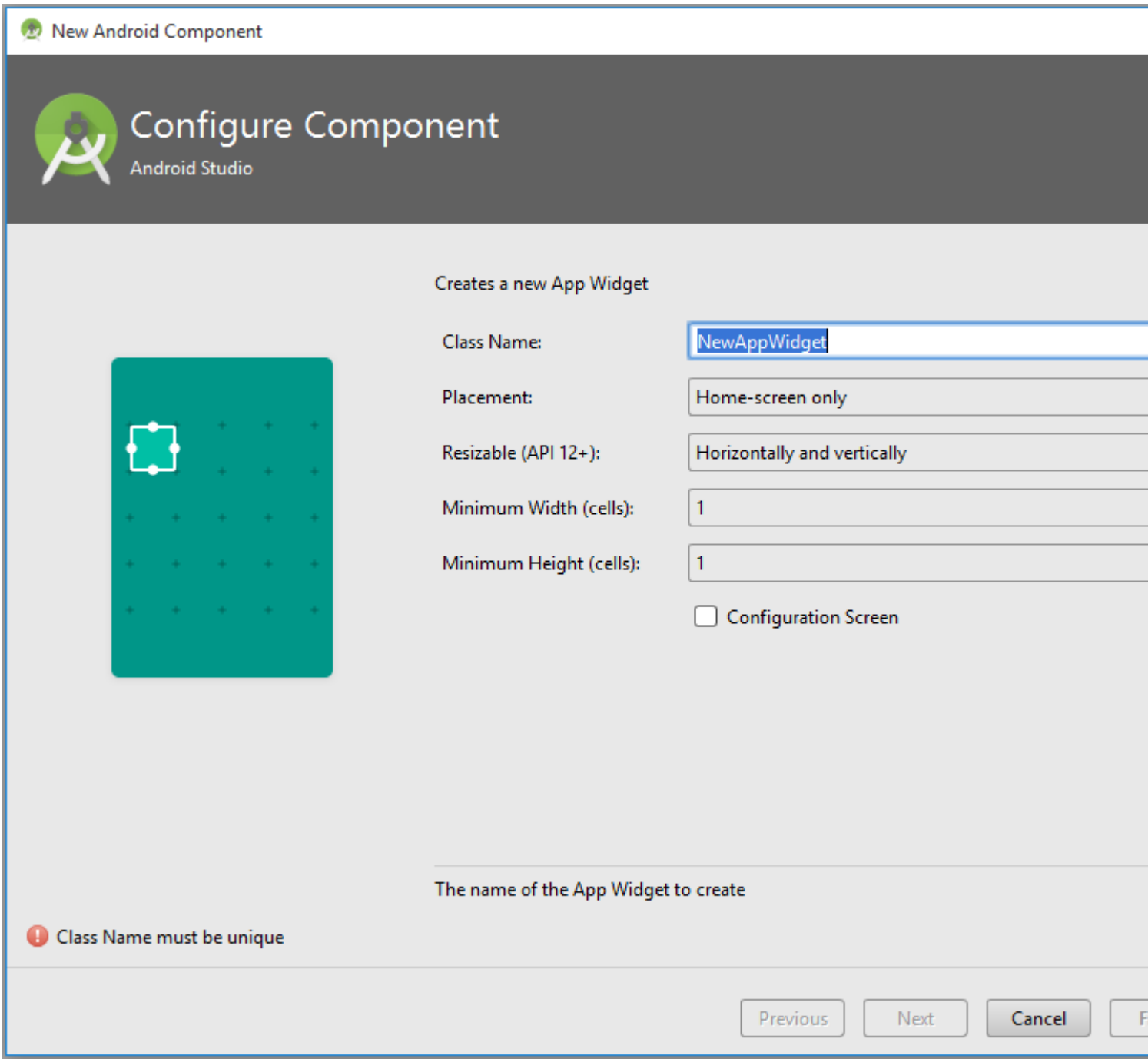
のAndroid Studioは、2でウィジェットをし、アプリケーションにします。

あなたのアプリケーションでに==> New ==>ウィジェット==> Appウィジェット

。



のようながされ、フィールドをします



しました。

あなたのアプリケーションにな**HelloWorld**ウィジェット レイアウトファイル、メタデータファイル、マニフェストファイルのなどをむをしてします。

オンラインでウィジェットをむ https://riptutorial.com/ja/android/topic/2812/ウィジェット

146: エスプレッソによるUIのテスト

エスプレッソ

エスプレッソのチートシートは、あなたがあなたのテストをくのにつでしょう、あなたがテストしたいもの

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

また、にになるのはです。

https://google.github.io/android-testing-support-library/docs/espresso/index.html

Googleによるエスプレッソビデオのなアドバイス https : //www.youtube.com/watch?v=isihPOY2vS4

トラブルシューティング

- スクロールしようとするときは、まずキーボードをじてください。

ウォッチアウト「エスプレッソ」バージョンをしないと、ViewActionのでされてももされません。 ViewActionバージョンにインポートがあるは、メソッドがまったくじであるため、これはではありません。

```
ViewActions.closeSoftKeyboard;
Espresso.closeSoftKeyboard();
```

- 々のテストではなくスイートでテストをにする、のテストのアクティビティがであるがあることにしてください。のテストのonDestroyがのテストonResumeのにびされていることにらないでください。 これはにはバグです。 http : //b.android.com/201513

Examples

エスプレッソをセットアップする

あなたのAndroidアプリケーションモジュールの`build.gradle`ファイルにのをします

```
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'
}
```

`build.gradle`ファイルの`testInstrumentationRunner`パラメータに`AndroidJUnitRunner`をします。

```
android {

  defaultConfig {
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }

}
```

さらに、インテントモックのサポートをするためにこのをします

```
androidTestCompile 'com.android.support.test.espresso:espresso-intents:2.2.2'
```

そしてこれをWebviewテストのサポートにします

```
// Espresso-web for WebView support
androidTestCompile 'com.android.support.test.espresso:espresso-web:2.2.2'
```

エスプレッソテストクラスをする

のjavaクラスを**src / androidTest / java**にき、します。

```
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
  }

}
```

くじる**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);
      }
    };
  }

}
```

エスプレッソシンプルなUIテスト

UIテストツール

UIテストににされている2つのなツールはAppiumとEspressoです。

アピウム	エスプレッソ
ブラックボックステスト	ホワイト/グレーボックステスト
あなたがるものはあなたがテストできるものです	アプリのをしてテストにすることができます。たとえば、テストをするにデータベースやプレビューにデータをする
にエンドツーエンドテストとユーザーフローにされます	やフローのをテストする
されたテストはiOSとAndroidでできるようにすることができます	Androidのみ
よくサポートされている	よくサポートされている
セレングリッドをしてのデバイス	スプーンのようなプラグインは、Googleののサポートが

アピウム	エスプレッソ
でテストをサポート	てくるまでします

エスプレッソをプロジェクトにする

```
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'
}
```

のサポートライブラリ、などをしている、をけるためにエスプレッソからバージョンをするがあります。

```
    // 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'
    }
```

これらのインポートに、アンドロイドテストランナーをbuild.gradle android.defaultConfigにするがあります。

```
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
```

デバイスの

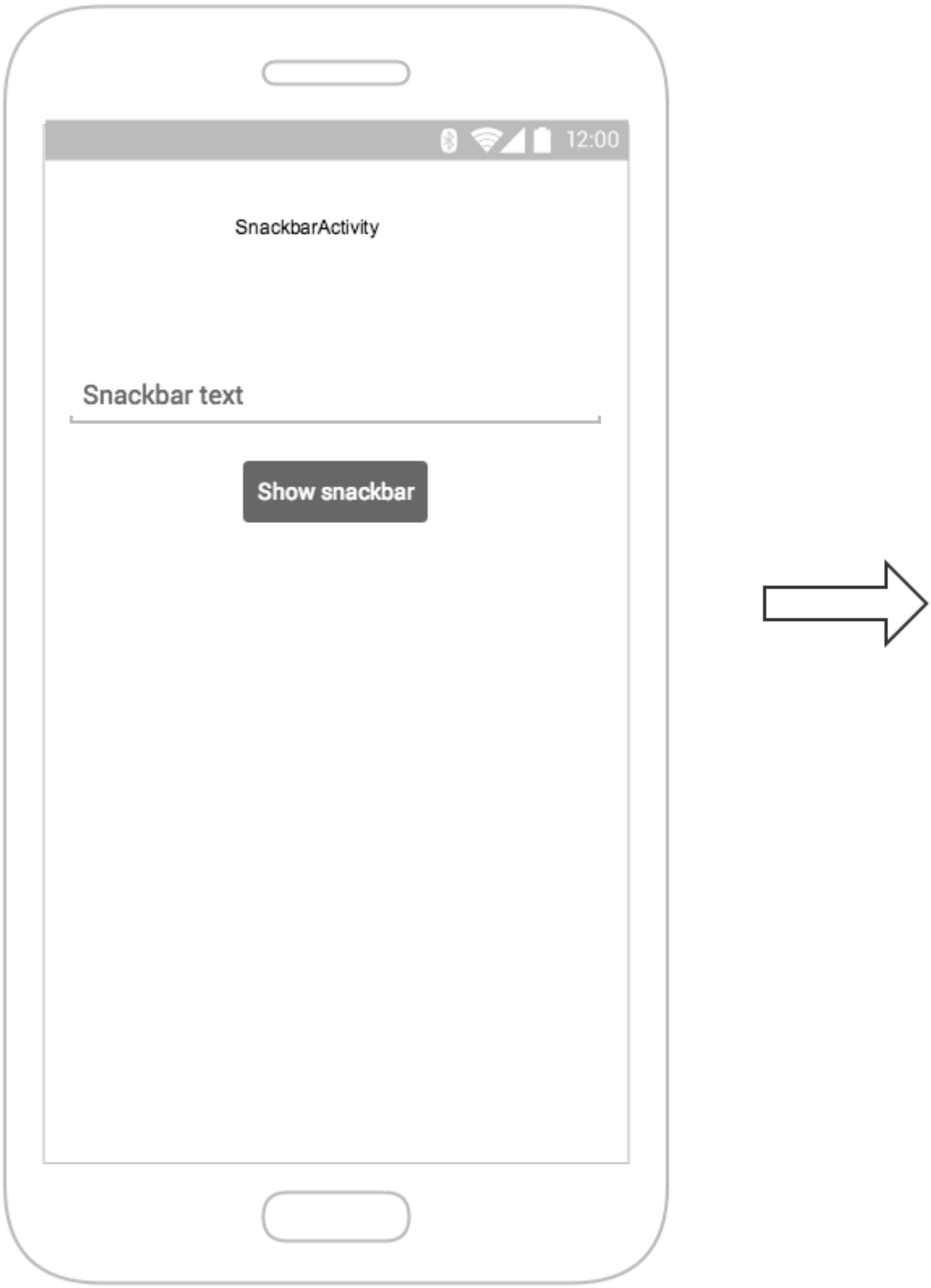
フレークテストのは、デバイスでのをうことをおめします。

- オプション/アニメーションをにする - テストのさを
- のオプション/をます - テストのデバイスがある、これはです
- デベロッパーオプション/ロガーバッファーサイズ - でにきなテストスイートをするにいに
- アクセシビリティ/タッチホールドの - エスプレッソでのタップのをけるためにい

からのかなりのセットアップはさてさて、それでは、さなテストをセットアップするをてみましょう

テストの

のがあるとします。



にはがまれます

- テキストフィ―ルド - **R.id.textEntry**
- クリックされるとされたテキストをむスナックバ―をするボタン - **R.id.shownSnackbarBtn**
- ユ―ザがしたテキストをむべきsnackbar - **android.support.design.R.id.snackbar_text**

に、フローをテストするクラスをしましょう

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

あなたがづいたように、あなたはにるかもしれない3-4ものがあります

onViewwithXYZ < - viewMatchersをすると、のをつけることができます

< - viewActionsを**click**すると、につかったにしてアクションをできます

checkmatchesisDisplayed < - viewAssertions、につかったでするチェック

これらのやそののは、 https //google.github.io/android-testing-support-library/docs/espresso/cheatsheet/index.htmlでごいただけます。

これで、クラス/テストをクリックしてテストをするか、テストをするかコマンドをすることでテストをできます

```
./gradlew connectedFLAVORNAMEAndroidTest
```

へのナビゲーション

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

これはであり、じコンテンツをつのビューとすることにしてください。

ビューにするアクションの

performメソッドをしてビューにして`ViewActions`をすることはです。
`ViewActions`クラスは、もなアクションのためのヘルパーメソッドをします

```
ViewActions.click()
ViewActions.typeText()
ViewActions.clearText()
```

たとえば、ビューをクリックするには

```
onView(...).perform(click());
onView(withId(R.id.button_simple)).perform(click());
```

1つのコールでのアクションをできます。

```
onView(...).perform(typeText("Hello"), click());
```

しているビューが`ScrollView`またはにあるは、 `scrollTo()`てビューをするがあるのアクション`click()`や`typeText()` をして`click()`。 これにより、のアクションにむにビューがされます。

```
onView(...).perform(scrollTo(), click());
```

onViewでビューをつける

`ViewMatchers`

をすると、のビュ―でビュ―をつけることができます。

ビュ―をつけるには、しいビュ―をするビュ―マッチャ―で`onView()`メソッドをします。 `onView()` メソッドは、 `ViewInteraction`のオブジェクトをします。

たとえば、 `R.id`によるビュ―のは、のようにです。

```
onView(withId(R.id.my_view))
```

テキストをむビュ―をつける

```
onView(withText("Hello World"))
```

エスプレッソカスタムマッチャ―

エスプレッソには、デフォルトでくのマッチャ―があり、チェックやをうのになビュ―をつけるのにちます。

もなものはのチ―トシ―トにあります

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

マッチャ―のをいくつかげます

- withIdR.id.of_object_you_are_looking_for;
- withText "オブジェクトがするテキストの";
- isDisplayed< - checkはのビュ―です
- doesNotExist< - ビュ―がしないことをする

これらはすべてのにはにですが、よりなビュ―をしているは、カスタムマッチャ―をするとテストがみやすくなり、のでテストをできます。

なマッチャ―のもなタイプは2つあります **TypeSafeMatcher BoundedMatcher**

TypeSafeMatcherをすると、アサ―ションしているビュ―のインスタンスをチェックするがあります。

たとえば、イメ―ジビュ―をするタイプセ―フマッチャ―にはしいdrawableがあります。

```
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("]");
        }
    }
}
```

マッチャーのいはのようにラップすることができます

```
  public static Matcher<View> withDrawable(final int resourceId) {
    return new DrawableMatcher(resourceId);
}

 onView(withDrawable(R.drawable.someDrawable)).check(matches(isDisplayed()));
```

バインドされたマッチャーは、タイプチェックをうはありませんが、それはにわれます。

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

マッチャーのはのでむことができます

http://hamcrest.org/

https://developer.android.com/reference/android/support/test/espresso/matcher/ViewMatchers.html

エスプレッソ

セットアップエスプレッソ

```
androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
androidTestCompile 'com.android.support.test:runner:0.5'
```

ViewMatchers - `Matcher<? super View>`をするオブジェクトのコレクション`Matcher<? super View>`インターフェイスをきます。これらのうちの1つを`onView`メソッドに`onView`、のビューのビューをつけることができます。

ViewActions - `ViewInteraction.perform()`メソッドにすことができる`ViewActions`のコレクションですたとえば、`click()`。

ViewAssertions - `ViewInteraction.check()`メソッドを`ViewAssertions`ことができる`ViewAssertions`のコレクション。ほとんどの、マッチアサーションをします。アサーションは、ビューマッチャーをしてされているビューのをアサートします。

エスプレッソのチートシート **by 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

エスプレッソによるuiのテスト

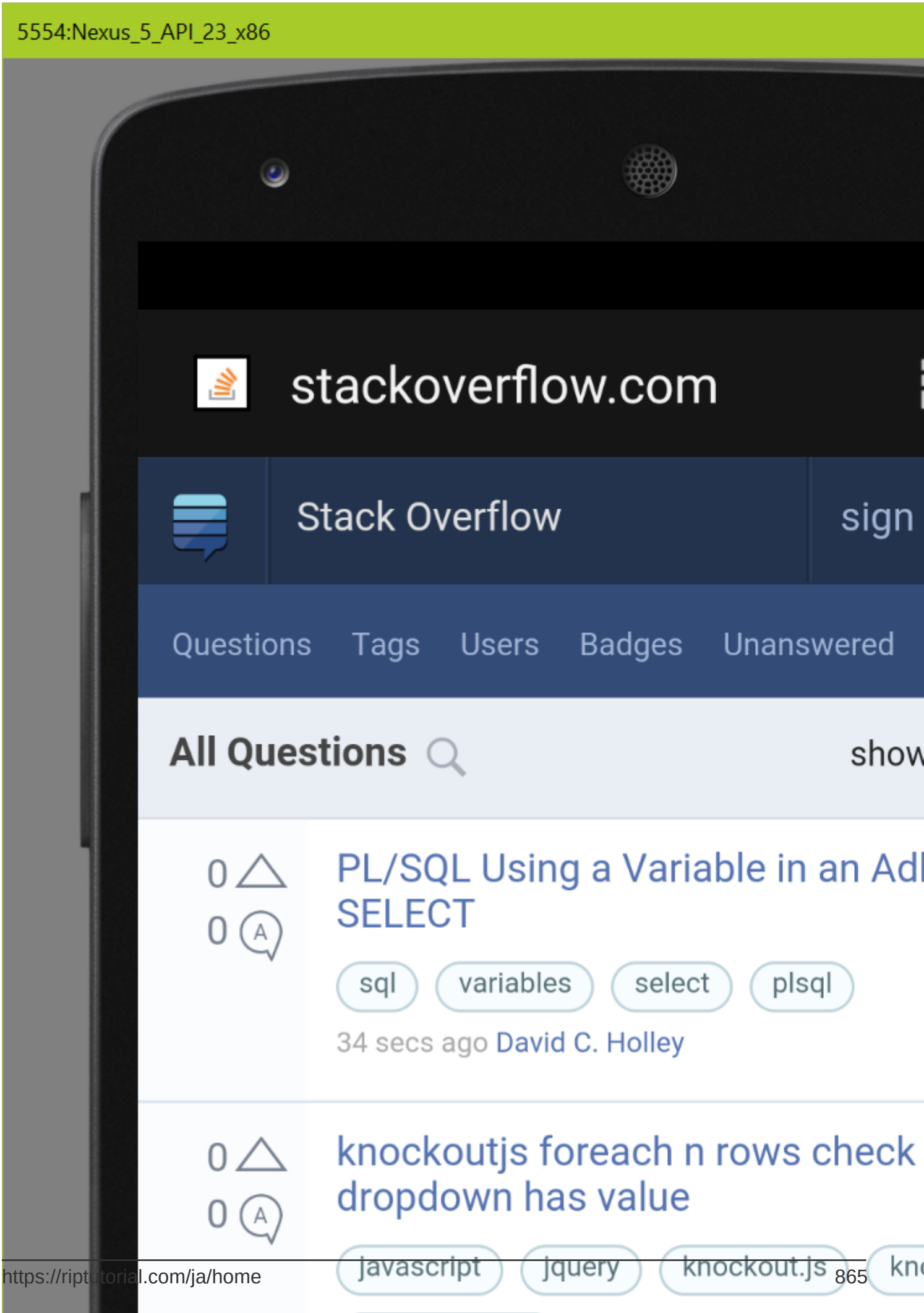
147: エミュレータ

AVDは*Android*デバイス

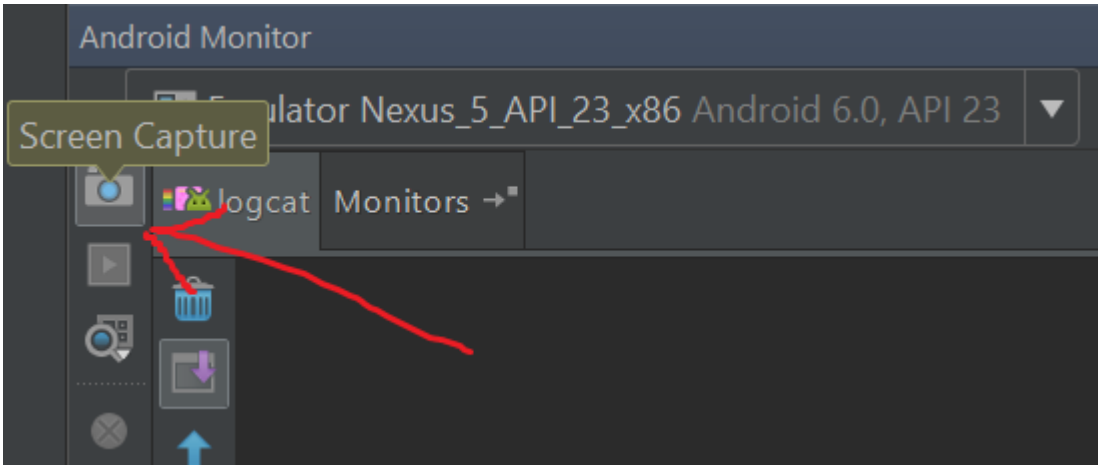
Examples

スクリーンショットをる

Android Emulator2.0からスクリーンショットをりたいは、 `Ctrl` + `S`キーをすか、サイドバーのカメラアイコンをクリックするだけです。


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

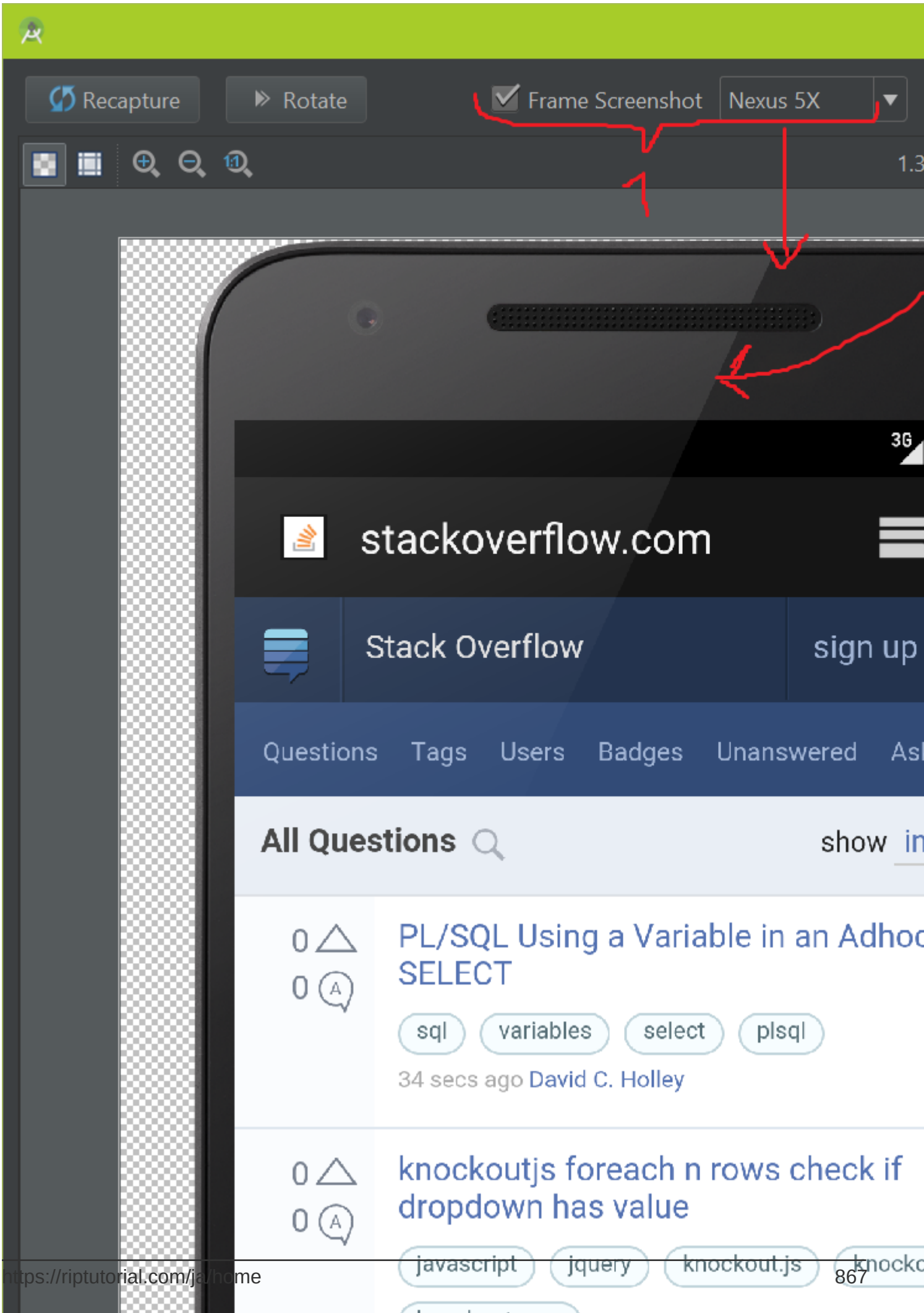
は、Androidモニターのカメラアイコンをクリックするがあります。



これはなとしであるため、しいデバイスをしたことをしてください。

スクリーンショットをったら、のをオプションですることもできますのも。

1. スクリーンショットのりのデバイスフレーム。
2. デバイスフレームののドロップシャドウ。
3. デバイスフレームとスクリーンショットをるスクリーングレア。


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

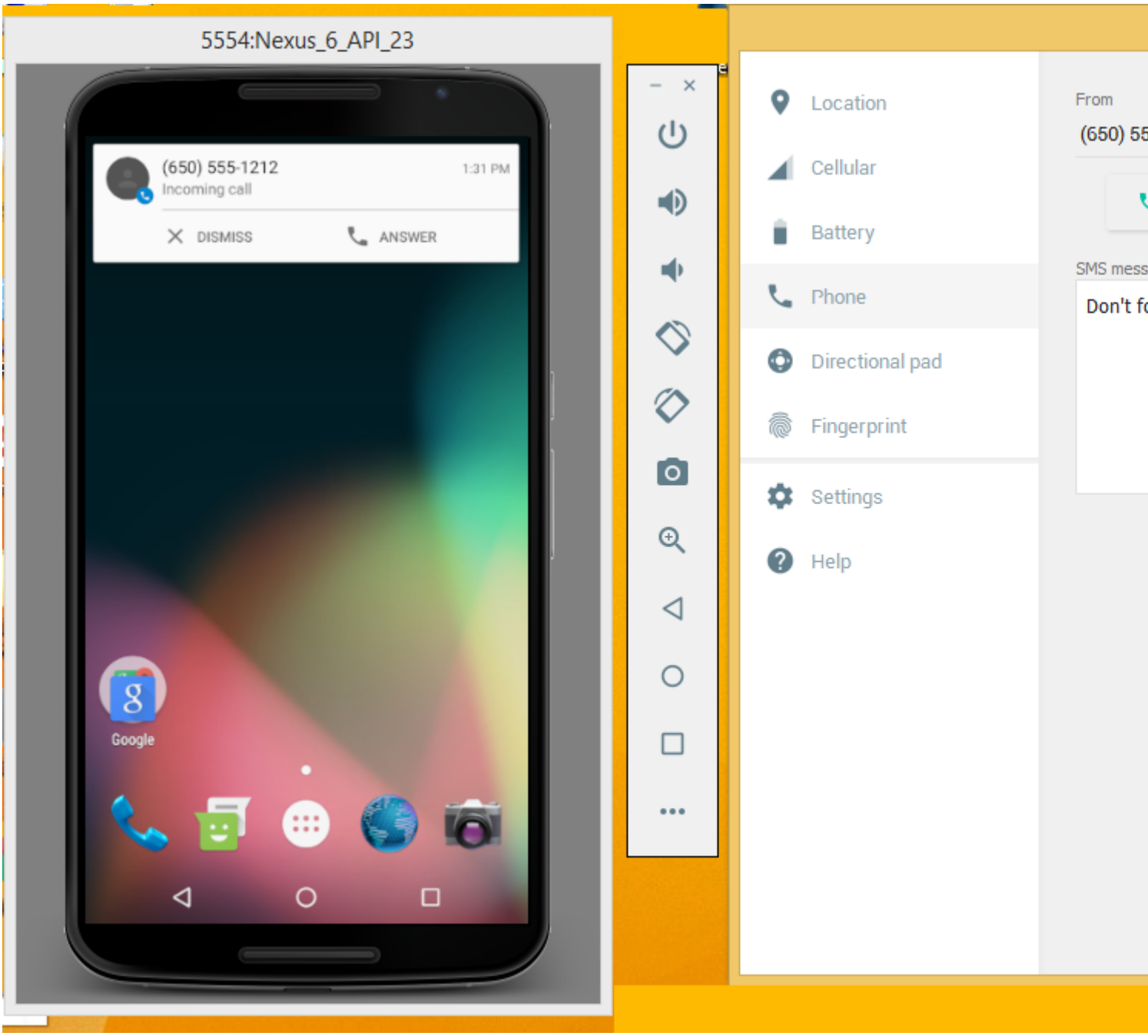
くことができます。

また、 `Tools` > `Android` > `AVD Manager`か、のスクリーンショットの2のツールバーのAVDマネージャーアイコンをクリックして、Androidスタジオ からAVDマネージャにアクセスすることもできます。



をシミュレートする

をシミュレートするには、3つのでされた「」ボタンをし、「」をして「」をします。オプションでをすることもできます。



エミュレータのにエラーをする

まず、BIOSセットアップで「 」をにしたことをしてください。

Android SDKマネージャをし、 **Extras**をし、 **Intel Hardware Accelerated Execution Manager**をし、ダウンロードがするまでちます。それでもしないは、SDKフォルダをき、`/extras/intel/Hardware_Accelerated_Execution_Manager/IntelHAXM.exe`をして`/extras/intel/Hardware_Accelerated_Execution_Manager/IntelHAXM.exe`。

のにってインストールをしてください。

または、OS Xのは、 `/extras/intel/Hardware_Accelerated_Execution_Manager/HAXM\ installation`ようなのプロンプトをせずにでき`/extras/intel/Hardware_Accelerated_Execution_Manager/HAXM\ installation`

> CPUがVT-xまたはSVMをサポートしていない、x86ベースのAndroidイメージはできません。わりにARMベースのをしてください。

インストールがしたら、コマンドプロンプトウィンドウをき、のコマンドをして、ドライバがしくしていることをし`sc query intelhaxm`

VMアクセラレーションでx86ベースのエミュレータをするにはコマンドラインからエミュレータをしているは、x86ベースのAVDをします。 `emulator -avd <avd_name>`

のをすべてしくすると、HAXMがにしていることをしてください。

オンラインでエミュレータをむ https://riptutorial.com/ja/android/topic/122/エミュレータ

148: オーディオトラック

Examples

ののトーンをする

ののをするには、まずのサウンドをするがあります。これはのようにしてわれます。

```
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
}
```

これで、されたバッファにわせてするようにAudioTrackをするがあります。これはのようにしてわれる

```
AudioTrack audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC,
                    sampleRate, AudioFormat.CHANNEL_OUT_MONO,
                    AudioFormat.ENCODING_PCM_16BIT, buffer.length,
                    AudioTrack.MODE_STATIC);
```

されたバッファをきみ、トラックをする

```
audioTrack.write(buffer, 0, buffer.length);
audioTrack.play();
```

おにてれば

オンラインでオーディオトラックをむ https://riptutorial.com/ja/android/topic/9155/オーディオトラック

149: オーバーレイにのWindowsの

Examples

ポップアップオーバーレイ

すべてのアプリケーションのにビューをするには、ビューをするウィンドウマネージャにりてるがあります。そのためには、システムアラートのアクセスがです。これは、マニフェストファイルにのをすることでできます。

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

アプリケーションがされた、ビューはウィンドウマネージャからされます。したがって、ビューをし、それをフォアグラウンドサービスによってウィンドウマネージャにりてるがいです。

WindowManagerへのビューのりて

のようにして、ウィンドウマネージャインスタンスをできます。

```
WindowManager mWindowManager = (WindowManager)
mContext.getSystemService(Context.WINDOW_SERVICE);
```

ビューのをするには、のようにいくつかのレイアウトパラメータをするがあります。

```
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;
```

これで、されたレイアウトパラメータとビューをウィンドウマネージャインスタンスにのようにりてることができます

```
mWindowManager.addView(yourView, mLayoutParams);
```

Voilaあなたのビューはのすべてのアプリケーションのにうまくされています。

ビューはキーガードのにかれません。

SYSTEM_ALERT_WINDOWのアンドロイド6.0の

アンドロイド6.0から、このはにするがあります。

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

アクセスをすると、6.0のエラーがされました。

```
Caused by: android.view.WindowManager$BadTokenException: Unable to add window
android.view.ViewRootImpl$W@86fb55b -- permission denied for this window type
```

-

のようにオーバーレイをし、

```
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);
}
```

をし、

```
@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...
        }
    }
}
```

オンラインでオーバーレイにのWindowsのをむ https://riptutorial.com/ja/android/topic/6214/オーバーレイ-に-のwindowsの

150: オットーイベントバス

オットーは`RxJava`と`RxAndroid`してされ`RxAndroid`。これらのプロジェクトでは、Ottoとじイベントドリブンプログラミングモデルができますが、よりれたをえ、スレッドのがしています。

Examples

イベントにする

このでは、 Otto Event Busをしてイベントをすについてします。

Android StudioでOtto Event Busをするには、モジュールgradleファイルにのをするがあります。

```
dependencies {
    compile 'com.squareup:otto:1.3.8'
}
```

たちがしたいイベントはなJavaオブジェクトです

```
public class DatabaseContentChangedEvent {
    public String message;

    public DatabaseContentChangedEvent(String message) {
        this.message = message;
    }
}
```

イベントをるにはバスがです。これはにシングルトンです

```
import com.squareup.otto.Bus;

public final class BusProvider {
    private static final Bus mBus = new Bus();

    public static Bus getInstance() {
        return mBus;
    }

    private BusProvider() {
    }
}
```

イベントをするには、BusProviderと`post`メソッドがです。ここでは、AsyncTaskのアクションがしたにイベントをします。

```
public abstract class ContentChangingTask extends AsyncTask<Object, Void, Void> {

    ...
```

```
    @Override
    protected void onPostExecute(Void param) {
        BusProvider.getInstance().post(
            new DatabaseContentChangedEvent("Content changed")
        );
    }
}
```

イベントをける

イベントをけるには、イベントタイプをパラメータとしてメソッドをし、 `@Subscribe`をしてをける`@Subscribe`ます。さらに、 `BusProvider`オブジェクトのインスタンスを/するがあります イベントののを。

```
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);
    }
}
```

そのイベントをけるには、そのクラスのインスタンスがするがあります。これは、あるアクティビティからのアクティビティにをするにはてはまりません。イベントバスのユースケースをしてください。

オンラインでオットーイベントバスをむ https://riptutorial.com/ja/android/topic/6068/オットーイベントバス

151: オリエンテーションの

リファレンス https : //guides.codepath.com/android/Handling-Configuration-Changes#references

Examples

アクティビティのと

アクティビティーがしめると、システムは`onSaveInstanceState()`びして、アクティビティーがキーとのペアのでを`onSaveInstanceState()`できるようにします。このメソッドのデフォルトでは、`EditText`ウィジェットのテキストや`ListView`スクロールなど、アクティビティのビューのにするがにされます。

アクティビティののをするには、 `onSaveInstanceState()`をし、Bundleオブジェクトにキーとのペアをするがあります。えば

```
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);
    }
}
```

アクティビティがされるに、システムはそのメソッドをびします。その、システムは`onRestoreInstanceState`をびして、バンドルからをすることができます。

```
@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);
}
```

インスタンスのは、のActivityonCreateメソッドでもできますが、がすべて`onRestoreInstanceState`たことをし、サブクラスがデフォルトのをするかどうかをめる`onRestoreInstanceState`でうのがです。はこのstackoverflow postをしてください。

`onSaveInstanceState`と`onRestoreInstanceState`は`onSaveInstanceState`に`onRestoreInstanceState`はできません。 Androidは、アクティビティがされるがあるに`onSaveInstanceState()`びします。ただし、`onSaveInstanceState`

がびされてもアクティビティがされず、としてonRestoreInstanceStateがびされないonRestoreInstanceStateがあります。

フラグメントのと

フラグメントには、をするがあるときにびされるonSaveInstanceState()メソッドもあります。

```
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);
    }
}
```

に、 onCreateViewでこのされたからデ―タをりすことができ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;
   }
}
```

フラグメントをにするには、のにフラグメントをにしていないことをするがあります。これは、のフラグメントがにするに、それらをしないようにすることをします。アクティビティでされているフラグメントは、コンフィグレ―ションのにタグでするがあります。

```
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();
        }
```

```
    }
}
```

これは、トランザクションのアクティビティにフラグメントをするたびに、のタグをめるようにするがあります。

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

このなパターンでは、フラグメントをにしてのをにすことができます。

をする

くの、にをしてアクティビティーをするときのをできます。ビューとがフラグメントにある、アクティビティがされたときにフラグメントをすることはにできます

```
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;
    }
}
```

このアプローチは、アクティビティライフサイクルにフラグメントがされないようにします。わりにフラグメントマネージャーのにされます。については、Androidのドキュメントをください
。

これでフラグメントをするにそのフラグメントがタグですでにするかどうかをすることができ、フラグメントはのをえてそのをします。については、ランタイムのガイドをしてください 。

ロックの

Androidアプリケーションのアクティビティののきをしたいは、 **AndroidManifest.xml**の`<activity>`の`android:screenOrientation`プロパティをするだけです。

```
<activity
    android:name="com.techblogon.screenorientationexample.MainActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
    <!-- ... -->
</activity>
```

これで、アクティビティはに「 ポートレート 」モードでされます。

の

アプリケーションがのにリソースをするがなく、アクティビティのをするのあるパフォーマンスがあるは、アクティビティがをすることをすることができます。アクティビティ。

ただし、このテクニックはのためにをけ、ほとんどのアプリケーションではされないはのとえるがあります。このアプローチをとるには、 **AndroidManifest.xml**のアクティビティに`android:configChanges`ノードをするがあります。

```
<activity android:name=".MyActivity"
          android:configChanges="orientation|screenSize|keyboardHidden"
          android:label="@string/app_name">
```

これらのの1つがされると、アクティビティはされず、わりに`onConfigurationChanged()`びしをけります。

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

のをしてください。アクティビティでどのようなをできるかについては、 androidconfigChangesのドキュメントとConfigurationクラスをしてください。

AsyncTaskの

- `AsyncTask`にのがあると、アクティビティがされてされます。
- `AsyncTask`すると、それではないUIがされます。

ローダーをすると、のIをにすることができます。

な

```
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;
    }
}
```

v4ライブラリをするかどうかはですが、コンパイルエラーがするため、とのをしないでください。するには、`android.support.v4.content`と`android.content`のインポートをしてください。

プログラムでのをロックする

に、コードののでデバイスをロック/ロックするのににつことがよくあることはよくあることです。

えば、をつ*Dialog*をしている、はのをロックしてダイアログがされないようにするため、のアクティビティをしないようにすることができます。

のようにしてマニフェストからローテーションロックをすることができます。

```
<activity
    android:name=".TheActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
</activity>
```

のようにすれば、プログラムでもにできます。

```
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);
        }
    }
}
```

そして、のようにびして、デバイスのをロックし、ロックをします

```
lockDeviceRotation(true)
```

そして

```
lockDeviceRotation(false)
```

オンラインでオリエンテーションのをむ https://riptutorial.com/ja/android/topic/4621/オリエンテーションの

152: カウントダウンタイマー

パラメータ

パラメータ	
`long millisInFuture`	タイマーがされる、タイマーをさせるまでの。ミリ。
`long countDownInterval`	タイマーのをするです。ミリ。
`long millisUntilFinished`	CountDownTimerのりを`onTick()`でされるパラメータです。ミリ

CountDownTimerはになクラスです - それはとてもうまくいきます。 CountDownTimerを/キャンセルできるのは2のにすように、/をするがあります。よりなやにするタイマーをするには、Timerオブジェクトをします。

Examples

シンプルなカウントダウンタイマーの

CountDownTimerは、した、のでりしアクションをするにです。このでは、30テキストビューをし、りをします。タイマーがすると、TextViewに「」とされます。

```
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();
```

よりな

このでは、アクティビティのライフサイクルについてCountDownTimerをまたはします。

```
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;
}
```

オンラインでカウントダウンタイマーをむ https://riptutorial.com/ja/android/topic/6063/カウントダウンタイマー

153: カスタムRangeSeekBarのステップインクリメントをする

き

Alex FlorescuがしたAndroid RangeSeekBarのカスタマイズはhttps://github.com/anothem/android-range-seek-barです。

シークバーをかすと、ステップインクリメントをすることができます

1- attrs.xmlにをする

```
<attr name="increment" format="integer|float"/>
```

2 - RangeSeekBar.javaのデフォルトをし、もします

```
private static final int DEFAULT_INCREMENT = 1;
private int increment;
```

3-プライベートvoid initContext context、AttributeSet attrsでインクリメントをする

```
if (attrs == null)
    increment = DEFAULT_INCREMENT;
else
    increment = a.getInt(R.styleable.RangeSeekBar_increment, DEFAULT_INCREMENT);
```

4 - されたされたボイドのインクリメントをするonDraw@NonNull Canvas canvas

minTextとmaxTextのをきえるがあります。だからわりに

- minText = valueToStringgetSelectedMinValue;
- maxText = valueToStringgetSelectedMaxValue;

あなたはっているでしょう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 - すぐするがあります。それがにてばい

Examples

ステップ7をする

```
<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" />
```

オンラインでカスタムRangeSeekBarのステップインクリメントをするをむ
https://riptutorial.com/ja/android/topic/8627/カスタムrangeseekbarのステップ-インクリメント-をする

154: カスタムビュ—の

Examples

カスタムビュ—の

にカスタマイズされたビュ—がなは、 `View` すべてのAndroidビュ—のス—パ—クラスをサブクラスし、カスタムサイズ `onMeasure(...)` と `onDraw(...)` メソッドをするがあります

1. カスタムビュ—スケルトンをします。これはにすべてのカスタムビュ—でじです。ここでは `SmileyView`というスマイリ—をできるカスタムビュ—のスケルトンをします。

```
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. `Paint` する `Paint`オブジェクトは、オブジェクトのレンダリング、りつぶし、のスタイルなどをするキャンバスのブラシです。ここでは、2つの`Paint`をします.1つはのりつぶしペイントで、もう1つはのストロークペイントで、とにします。

```
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. の**`onMeasure(...)`**メソッドをします。これは、レイアウト `FrameLayout` がカスタムをにできるようにするためにです。ビュー―のさとをするためにできるの`measureSpecs`をします。ここでは、さとがじであることをしてをします

```
@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);
}
```

`onMeasure(...)`は`setMeasuredDimension(..)`へのびしがなくとも1つ`setMeasuredDimension(..)`そうしないと、カスタムビュー―が`IllegalStateException`クラッシュします。

4. の**`onSizeChanged(...)`**メソッドをします。これにより、カスタムビュー―ののさとをキャッチして、レンダリングコ―ドをにできます。ここでは、とをします。

```
@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. の**`onDraw(...)`**メソッドをします。これはビュー―ののレンダリングをするです。これはな`Canvas`オブジェクトをしますなすべてのメソッドの`Canvas`ドキュメントを。

```
@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. カスタムビュ―をレ イアウトにするカスタムビュ―をのレイアウトファイルにできるようになりました。ここでは`FrameLayout`にラップし`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>
```

ビュ―コ―ドのにプロジェクトをビルドすることをおめします。それをしなければ、Android Studioのプレビュ―でそのビュ―をることはできません。

すべてをまとめると、のレイアウトをむアクティビティをした、のがされます。



ビュ―にをする

カスタムビュ―では、Androidレイアウトリソ―スファイルでできるカスタムをすることもできます。カスタムビュ―にをするには、をするがあります。

1. のとタイプをします。これは`res/values/attrs.xml`でされ`res/values/attrs.xml` にじてします。のファイルは、ののと、ののをしています。

```
<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. レイアウトのをします。これは、カスタムをするレイアウトファイルでうことができます。のレイアウトファイルは、やかないスマイリーをむをします。

```
<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>
```

ヒントカスタムアトリビューートは、Android Studio 2.1のプレフィックスのバージョンでもの`tools:`はしません。このでは、 `app:smileyColor`を`tools:smileyColor`と、 `smileyColor`がまたはデザインにされなくなります。

3. あなたのをんでくださいこれはあなたのカスタムビューのソースコードのでわれます。`SmileyView`ののスニペットは、のをしています。

```
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.

オプションデフォルトのスタイルをこれはデフォルトのスタイルをし、それをカスタムビュ—のにみむことによってわれます。のデフォルトスマイリ—スタイルは、せなのスタイルをします。

```
<!-- styles.xml -->
<style name="DefaultSmileyStyle">
    <item name="smileyColor">#ffff00</item>
    <item name="smileyExpression">happy</item>
</style>
```

これはたちにされます`SmileyView`のびしののパラメ—タとして、それをすることにより、`obtainStyledAttributes` ステップ3のコ—ドをしてください

```
TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView,
defStyleAttr, R.style.DefaultSmileyViewStyle);
```

されたレイアウトファイルステップ2のコ—ドをにされたは、デフォルトスタイルのするをきすることにしてください。

5. オプションテ—マにスタイルをします。これは、テ—マでできるしいスタイルをし、そののスタイルをすることによってわれます。ここではにリファレンス`smileyStyle`をけます

```
<!-- attrs.xml -->
<attr name="smileyStyle" format="reference" />
```

に、たちのアプリテ—マにスタイルをしますここでは、ステップ4のデフォルトスタイルをします。

```
<!-- themes.xml -->
<style name="AppTheme" parent="AppBaseTheme">
    <item name="smileyStyle">@style/DefaultSmileyStyle</item>
</style>
```

ビュ—の

ビュ—は、のプログラムコ—ドによってのビュ—としてわれるカスタム`ViewGroup`です。このようなViewGroupは、このではContactにすることができるため、 DDDのようなデザインではにです。がされているすべてのでできます。

つまり、のコントロ—ラコ—ド、アクティビティ、フラグメント、またはアダプタは、デ—タオブジェクトをさまざまなUIウィジェットにすることなく、ビュ—ににすことができます。

これは、コ—ドのをにし、 SOLID priciplesにってよりいデザインをします。

レイアウト**XML**

これはあなたがめるです。 `<include/>`ように、あなたがしていることがかっているXMLののビットがあります。それをのXMLファイルにし、ル—トタグを`<merge>`でラップします。

```
<?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>
```

このXMLファイルは、Android Studioのレイアウトエディタでにします。のレイアウトとじようにうことができます。

ViewGroup

XMLファイルをしたら、カスタムビューグループをします。

```
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);
        }
    }
}
```

`init(Context, AttributeSet)`メソッドは、ビューへののでしたカスタムXMLをみむです。

これらのをすることで、あなたのアプリでそれをうことができます。

XMLでの

は、のContactViewをメッセージのリストのにくをす`fragment_contact_info.xml`のです

```
<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>
```

コードでの

のリストをする`RecyclerView.Adapter`のをにします。このは、Viewからにされたときにコントローラコードがどのくらいクリーナーになるかをしています。

```
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
        }
    }
}
```

CustomViewのパフォーマンスにするヒント

onDrawにしいオブジェクトをりてないでください

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = new Paint(); //Do not allocate here
}
```

キャンバスでdrawableをくわりに...

```
drawable.setBounds(boundsRect);

drawable.draw(canvas);
```

のためにビットマップをする

```
canvas.drawBitmap(bitmap, srcRect, boundsRect, paint);
```

さなだけをするためにビュ―をしないでください。わりにのをします。

```
invalidate(boundToBeRefreshed);
```

あなたのビュ―がしたアニメ―ションをしているえば、1ごとにされるウォッチフェイス`onStop()`、アクティビティの`onStop()`でアニメ―ションをし、アクティビティの`onStart()`でアニメ―ションをするがあります。

ビュ―の`onDraw`メソッドでをわないで、 `invalidate()`びすにをするがあります。このをすると、ビュ―のフレ―ムのドロップをできます。

ロ―テ―ション

ビュ―のなは、、などです。ほとんどのは、ビットマップまたはグラデ―ションをカスタムビュ―ですると、このにしています。ビュ―にビュ―がされ、そのカスタムビュ―でビットマップをさせるがある、たちのくはになるとえています。くのは、ビットマップのはにコストがかかるとえています。そのためには、ビットマップのピクセルをするがあるからです。しかし、は、それほどしいものではないということですビットマップをさせるのではなく、キャンバスをさせるだけです。

```
// 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.
```

drawableRightとしてのSVG / VectorDrawableのビュー

このビューをするなは、5.0のデバイスでは、TextView / EditTextのなデバイスでsvgをサポートしていないことです。もうつのは、々がすることができ、あるheightとwidthのdrawableRightでEditText。はそれをプロジェクトからりし、のモジュールでした。

モジュールcustom_edit_drawableの - c_d_e

"c_d_e_"は、アプリケーションモジュールのリソースがってそれらをきしないようにするためにします。 "abc"はサポートライブラリのgoogleでされています。

build.gradle

```
dependencies {
   compile 'com.android.support:appcompat-v7:25.3.1'
}
```

AppCompat> = 23をする

レイアウトファイル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>
```

カスタム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>
```

コード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();
        }
    }
}
```

のビュ―をする

レイアウト**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>
```

アクティビティ**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);
    }
}
```

タッチイベントへの

くのカスタムビュ―では、タッチイベントのでユ―ザ―のをけれるがあります。 `onTouchEvent` をオ―バ―ライドすることで、タッチイベントにアクセスすることができます。できるアクションはいくつかあります。なものはのとおりです。

- `ACTION_DOWN` これは、がにビュ―にれるとトリガ―されます。

- `ACTION_MOVE` これは、あなたのがしだけビュー—をするたびにびされます。それはもびされます。
- `ACTION_UP` これは、をからすときにびされるのアクションです。

ビュー—にのメソッドをし、でタッチしてしたときにログをできます。

```
@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;
}
```

- Androidのドキュメントタッチイベントへの

オンラインでカスタムビュー—のをむ https://riptutorial.com/ja/android/topic/1446/カスタムビュー—の

155: カスタムフォント

Examples

あなたのアプリにカスタムフォントをく

1. プロジェクトフォルダにし、
2. に、app - > src - > mainをします。
3. メインフォルダーに 'assets - > fonts'フォルダーをします。
4. あなたの 'fontfile.ttf'をfontsフォルダにれてください。

フォントの

```
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");
}
```

TextViewでのカスタムフォントの

```
public void setFont(TextView textView) {
    textView.setTypeface(myFont);
}
```

TextViewでフォントをxmlでするJavaコード

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どこにres / valuesをするか

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="TextViewPlus">
        <attr name="customFont" format="string"/>
    </declare-styleable>
</resources>
```

い

```
<?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>
```

キャンバステキストのカスタムフォント

アセットからフォントをキャンバスにします。

```
Typeface typeface = Typeface.createFromAsset(getAssets(), "fonts/SomeFont.ttf");
Paint textPaint = new Paint();
textPaint.setTypeface(typeface);
canvas.drawText("Your text here", x, y, textPaint);
```

なのみみ

カスタムフォントをみむと、パフォ―マンスがするがあります。はにこのヘルパ―をすることをおめします。これはにされているフォントをHashtableに/ロ―ドします。

```
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;
}
```

}

```
Typeface typeface = TypefaceUtils.getTypeface(context, "RobotoSlab-Bold.ttf");
setTypeface(typeface);
```

のアクティビティにするカスタムフォント

```
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();
    }
}
```

に、あなたのでは、 `onCreate()` メソッドで

```
// Put your font to assets folder...

ReplaceFont.changeDefaultFont(getApplication(), "DEFAULT", "LinLibertine.ttf");
```

Android Oのフォントの

Android Oはフォントのをします。

Android Oでは、 フォントを*XML*にするというしいがされています 。このをすると、フォントをリソースとしてできます。つまり、フォントをアセットとしてバンドルするはありません。フォントは*R*ファイルでコンパイルされ、リソースとしてシステムでににになります。

しいフォントをするには、のがです。

- しいリソースディレクトリ`res/font`します。
- このフォントフォルダにフォントファイルをします。たとえば、することによって`myfont.ttf` 、あなたがでこのフォントをすることができます`R.font.myfont`。

のXMLファイルを`res/font`ディレクトリにして、のフォントファミリをすることもできます。

```
<?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>
```

フォントファイルとフォントファミリファイルのをじでできます。

- **XML**ファイルでは、のように`android:fontFamily`をし`android:fontFamily`。

  ```
  <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:fontFamily="@font/myfont"/>
  ```

  またはのように

  ```
  <style name="customfontstyle" parent="@android:style/TextAppearance.Small">
      <item name="android:fontFamily">@font/myfont</item>
  </style>
  ```

- コードでは、のコードをします。

  ```
  Typeface typeface = getResources().getFont(R.font.myfont);
  ```

```
textView.setTypeface(typeface);
```

オンラインでカスタムフォントをむ https://riptutorial.com/ja/android/topic/3358/カスタムフォント

156: カメラ2 API

パラメーター

パラメータ	
`CameraCaptureSession`	`CameraDevice`されたキャプチャセッション。カメラからのイメージのキャプチャ、またはにじセッションでカメラからキャプチャされたイメージのにされます。
`CameraDevice`	Androidにされた1のカメラの
`CameraCharacteristics`	CameraDeviceをするプロパティ。これらのプロパティは、されたCameraDeviceにしてされ、 `getCameraCharacteristics(String)`してCameraManagerインターフェイスをしてクエリできます。
`CameraManager`	`CameraDevices`を、け、およびするためのシステムサービスマネージャ。 `Context.getSystemService()`びすと、このクラスのインスタンスをできます。
`CaptureRequest`	カメラデバイスからのイメージをキャプチャするためになとののパッケージ。キャプチャハードウェアセンサ、レンズ、フラッシュ、パイプライン、アルゴリズム、およびバッファのがまれます。このキャプチャのためにイメージデータをするターゲットサーフェスのリストもまれています。 `createCaptureRequest(int)`びしてした`CaptureRequest.Builder`インスタンスをしてできます。
`CaptureResult`	イメージセンサからののイメージキャプチャののサブセット。キャプチャハードウェアセンサー、レンズ、フラッシュ、パイプライン、アルゴリズム、およびバッファののサブセットがまれます。これは、 `CameraDevice`をしたに`CameraDevice`によってされ`CaptureRequest`

- Camera2 APIは、API 21Lollipop
- Androidにに21 + ROMがされていても、Camera2 APIをしているというはなく、するのはメーカーのですLG G2にはLollipopのサポートがありますが、Camera2 APIはありません
- Camera2では、Camera "Camera1"はされていません
- きなをもって、きなをたします。このAPIをすると、それをさせるのはです。
- えておいて、にあなたのアプリでをりたいだけなら、Camera2をするはなく、Intentをしてデバイスのカメラアプリをき、それをけることができます

Examples

メインカメラを**TextureView**でプレビュー—する

この、API 23にしてするので、もされます。

マニフェストにのをするがありますしているAPIレベルになく。

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

`TextureView`をデバイスのカメラのプレビュー—でりつぶすアクティビティCamera2Activity.javaをしようとしています。

たちがしようとしているアクティビティはなAppCompatActivityです

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

そのをするにはをむがあるかもしれません

`MAX_PREVIEW_SIZE`カメラ2 APIによっては1920×1080であります

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

`TextureView.SurfaceTextureListener`は、 `TextureView`のいくつかのライフサイクルイベントをします。この、たちはそれらのイベントにをけています。 SurfaceTextureのができたら、カメラをします。サイズがされると、それにじてカメラからのプレビュー—がされます

```
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`は、1つのデバイスのカメラをします。このでは、の`CameraDevice` IDをします

```
private String mCameraId;
```

これは、カメラのプレビューを「」するためにするビュー `TextureView` です

```
private TextureView mTextureView;
```

カメラプレビューのための`CameraCaptureSession`

```
private CameraCaptureSession mCaptureSession;
```

いている`CameraDevice`への

```
private CameraDevice mCameraDevice;
```

カメラプレビューの`Size`。

```
private Size mPreviewSize;
```

`CameraDevice.StateCallback`ときにびされ`CameraDevice`そのをします

```
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();
    }

};
```

UIをブロックすべきでないタスクをするためののスレッド

```
private HandlerThread mBackgroundThread;
```

バックグラウンドでタスクをするための`Handler`

```
private Handler mBackgroundHandler;
```

キャプチャをする`ImageReader`

```
private ImageReader mImageReader;
```

カメラプレビューの`CaptureRequest.Builder`

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

`CaptureRequest`によってされた`mPreviewRequestBuilder`

```
private CaptureRequest mPreviewRequest;
```

カメラをじるにアプリがしないようにする`Semaphore`。

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

のID

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

Androidライフサイクルメソッド

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

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

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

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

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

```
    super.onPause();
}
```

Camera2 メソッド

これらはCamera2 APIをするメソッドです

```
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);
    }
}
```

のカメラをじる

```
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();
    }
}
```

カメラのメンバをする

```
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();
    }
}
```

カメラのプレビューにしい`CameraCaptureSession`をします。

```
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();
    }
}
```

のメソッド 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);
    }
}
```

バックグラウンドスレッド/ハンドラメソッド

```
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();
    }
}
```

ユーティリティメソッド

カメラでサポートされているSizeのがあれば、それぞれのテクスチャビューのサイズとしてなくともきいものをし、それぞれのサイズとじきさで、アスペクトがされたとするものをします。しないは、それぞれのサイズとじきさで、アスペクトがされたとするのものをします

```
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];
    }
}
```

このメソッドは、mTextureViewへのmTextureView Matrixを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);
}
```

このは、2つの`Size`をそのについてします。

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

あまりない

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

オンラインでカメラ2 APIをむ https://riptutorial.com/ja/android/topic/619/カメラ2-api

157: カメラとギャラリー

Examples

カメラからフルサイズのをる

をるには、まず`AndroidManifest.xml`なをするがあります。 2つのがです。

- `Camera` - カメラアプリをきます。`required`が`true`されて`true`、ハードウェアカメラをたないはこのアプリケーションをインストールできません。
- `WRITE_EXTERNAL_STORAGE` - このは、キャプチャされたがされるしいファイルをするためにです。

AndroidManifest.xml

```
<uses-feature android:name="android.hardware.camera"
          android:required="true" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

カメラからフルサイズのをることのなアイデアは、カメラアプリをいてをるに、しいファイルをするがあるということです。

```
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();
    }
}
```

をる

カメラにアクセスするをAndroidManifestファイルにします

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

XMLファイル

```
<?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>
```

アクティビティ

```
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;
    }
}
```

カメラやギャラリーをし、カメラのをストレージにする

まずUriとtempフォルダとコードがです。

```
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;
```

に、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();
    }
}
```

CameraとGalleryをく

```
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);
    }

}
```

に`onActivityResult`メソッドで

```
@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);
}
```

あなたは`AndroidManifest.xml` theeseのアクセスがです

```
<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" />
```

また、のみみ/きみなどのアクセスをするがありまする。

は`openGallery`メソッドで`READ_EXTERNAL_STORAGE`をチェックしてい`openGallery`

の`requestPermission`メソッド

```
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);
    }
}
```

に、 `onRequestPermissionsResult`メソッドをオーバーライドします。

```
@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`メソッド

```
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();
}
```

カメラのをする

プログラムでをします。

```
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);
```

とフェッチされた**URI**からしくされたビットマップをデコ―ドします。

```
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;
}
```

オンラインでカメラとギャラリーをむ https://riptutorial.com/ja/android/topic/4789/カメラとギャラリー

158: キーボード

Examples

ユーザーがのをタップするとキーボードをす

アクティビティにコードをします。

これは**Fragment**でもしますが、 **Fragment**にこのコードをするはありません 。

```
@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);
}
```

キーボードのオープンとクローズのためのコールバックをする

アイデアはのレイアウトをすることです。ながあるは、そのソフトキーボードがいくらかかめられます。

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

`onCreate`で`mLastContentHeight`をします

```
mLastContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();
```

リスナーをします

```
rootView.getViewTreeObserver().addOnGlobalLayoutListener(keyboardLayoutListener);
```

`destroy`リスナーをすることをれないでください

```
rootView.getViewTreeObserver().removeOnGlobalLayoutListener(keyboardLayoutListener);
```

オンラインでキーボードをむ https://riptutorial.com/ja/android/topic/5606/キーボード

159: グーグルプレイストア

Examples

アプリのGoogle Playストアのリスティングをく

のコードスニペットは、なでアプリのGoogle Playストアのリスティングをくをしています。、ユーザーにあなたのアプリのレビューをすようにするときにします。

```
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);
}
```

このコードは、アプリがインストールされているはGoogle Playストアをきます。それのは、Webブラウザがきます。

サイトのアカウントのすべてのアプリケーションのリストをむGoogle Playストアをきます

Google Playストアアプリですべてのサイトアプリケーションをして、アプリで[のアプリを]ボタンをできます。

```
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);
    }
}
```

オンラインでグーグルプレイストアをむ https://riptutorial.com/ja/android/topic/10900/グーグルプレイストア

160: クラッシュレポートツール

のなwikiはgithubでここにあります。

Examples

ファブリック - クラッシュ-スティックス

Fabricは、アプリケーションをするためにみわせることができるなキットをするモジュールモバイルプラットフォームです。 **Crashlytics**は、アプリケーションをにおよびできるようにする、Fabricによってされるクラッシュおよびのレポートツールです。

Fabric-Crashlyticsの

ステップ**1** `build.gradle`する

プラグインレポと*gradle*プラグインをしてください

```
 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.+'
  }
}
```

プラグインをする

```
apply plugin: 'com.android.application'
//Put Fabric plugin after Android plugin
apply plugin: 'io.fabric'
```

*Fabric Repo*をします。

```
repositories {
  maven { url 'https://maven.fabric.io/public' }
}
```

*Crashlyrics*キットをする

```
dependencies {
```

```
  compile('com.crashlytics.sdk.android:crashlytics:2.6.6@aar') {
    transitive = true;
  }
}
```

2 `AndroidManifest.xml` **API**キーとインターネットアクセスをする

```
<?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>
```

ステップ**3**あなたのコードでにキットをします。えば

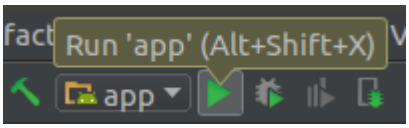
```
public class MainActivity extends ActionBarActivity {

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

      //Init the KIT
      Fabric.with(this, new Crashlytics());

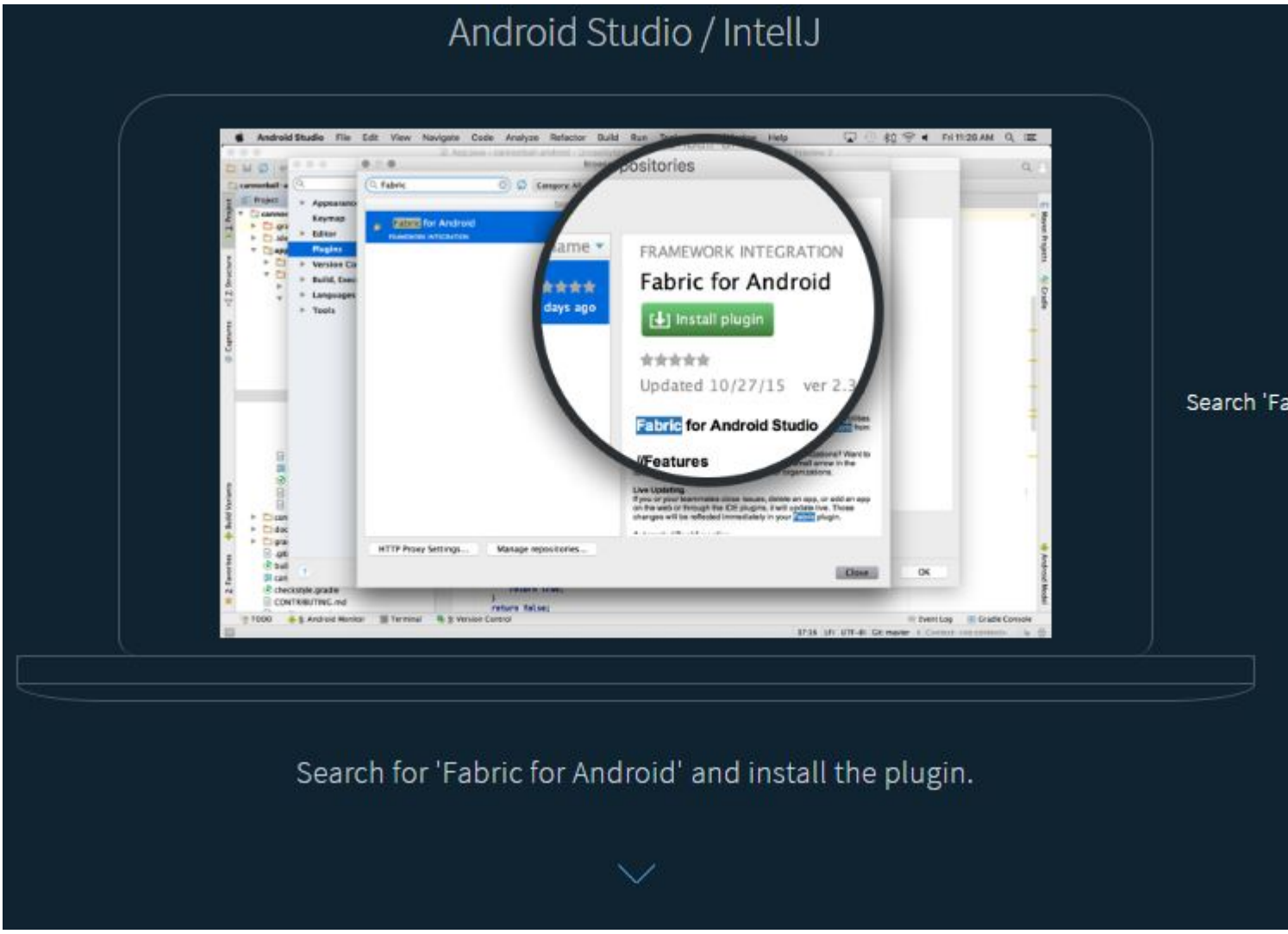
      setContentView(R.layout.activity_main);
    }
}
```

ステップ**4**プロジェクトをビルドします。ビルドしてするには



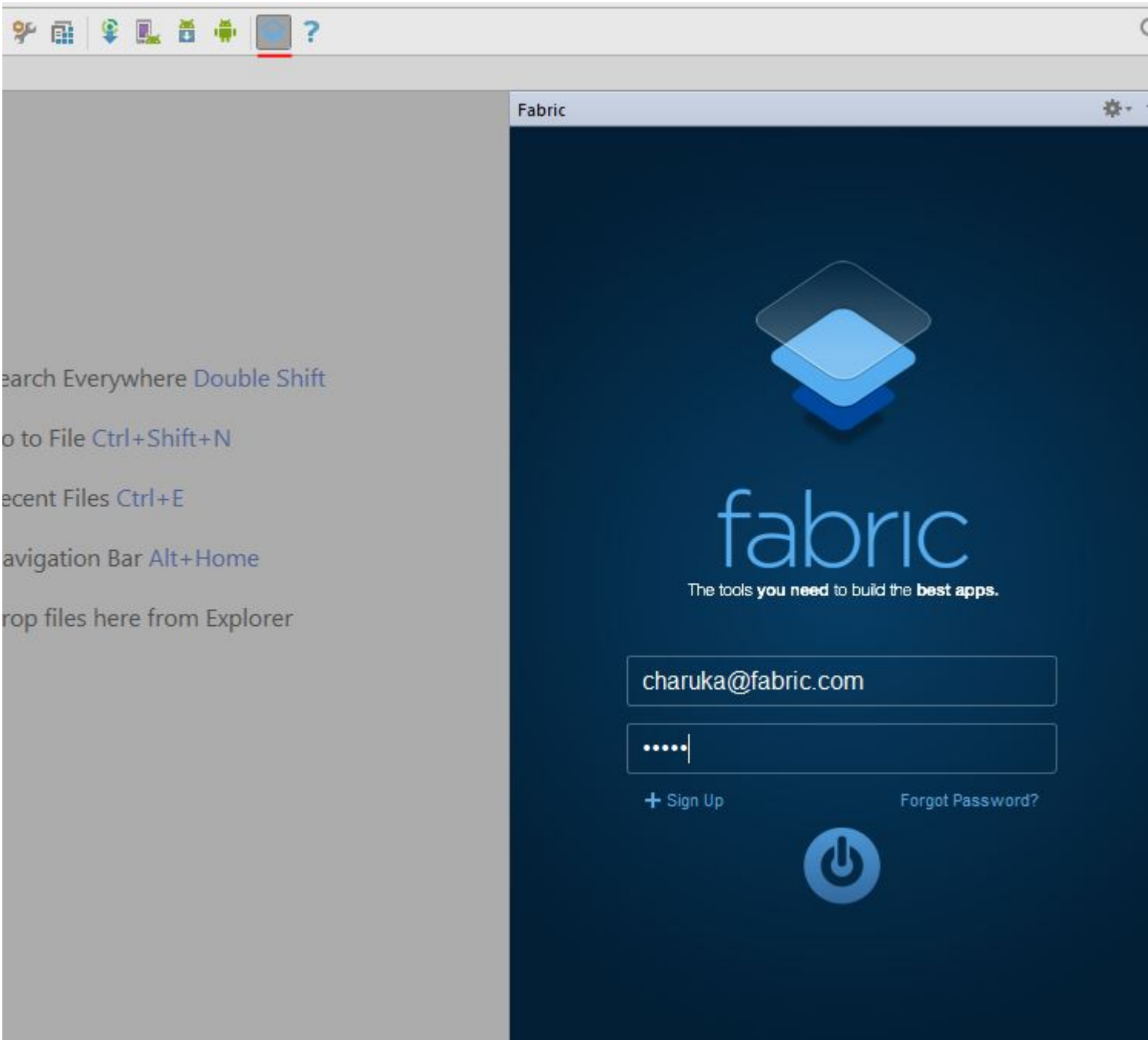
Fabric IDEプラグインの

キットは、 このリンクにいてAndroid StudioまたはIntelliJのFabric IDEプラグインをしてインストールできます。



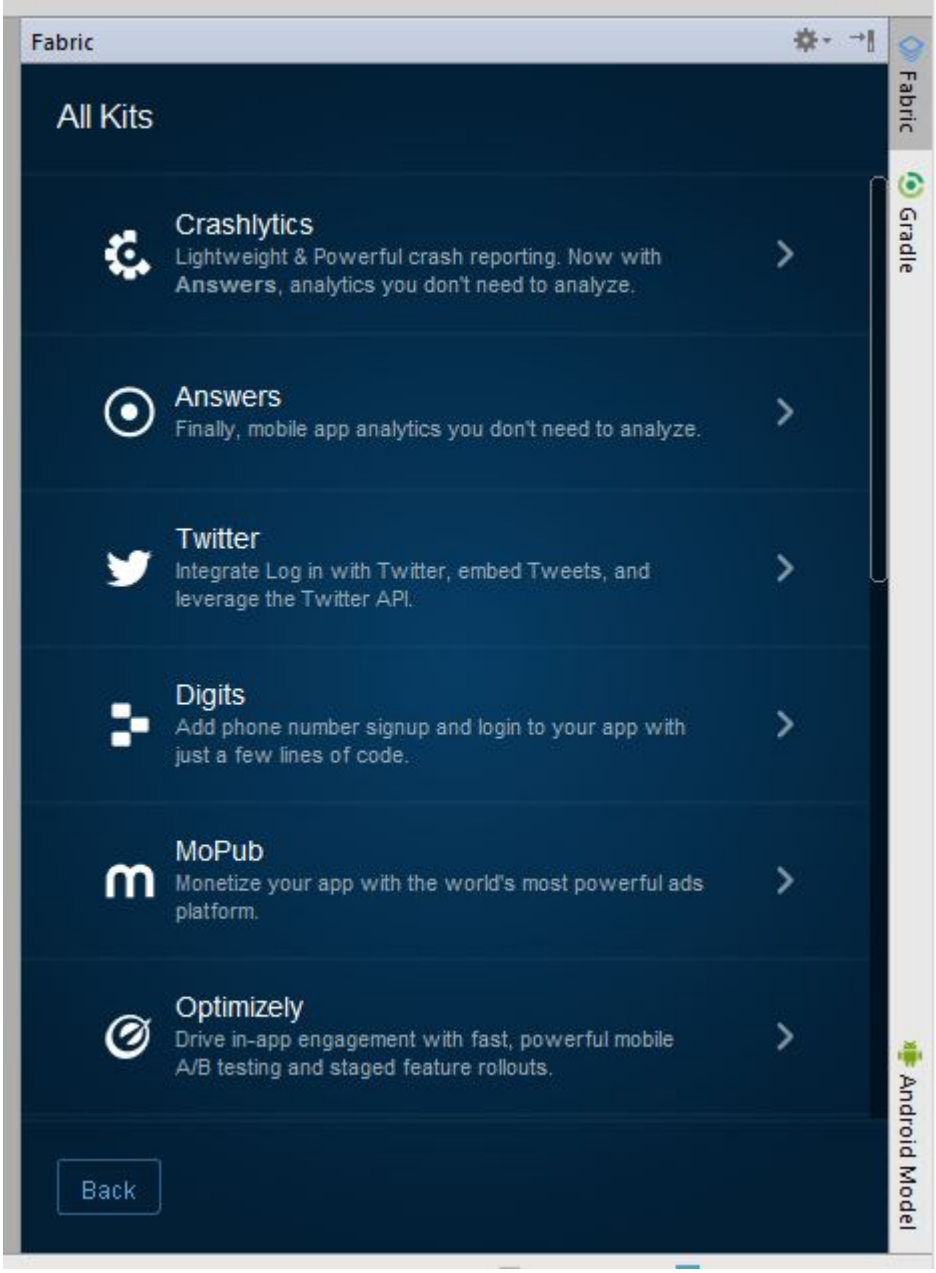
プラグインをインストールしたら、Androidのメーカーをすると**Android Studio**をしてアカウントにログインします。

ショートキー> `CTRL + L`

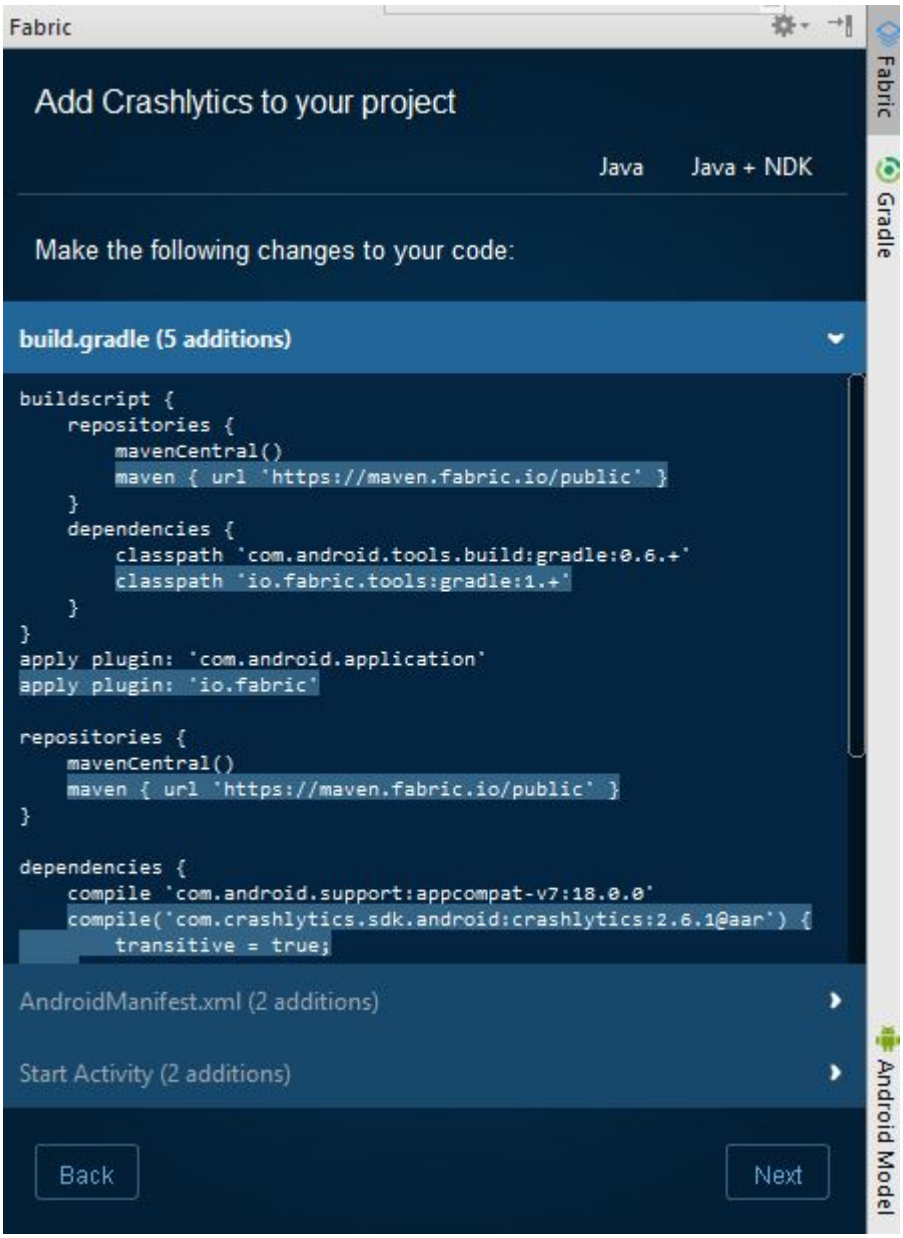


に、あなたがっているプロジェクト/あなたがいたプロジェクトをし、なプロジェクトをして、へをクリックします。

したいキットをしてください。そのは**Crashlytics**です



その、 `Install` 。は**gradle**プラグインのようにでするはありませんが、わりにあなたのためにされます。



ACRAによるクラッシュレポート

ステップ1のACRA AARのをアプリケーションのgradlebuild.gradleにします。

ステップ2アプリケーションクラスApplicationをするクラス、それをしない `@ReportsCrashes` アノテーションをし、 `attachBaseContext()` メソッドをオーバーライドします。

3アプリケーションクラスでACRAクラスをする

```
@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);
            }
    }
```

AUTH_USERNAMEとAUTH_PASSWORDは、なバックエンドのです。

ステップ4AndroidManifest.xmlにApplicationクラスをする

```
 <application
        android:name=".MyApplication">
        <service></service>
        <activity></activity>
        <receiver></receiver>
 </application>
```

ステップ5クラッシュしたアプリケーションからレポートをけるinternetアクセスがあることをする

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

サイレントレポートをバックエンドにするは、のをしてしてください。

```
ACRA.getErrorReporter().handleSilentException(e);
```

ファブリックでテストクラッシュをする

タップしてクラッシュをきこすことができるボタンをします。このコードをレイアウトにりけ、ボタンをします。

```
<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" />
```

RuntimeExceptionをスローする

```
public void forceCrash(View view) {
    throw new RuntimeException("This is a crash");
}
```

アプリをしてしいボタンをタップすると、クラッシュするがあります。 12で、Crashlyticsダッシュボードでクラッシュをできるだけでなく、メールをけることができます。

Sherlockをしてクラッシュをキャプチャする

Sherlockはすべてのクラッシュをし、としてします。をタップすると、クラッシュのがすべてされ、デバイスとアプリケーションのがされます

あなたのアプリケーションに**Sherlock**をするには

あなたはあなたのプロジェクトにgradleのとしてSherlockをするだけです。

```
dependencies {
    compile('com.github.ajitsing:sherlock:1.0.1@aar') {
        transitive = true
    }
}
```

Androidスタジオをした、アプリケーションクラスでSherlockをします。

```
package com.singhajit.login;

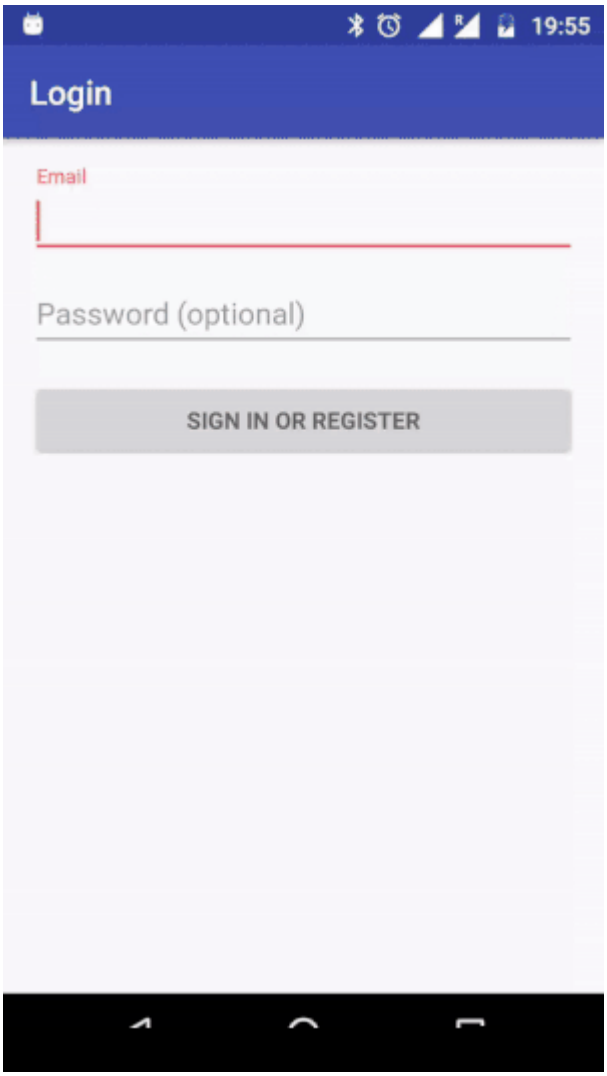
import android.app.Application;

import com.singhajit.sherlock.core.Sherlock;

public class SampleApp extends Application {
  @Override
  public void onCreate() {
    super.onCreate();
    Sherlock.init(this);
  }
}
```

それはあなたがするがあることすべて。また、Sherlockはクラッシュをするだけではありません。すべてのをチェックアウトするには、このをください 。

デモ



オンラインでクラッシュレポートツールをむ https://riptutorial.com/ja/android/topic/3871/クラッシュレポートツール

161: コーディネーターのレイアウトとき

CoordinatorLayoutはなFrameLayoutであり、このViewGroupのはにあるビューをすることです。

CoordinatorLayoutのなは、XMLファイルのビューのアニメーションとトランジションをするです。

CoordinatorLayoutは、2つのなをとしています。

のアプリケーションのまたはクロームのレイアウト

1つのビューとののののためのコンテナとして

`CoordinatorLayout`は、 `FrameLayout`をするコンテナです。
りけることで`CoordinatorLayout.Behavior`ののに`CoordinatorLayout` 、あなたはタッチイベント、ウィンドウインセット、、レイアウト、およびネストされたスクロールをできるようになります。

`CoordinatorLayout`ビューの`Behaviors`をすることにより、のにくのなるインタラクションをすることができ、それらのビューはにやりとりすることもできます。ビュークラスは、 `DefaultBehavior` アノテーションをして`CoordinatorLayout`としてする、デフォルトのをできます。

Examples

シンプルなをする

`Behavior`をするには、 `CoordinatorLayout.Behavior`クラスをするだけです。

CoordinatorLayout.Behaviorをする

```
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);
    }
}
```

このは、びされる`CoordinatorLayout`ビューにアタッチするがあります。

プログラムでをアタッチする

```
MyBehavior myBehavior = new MyBehavior();
CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams)
view.getLayoutParams();
params.setBehavior(myBehavior);
```

XMLでビヘイビアをアタッチする

`layout_behavior`をして、ビヘイビアをXMLにすることができます。

```
<View
  android:layout_height="...."
  android:layout_width="...."
  app:layout_behavior=".MyBehavior" />
```

をにアタッチする

カスタムビューでしているは、 `@CoordinatorLayout.DefaultBehavior`アノテーションをしてビヘイビアをアタッチできます。

```
@CoordinatorLayout.DefaultBehavior(MyBehavior.class)
public class MyView extends ..... {

}
```

SwipeDismissBehaviorの

`SwipeDismissBehavior`はのビューでし、 `CoordinatorLayout`してレイアウトでスワイプをします。

ちょうどする

```
        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);
```

ビュー—のをする

ビュー—のをするには、 CoordinatorLayout.Behaviorをします。 のView Viewをするには、のをいView 。

- layout_anchorをします。
- カスタムBehaviorをし、 trueすlayoutDependsOnメソッドをしtrue。

たとえば、のツー—ルバー—がされたときにImageViewをするためのBehaviorをするにはツー—ルバー—、のをします。

- カスタムビヘイビアをします 。

```
public class MyBehavior extends CoordinatorLayout.Behavior<ImageView> {...}
```

- trueすlayoutDependsOnメソッドをオー—バー—ライドしtrue。 このメソッドは、レイアウトにがするたびにびされます。

```
@Override
public boolean layoutDependsOn(CoordinatorLayout parent,
        ImageView child, View dependency) {
    // Returns true to add a dependency.
    return dependency instanceof Toolbar;
}
```

- たびlayoutDependsOnすtrueがonDependentViewChangedとばれています。

```
@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);
}
```

オンラインでコーディネーターのレイアウトとをむ https://riptutorial.com/ja/android/topic/5714/コーディネーターのレイアウトと

162: コールバックURL

Examples

Instagram OAuthによるコールバックURLの

コールバック*URL*のの1つはOAuthです。たちは、Instagramのログインでこれをやってみましょうユーザーがのをし、 ログインボタンをクリックした、Instagramのは、をし、します `access_token` 。 たちのアプリでは、 `access_token` がです。

アプリがそのようなリンクをくことができるようにするには、コールバックURLを`Activity`にするがあります。 `<intent-filter/>`を`Activity`にすると、そのコールバックURLにします。コールバックURLは`appSchema://appName.com`ます。 に、 *Manifest.xml*ファイルののの`Activity`にのをするがあります。

```
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.BROWSABLE"/>
<data android:host="appName.com" android:scheme="appSchema"/>
```

のの

- `<category android:name="android.intent.category.BROWSABLE"/>`ターゲットアクティビティは、リンクによってされるデータをするためにWebブラウザによってされることをします。
- `<data android:host="appName.com" android:scheme="appSchema"/>`は、 `<data android:host="appName.com" android:scheme="appSchema"/>`コールバックURLのスキーマとホストをします。
- これらのはすべて、ブラウザでコールバックURLがびされるたびに、の`Activity`をくようにします。

さて、あなたの`Activity`でURLのをするには、のように`onResume()`メソッドをオーバーライドするがあります

```
@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.
}
```

これでInstagramから`access_token`がされました。これはInstagramのさまざまなAPIエンドポイントでされています。

オンラインでコールバックURLをむ https://riptutorial.com/ja/android/topic/4790/コールバックurl

163: コンテキスト

き

Googleのドキュメントでは、「アプリケーションにするグローバルなへのインターフェイス。アプリケーションのリソースやクラスにアクセスできるほか、アクティビティ、やインテントなどのアプリケーションレベルののアップコールがです」

よりにえば、Contextはアプリケーションののです。オブジェクトにをすることで、アプリケーションののでがこっているのかをることができます。

- `getApplicationContext()`
- `getBaseContext()`
- `getContext()`
- `this`

このStackOverflowページには、コンテキストののでよくかれたがいくつかあります。

コンテキストとはですか

Examples

な

アクティビティの

```
Context context = getApplicationContext();
```

フラグメントの

```
Context context = getActivity().getApplicationContext();
```

`this` アプリケーション、アクティビティ、サービス、IntentServiceクラスなど、Contextからされたクラスの

```
TextView textView = new TextView(this);
```

の`this`

```
Intent intent = new Intent(this, MainActivity.class);
startActivity(intent);
```

オンラインでコンテキストをむ https://riptutorial.com/ja/android/topic/9774/コンテキスト

164: サービス

き

サービスは、されるをしたり、リモートプロセスのをするためにバックグラウンドでされます。サービスは、ユーザーのでバックグラウンドでのみされるユーザーインターフェイスをしません。たとえば、ユーザーがのAppにいるにバックグラウンドでをしたり、Androidデバイスとのユーザーのやりとりをげることなくインターネットからデータをダウンロードしたりすることができます。

AndroidManifest.xmlでサービスをしていない、サービスをしようとすると`ServiceNotFoundException`がされます。

`IntentService`にするは、 `IntentService`してください IntentServiceの

Examples

サービスの

サービスをすることはにです。アクティビティからをって`startService`をびすだけです

```
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.
```

サービスのライフサイクル

サービスライフサイクルにはのコールバックがあります

- `onCreate()`

サービスがにされたときにされ、なをします。このメソッドは、サービスがまだされていないにのみされます。

- `onStartCommand()`

`startService()`がアクティビティやBroadcastReceiverのようなのコンポーネントによってびされるたびにされ`startService()`。 このメソッドをすると、 `stopSelf()`または`stopService()`をびすまでサービスがされます。それになく、あなたがびしの`onStartCommand()`メソッド`stopSelf()`と`stopService()`サービスをするために、だけするがあります。

- `onBind()`

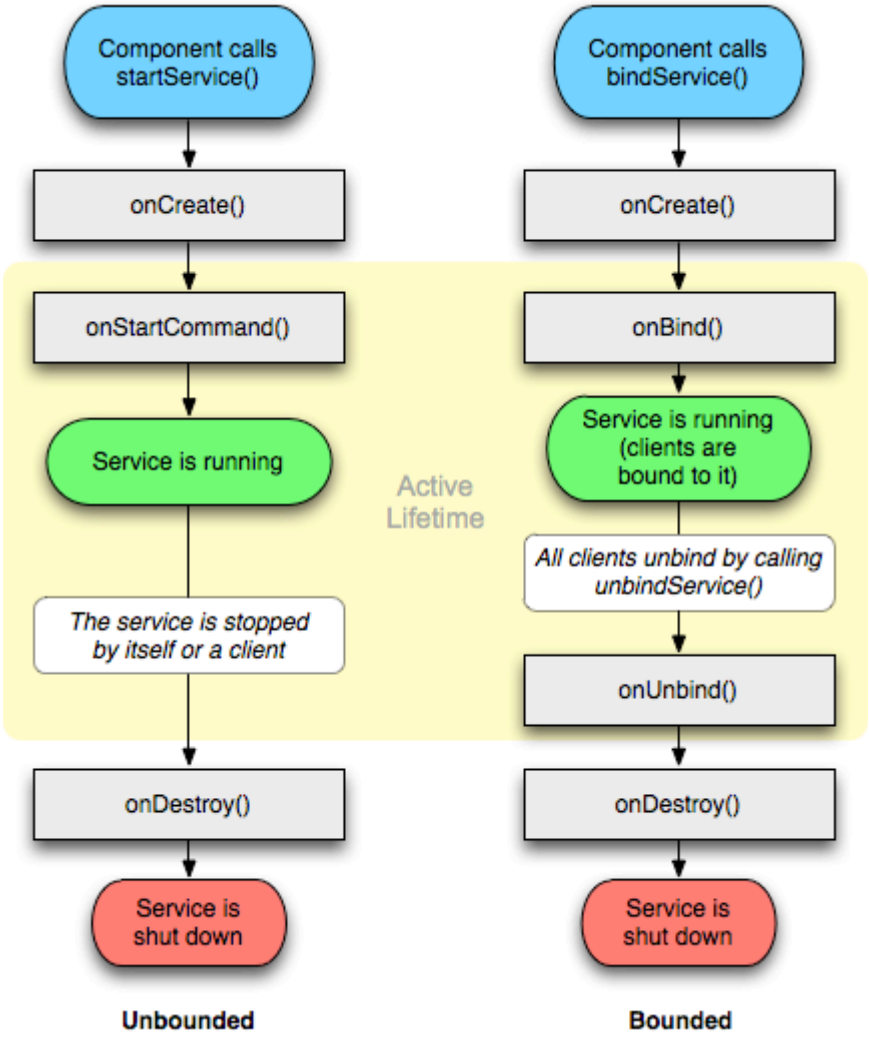
コンポーネントが`bindService()`をびして`bindService()`のインスタンスをし、チャネルをサービスにするときにされます。 `bindService()`をびすと、バインドされたクライアントがあるり、サービスはされたままになります。

- `onDestroy()`

サービスがもはやされなくなったときにされ、りてられたリソースのがになります。

サービスのライフサイクルに、 `onConfigurationChanged()`や`onLowMemory()`などののコールバックがびされるがあることにすることはです。

https://developer.android.com/guide/components/services.html



サービスのプロセスの

`android:process`フィールドは、サービスがされるプロセスのをします。、アプリケーションのすべてのコンポーネントは、アプリケーションにされたデフォルトのプロセスでされます。ただし、コンポーネントはのプロセスをしてデフォルトをオーバーライドすることができ、アプリケーションをのプロセスにすることができます。

このにりてられたがコロン":"でまる、サービスはそれののプロセスでされます。

```
<service
  android:name="com.example.appName"
  android:process=":externalProcess" />
```

プロセスがでまる、サ―ビスはそののグロ―バルプロセスでされます。これにより、なるアプリケ―ションのコンポ―ネントがプロセスをし、リソ―スのをできます。

バインダ―のけをりてバインドされたサ―ビスをする

`Service`クラスをするクラスをし、オ―バ―ライドされたメソッド`onBind`て、ロ―カルのバインダ―インスタンスをします。

```
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;
    }
}
```

に、あなたのアクティビティで`onStart`コ―ルバックでサ―ビスにバインドし、 `ServiceConnection` インスタンスをして`onStop`バインドをします。

```
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;
        }
    };
}
```

リモートサービスのAIDL

.aidlファイルをしてサービスアクセスインターフェースを.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();
}
```

すぐアプリケーションをビルドしたら、sdkツールはな.javaファイルをします。このファイルには、aidlインタフェースをし、するがあるStubクラスがまれます

```
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;
    }
}
```

その、

```
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;
    }
}
```

アンバウンドサービスの

まず、 `<application>`タグの`AndroidManifest.xml`にサービスをします。

```
<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>
```

のパッケージ.AllServices.RecordingServiceでサービスクラスをするは、サービスがどこにあるかをするがあります。したがって、の、のようにします。

```
android:name=".RecordingService"
```

に

```
android:name=".AllServices.RecordingService"
```

そうするもなは、なパッケージをすることです。

に、のサービスクラスをします。

```
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();
    }

}
```

このサ―ビスは、にをし、 `doSomething()` メソッドがびされたときにト―ストをすることができます。

あなたがづくように、それはシングルトンとしてされ、のインスタンスをしますが、のシングルトンファクトリメソッドはありません。なぜなら、サ―ビスはにシングルトンであり、によってされるからです。このインスタンスは、にサ―ビスへの「ハンドル」をるためににちます。

に、アクティビティからサ―ビスをおよびするがあります。

```
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);
    }
}
```

このでは、サービスはのにじてじでおよびします。

`doSomething()`メソッドをアクティビティからびすこともできます

```
public void makeServiceDoSomething(){
    if( RecordingService.isRunning )
        RecordingService.instance.doSomething();
}
```

オンラインでサービスをむ https://riptutorial.com/ja/android/topic/137/サービス

165: さまざまな、サイズでをサポート

と

サイズ

スクリーンのとしてされたのなサイズ。にするため、Androidはすべてののサイズを4つのされたサイズ、、、にグループします。

スクリーン

ののピクセル。はdpidots per inchとばれます。えば、「」スクリーンは、「」または「」スクリーンとして、のエリアのピクセルがよりない。にするために、Androidはすべてののを、、、、、の6つのされたにグループします。

オリエンテーション

ユーザーのからたのき。これはまたはのいずれかです。つまり、のアスペクトはそれぞれワイドまたはです。なるデバイスがデフォルトでなるでするだけでなく、ユーザーがデバイスをさせるときににきがわるがあることにしてください。のピクセルの。ののサポートをする、アプリケーションはでしません。アプリケーションは、なサイズとのグループでされているように、のサイズとにのみするがあります。ピクセルdpUIレイアウトをするときにするピクセルで、レイアウトのやにしないでをす。にしないピクセルは、160dpiのの1つのピクセルにします。これは、「」ののでシステムがしているベースラインのです。に、システムはのののにづいて、にじてdpユニットのスケーリングをにします。 dpのピクセルへのはですpx = dp *dpi / 160。えば、240dpiのスクリーンでは、1dpは1.5ピクセルにしい。アプリケーションのUIをするときには、dpをにして、なるのにUIがしくされるようにするがあります。

- **px**

  ピクセル - ののピクセルにします。

- に

  インチ - のなサイズにづいています。 1インチ= 2.54センチメートル

- **mm**

  ミリメートル - のなサイズにづいています。

- **pt**

  ポイント - スクリーンのなサイズにづいて1/72インチ。

- **dpまたはdip**

  Density-Independent Pixels - のにづくな。これらのは160 dpiのをにしているため、1 dpは160 dpiのの1ピクセルです。 dpとピクセルのはのにじてしますが、ずしもではありません。コンパイラは "dip"と "dp"のをけれますが、 "dp"は "sp"としています。

- **sp**

  にされないピクセル - これはdpとていますが、ユーザーのフォントサイズのによってもされます。フォントサイズをするときは、をすることをおめします。そのため、のとユーザーのみのでされます。 Androidののから

		インチたりの		すべてのでじサイズ
px	ピクセル		いいえ	いいえ
に	インチ	1	はい	はい
mm	ミリメートル	25.4	はい	はい
pt	ポイント	72	はい	はい
dp	ピクセル	160	はい	いいえ
sp	スケールピクセル	160	はい	いいえ

- https://developer.android.com/guide/practices/screens_support.html
- http://developer.android.com/guide/topics/resources/more-resources.html

Examples

の

Androidでは、のデバイスのについてシステムがリソースをするをできるように、いくつかのがサポートされています。は、Androidプロジェクトのリソースディレクトリにできるで、のリソースがされているをします。

をするには、のようにします。

1. プロジェクトのres /ディレクトリにしいディレクトリをし、 `<resources_name>-<qualifier>` でをけます。 `<resources_name>` は、リソースdrawableやlayoutなどです。
2. `<qualifier>` はで、これらのリソースがされるhdpiまたはxlargeなどをします。

たとえば、のアプリケーションリソースディレクトリは、さまざまなサイズとなるなレイアウトをします。ランチャーアイコンには、 `mipmap/` フォルダーをします。

```
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
```

dpとspをピクセルにする

`Paint.setTextSize` などのピクセルをするがあるでも、デバイスについてスケーリングするがあるは、dpとspをできます。

```
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);
```

また、リソースをロードするコンテキストがあるは、ディメンションリソースをピクセルにすることもできます。

```
<?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);
```

テキストサイズとなるAndroidサイズ

によっては、3つのオプションしかたないがよいもあります

```
 style="@android:style/TextAppearance.Small"
 style="@android:style/TextAppearance.Medium"
 style="@android:style/TextAppearance.Large"
```

のサイズとするために、さいものときいものをします。

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="@android:style/TextAppearance.Small"/>
```

の、もするはありません。

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

これをすると、さまざまなサイズのディメンションをテストしてすることをけることができます。

オンラインでさまざまな、サイズでをサポートをむ https://riptutorial.com/ja/android/topic/1086/さまざまな-サイズでをサポート

166: ジェスチャー

ジェスチャーの

Examples

スワイプ

```
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() {
    }

}
```

ビューに...

```
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");
    }

});
```

ジェスチャー

```
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;
    }

}
```

オンラインでジェスチャーをむ https://riptutorial.com/ja/android/topic/4711/ジェスチャー

167: システムフォントのとフォントの

き

のは、 */ system / fonts /*ディレクトリにされているシステムフォントのデフォルトをすると、システムフォントをして`TextView`のをするをしています。

Examples

システムフォントの

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

TextViewにシステムフォントをする

のコードでは、 `fontsname`をするフォントのにきえるがあります。

```
TextView lblexample = (TextView) findViewById(R.id.lblexample);
lblexample.setTypeface(Typeface.createFromFile("/system/fonts/" + "fontsname" + ".ttf"));
```

オンラインでシステムフォントのとフォントのをむ https://riptutorial.com/ja/android/topic/10930/システムフォントのとフォントの

168: ジャクソン

き

JacksonはJSONをするためのJavaライブラリです。ジャクソンはにとって、、、にづいたなみわせをしています。

ジャクソンの

マルチモード、にいコラボレーション

だけでなく、

ジェネリックをにサポート

のサポート

Examples

なデータバインドの

JSONデータ

```
{
  "name" : { "first" : "Joe", "last" : "Sixpack" },
  "gender" : "MALE",
  "verified" : false,
  "userImage" : "keliuyue"
}
```

それを**User**インスタンスにするには、2の**Java**がです。

```
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;
    }
}
```

JSONにマーシャリングすることもにです。

```
mapper.writeValue(new File("user-modified.json"), user);
```

オンラインでジャクソンをむ https://riptutorial.com/ja/android/topic/10878/ジャクソン

169: ショートカットマネージャ

Examples

ダイナミックランチャーショートカット

```
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));
```

たちはすべてのショートカットをにすることができます -

```
 shortcutManager.removeAllDynamicShortcuts();
```

のDynamic Shorcutsをするには

```
shortcutManager.updateShortcuts(Arrays.asList(shortcut);
```

`setDynamicShortcuts(List)`はショートカットのリストをするためにされ、
`addDynamicShortcuts(List)`はショートカットをのショートカットのリストにするためにされます

オンラインでショートカットマネージャをむ https://riptutorial.com/ja/android/topic/7661/ショートカットマネージャ

170: ジョブスケジューリング

`onStartJob()`ように、くのコードをしたり、 `JobService`でのをしたりすることにはしてください。このコードはメイン/ UIスレッドでされるため、ブロックされたUI、していないアプリケーション、またはアプリケーションのクラッシュにつながるがあります。

そのため、たとえば`Thread`または`AsyncTask`をしてをオフロードするがあります。

Examples

な

しいJobServiceをする

これは、 `JobService`クラスをし、なメソッドを`onStartJob()`および`onStopJob()`またはオーバーライドすることによってわれます。

```
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;
    }
}
```

AndroidManifest.xmlにしいJobServiceをします。

のはです 。そうしないと、ジョブをできなくなります。

*AndroidManifest.xml*の `<application> </application>`にしい`<service>`として`MyJobService`クラスをします。

```
<?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>
```

ジョブのセットアップと

しいJobServiceをして*AndroidManifest.xml*にしたら、のをできます。

- `onButtonClick_startJob()`は、なジョブをしてします。 `JobInfo.Builder`では、なジョブにも、のくのやをできます。たとえば、ジョブをするためにまたはネットワークがしまれていることをできます。
- `onButtonClick_stopJob()`は、のすべてのジョブをキャンセルします。

```
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
    }
}
```

`onButtonClick_startJob()`びした、アプリケーションはユーザーがホームボタンをしてアプリケーションがされなくなってもであっても、10でされます。

`onButtonClick_stopJob()`ののジョブをすべてキャンセルするわりに、 `jobScheduler.cancel()`をびして、ジョブIDについてのジョブをキャンセルすることもできます。

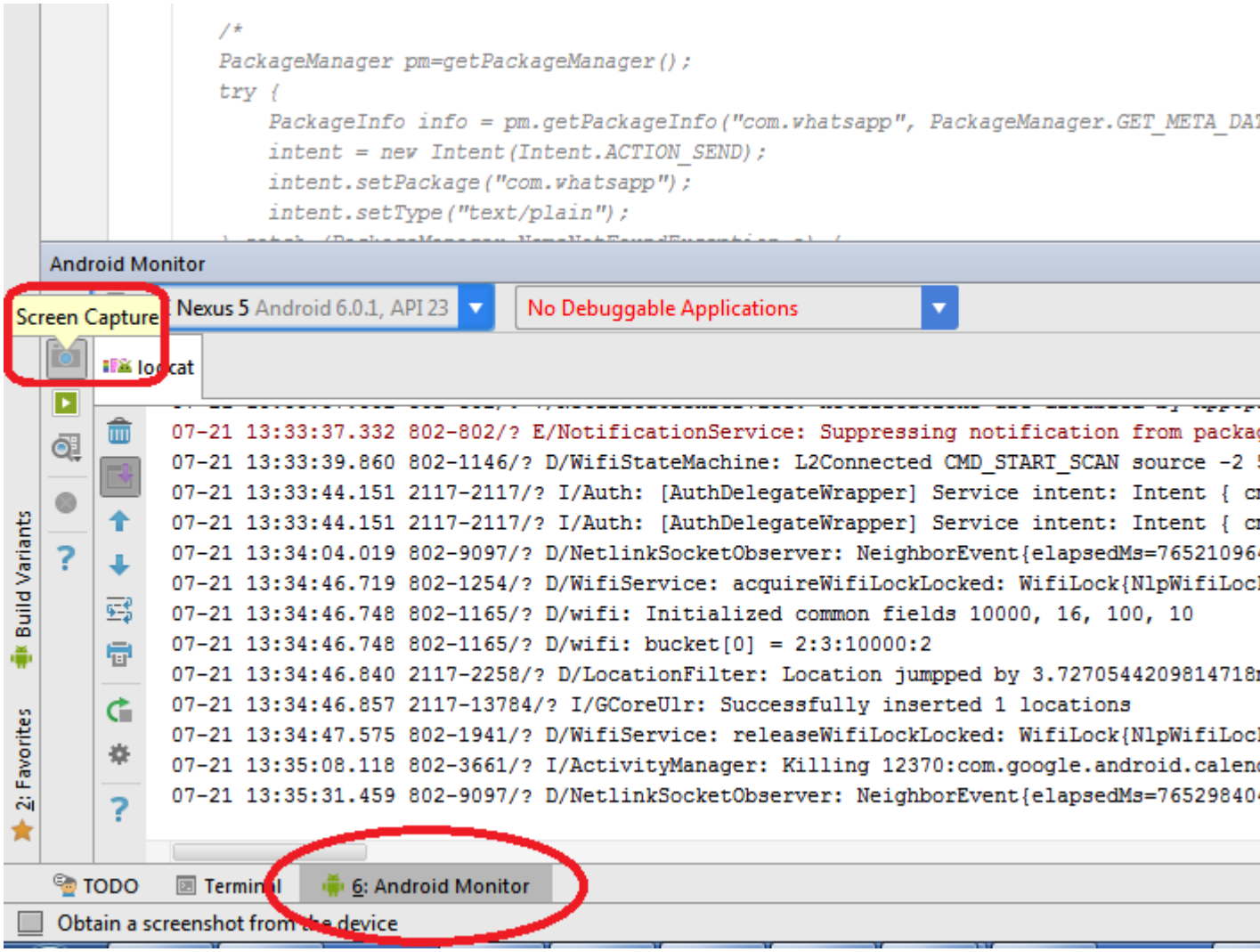
オンラインでジョブスケジューリングをむ https://riptutorial.com/ja/android/topic/6907/ジョブスケジューリング

171: スクリーンショットのキャプチャ

Examples

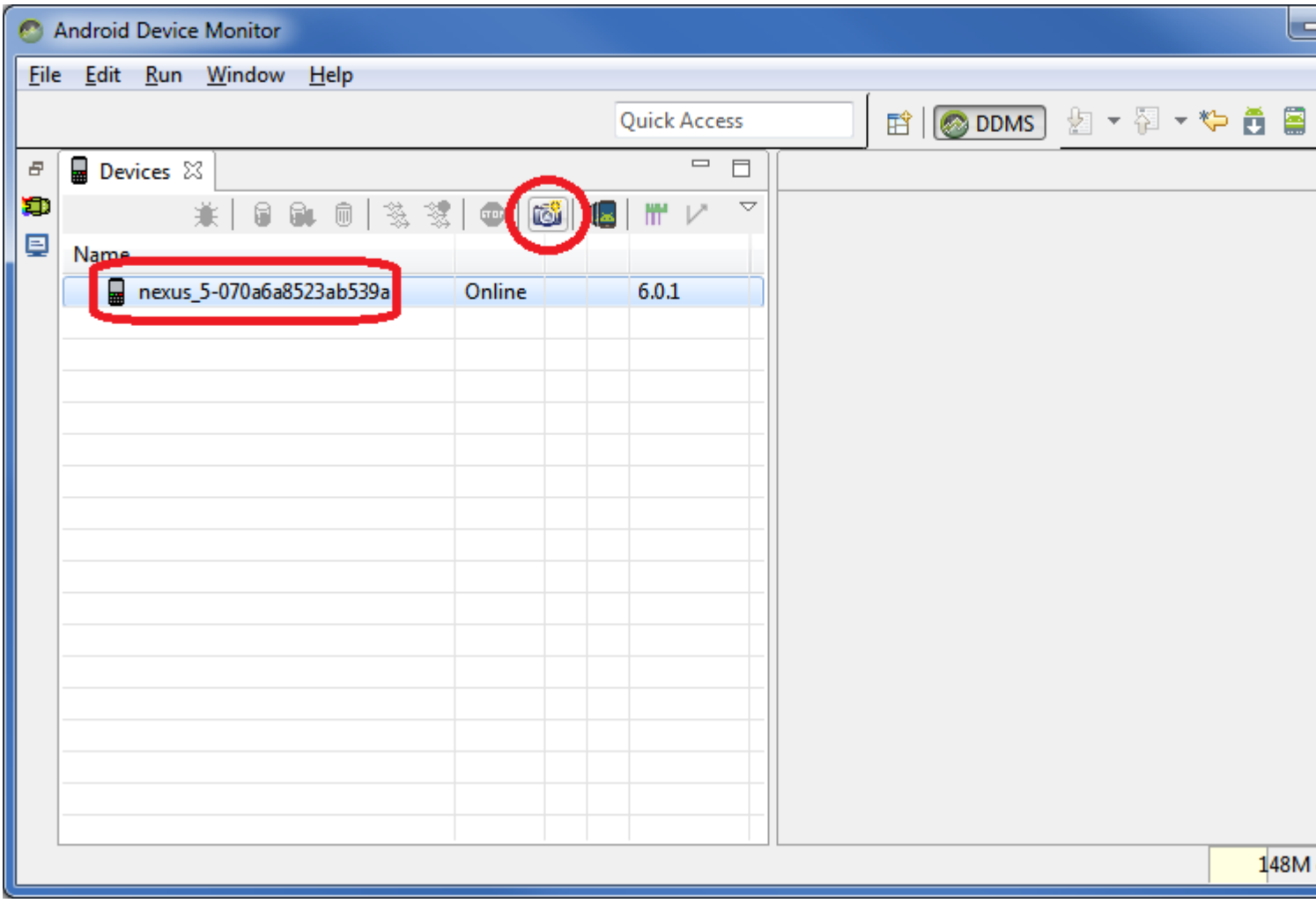
Android Studioでスクリーンショットをキャプチャする

1. Androidモニタタブをく
2. キャプチャボタンをクリックします。



Androidデバイスモニタによるスクリーンショットのキャプチャ

1. いているAndroidデバイスモニタ つまり、*C<ANDROID_SDK_LOCATION> \ tools \ monitor.bat*
2. あなたのデバイスをしてください
3. キャプチャボタンをクリックします。



ADBによるスクリーンショットのキャプチャ

のでは、デバイスのストレージにスクリーンショットをしています。

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

ADBでスクリーンショットをキャプチャし、PCに

LinuxまたはWindows with Cygwinをしているは、のコマンドをできます。

```
adb shell screencap -p | sed 's/\r$//' > screenshot.png
```

のビューのスクリーンショットをる

のView vスクリーンショットをりたいは、のコードをできます

```
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();
}
```

オンラインでスクリーンショットのキャプチャをむ https://riptutorial.com/ja/android/topic/4506/スクリーンショットのキャプチャ

172: ストリングのフォーマット

Examples

リソースをフォーマットする

リソースにワイルドカードをし、にそれらをみむことができます

1. strings.xmlを

```
<string name="my_string">This is %1$s</string>
```

2. にしてフォーマット

```
String fun = "fun";
context.getString(R.string.my_string, fun);
```

タイムスタンプをにフォーマットする

パターンのについては、「SimpleDateFormatリファレンス」をしてください。

```
Date now = new Date();
long timestamp = now.getTime();
SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy", Locale.US);
String dateStr = sdf.format(timestamp);
```

データのへの

データとの

int、float、double、long、booleanなどのデータは、String.valueOfをしてにフォーマットできます。

```
String.valueOf(1); //Output -> "1"
String.valueOf(1.0); //Output -> "1.0"
String.valueOf(1.2345); //Output -> "1.2345"
String.valueOf(true); //Output -> "true"
```

これとはに、をのデータにフォーマットする

```
Integer.parseInt("1"); //Output -> 1
Float.parseFloat("1.2"); //Output -> 1.2
Boolean.parseBoolean("true"); //Output -> true
```

オンラインでストリングのフォーマットをむ https://riptutorial.com/ja/android/topic/1346/ストリングのフォーマット

173: スナックバー

- Snackbar makeビュー、CharSequenceテキスト、int
- スナックバーメイクビュー、int resId、int duration

パラメーター

パラメータ	
る	ビューをつけるためのビュー。
テキスト	CharSequenceするテキスト。フォーマットされたテキストにすることができます。
resId	intするリソースのリソースID。フォーマットされたテキストにすることができます。
	intメッセージをする。これは、LENGTH_SHORT、LENGTH_LONGまたはLENGTH_INDEFINITEになります。

Snackbarはにするなフィードバックをします。モバイルののに、きなデバイスのになメッセージがされます。スナックバーはののすべてののにされ、にできるのは1つだけです。

タイムアウトやののでのユーザーのやりとり、にしいサーフェスやアクティビティをびすのにえます。 Snackbarはからスワイプできます。

SnackBarをするに、 build.gradleファイルにデザインサポートライブラリのをするがあります。

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

https://developer.android.com/reference/android/support/design/widget/Snackbar.html

Examples

シンプルなSnackbarの

Snackbarをするには、のようにします。

```
Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG).show();
```

このviewは、 SnackbarにするのにしたをつけるためにされSnackbar。 、これはXMLでした CoordinatorLayoutで、スワイプなどのをしたり、のウィジェット FloatingActionButton をにしたりすることができます。 CoordinatorLayoutがないは、ウィンドウのコンテンツビューがされます。

にに、 Snackbarアクションをし Snackbar。 なは、「りし」アクションです。

```
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();
```

Snackbarをしてでることができます

```
Snackbar snackbar = Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG);
snackbar.show();
```

Snackbarのテキストのをしたい

```
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();
```

デフォルトでは、Snackbarはのスワイプをします。 Snackbarでは、スワイプがにスワイプされたときにsnackBarをするをしています。

カスタムスナックバー

スナックバーをカスタマイズする

```
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;
```

```
    }
```

フラグメントまたはアクティビティからをびす

```
 SnackBar.makeText(MyActivity.this, "Please Locate your address at Map",
Snackbar.LENGTH_SHORT).show();
```

コールバックきスナックバー

Snackbar.Callbackをして、スナップバーがユーザーまたはタイムアウトによってされたかどうかをできます。

```
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();
```

カスタムスナックバー

このでは、カスタムUndoアイコンきのいSnackbarをしています。

```
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();
```

スナックバーとトーストどちらをうべきですか

トーストはに、したまたはこらなかったらかのアクションにしてユーザーにをするにされ、このアクションではユーザーがのアクションをとるはありません。メッセージがされたときのように、たとえば

```
Toast.makeText(this, "Message Sent!", Toast.LENGTH_SHORT).show();
```

スナックバーはのにもされます。しかしは、ユーザーにをこすをえることができます。たとえば、ユーザーがってをした、そのをにしたいとします。スナックバーに「りし」アクションをすることができます。このような

```
Snackbar.make(getCurrentFocus(), "Picture Deleted", Snackbar.LENGTH_SHORT)
        .setAction("Undo", new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Return his picture
            }
        })
        .show();
```

トーストはユーザーとのやりとりがないときにされます。スナックバーは、ユーザーがのアクションをるか、のアクションをにすことができるようにするためにされます。

カスタムスナックバーなビュー

SnackbarへのビューをすなしにSnackbarをすると、すべてのレイアウトはandroid.R.id.contentのandroidでされます。

```
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();
    }
}
```

コールクラス

しいCustomSnackBarfindViewByIdCustomSnackBar.VIEW_PARENT、 "message"、CustomSnackBar.STATE_ERROR;

オンラインでスナックバーをむ https://riptutorial.com/ja/android/topic/1500/スナックバー

174: スピナー

Examples

あなたのにスピナーをする

/res/values/strings.xml

```
<string-array name="spinner_options">
    <item>Option 1</item>
    <item>Option 2</item>
    <item>Option 3</item>
</string-array>
```

レイアウトXMLでは

```
<Spinner
    android:id="@+id/spinnerName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_options" />
```

アクティビティ

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

なスピナーの

スピナードロップダウンのです。のレイアウト

```
  <Spinner
    android:id="@+id/spinner"     <!-- id to refer this spinner from JAVA-->
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

</Spinner>
```

に`spinner`をする`spinner`をするはに2りあります。

1. XMLから、 **res**のの**values**ディレクトリに**array.xml**をします 。 この`arrayarray`

```
  <string-array name="defaultValue">
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
</string-array>
```

sppiner XMLにこのをする

```
            android:entries="@array/defaultValue"
```

2. JAVAをしてをすることもできます

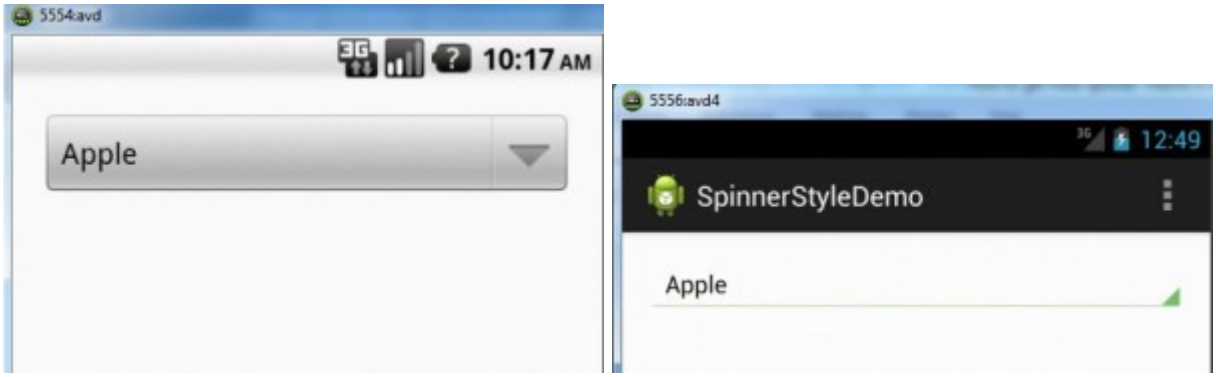
`activity` cityArea =SpinnerでしているはfindViewByIdR.id.cityArea;そうでないは、 `fragment`している

```
cityArea = (Spinner) findViewById(R.id.cityArea);
```

`arrayList`の`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));
```

これはのようになります



デバイスのAndroidのバージョンにじてスタイルをレンダリングします

はデフォルトテーマのいくつかです

アプリがマニフェストでテーマをにしない、Androidシステムはアプリのの`targetSdkVersion`についてデフォルトテーマをし、アプリののをします。

Android SDKのバージョン	デフォルトのテーマ
バージョン<11	@androidスタイル/テーマ
バージョン1113	@androidstyle / Theme.Holo

Android SDKのバージョン	デフォルトのテーマ
14	@androidstyle / Theme.DeviceDefault

`Spinner`はXMLのけをりてにカスタマイズすることができます。

```
 android:background="@drawable/spinner_background"

 android:layout_margin="16dp"

 android:padding="16dp"
```

XMLでカスタムのをし、それをします。

スピナーでしたのやそののをにできます

```
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) {

        }
    });
```

スピナーでしたのテキストのをする

これはXMLで2つのでうことができます

```
<item android:state_activated="true" android:color="@color/red"/>
```

ポップアップでしたのがわります。

JAVAからこれをうsetOnItemSelectedListener...

```
 @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long
id) {
                 ((TextView) parent.getChildAt(0)).setTextColor(0x00000000);
// similarly change `background color` etc.
            }
```

オンラインでスピナーをむ https://riptutorial.com/ja/android/topic/3459/スピナー

175: スプラッシュの

のスプラッシュは、それをするもなではありません。このように、なスプラッシュスクリーンです。

Examples

なスプラッシュ

スプラッシュは、のアクティビティとですが、バックグラウンドでのスタートアップのニーズをすべてできます。

マニフェスト

```
<?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>
```

、たちのスプラッシュがのアクティビティとしてびされます。

いくつかのなappもするスプラッシュスクリーンのをにします。

```
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);

    }

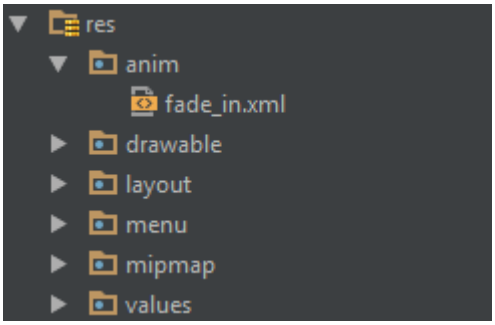
    public void onPause(){
        super.onPause();
        finish();
    }

}
```

このでは、Androidスタジオをしてできるシンプルでなアニメーションのスプラッシュをします。

ステップ1アニメーションをする

*res*ディレクトリに*anim*というのしいディレクトリをします。クリックして、 *fade_in.xml*というのしいアニメーションリソースファイルをします。



に、のコードを*fade_in.xml*ファイルにします。

```
<?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>
```

ステップ2アクティビティをする

`Splash`というのAndroid Studioをしてのアクティビティをします。に、のコードをします。

```
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);
    }
}
```

に、のコードをレイアウトファイルにします。

```
<?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>
```

ステップ3デフォルトランチャーをする

*AndroidManifest*ファイルにのコードをして、 `Splash`アクティビティをランチャーにりえます

```
<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>
```

に、 *AndroidManifest*ファイルからのコードをして、デフォルトのランチャーアクティビティをします。

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />

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

オンラインでスプラッシュのをむ https://riptutorial.com/ja/android/topic/9316/スプラッシュの

176: スマートカード

Examples

スマートカードの

については、をめるためのスニペットがあります

```
//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);
}
```

Javaでは、Androidでできないパッケージjavax.smarcardをしてがわれるので、ここでAPDUスマートカードコマンドのやについてえてください。

、のえでわれたように

バルクアウトエンドポイントにAPDUスマートカードコマンドをするだけでは、バルクエンドポイントにしてAPDUをすることはできません。エンドポイントをするには、のコードスニペットをしてください。

```
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;
                }
            }
```

```
        }
    }
```

これで、APDUコマンドとAPDUブロックをするバルクインとバルクアウトのエンドポイントがされました。

コマンドのについては、のコードスニペットをしてください。

```
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));
    }
}
```

/のについては、のコードスニペットをしてください。

```
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;
}
```

ここでこのえがえる、にされるコマンドはのとおりです。

PC_to_RDR_IccPowerOnカードをアクティブにするコマンド。

ここではUSBデバイスクラスのセクション6.1.1をんですることができます。

に、このコマンドのをここにします62000000000000000000これをするはのとおりです。

```
write(connection, epOut, "62000000000000000000");
```

APDUコマンドをにした、のコマンドをしてをみることができます。

```
read(connection, epIn);
```

のようなものをける

```
80 18000000 00 00 00 00 00 3BBF11008131FE4545504100000000000000000000000F1
```

ここでコードでけったレスポンスは、コードからの`read()`メソッドの`result`に`read()`ます

オンラインでスマートカードをむ https://riptutorial.com/ja/android/topic/10945/スマートカード

177: スワイプしてする

1. **setColorSchemeResources**は、SwipeToRefreshLayout インジケータのをします。
2. **setOnRefreshListener**は、レイアウトをスワイプしたときのをします。
3. **applayout_behavior = "@/ appbar_scrolling_view_behavior"**レイアウトにツールバーがある、ツールバーのscrollFlagsにこれをすると、スクロールダウンにツールバーがにスライドし、スクロールアップにびスライドします。

Examples

RecyclerViewでスワイプしてする

RecyclerViewで スワイプ·トゥ·リフレッシュ·レイアウトをするには、アクティビティー/フラグメント·レイアウト·ファイルにをします。

```
<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>
```

あなたのActivity / Fragmentにをして**SwipeToRefreshLayout**をします

```
    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
        }
    });
```

スワイプ·ツー·リフレッシュをあなたのアプリにえる

のでアプリケーションの`build.gradle`ファイルにのがされていることをしてください。

```
compile 'com.android.support:support-core-ui:24.2.0'
```

に、レイアウトに`SwipeRefreshLayout`をします。

```
<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>
```

に、 `SwipeRefreshLayout.OnRefreshListener`リスナーをします。

```
mSwipeRefreshLayout = (SwipeRefreshLayout) findViewById(R.id.swipe_refresh_layout);
mSwipeRefreshLayout.setOnRefreshListener(new OnRefreshListener() {
    @Override
    public void onRefresh() {
         // your code
    }
});
```

オンラインでスワイプしてするをむ https://riptutorial.com/ja/android/topic/5241/スワイプしてする

178: セキュアなSharedPreferences

き

プリファレンスは、 キーベースの**XML**ファイルです 。
`/data/data/package_name/shared_prefs/<filename.xml>`ます。

rootをつユーザーはこのにし、そのをすることができます。のをしたいは、なとメカニズムをできます。

はしてのためにされたものではなく、データをするためのなです。

1. public static String encryptストリング;
2. public static String decryptストリング;

パラメーター

パラメータ	
	またはする。

はにされたことはなく、データをするためのなです。

ユーザーのなどのなをするためのプリファレンスをすることはおめできません。ユーザーのパスワードなどをするには、Androidの`AccountManager`などののをするがあります。

Examples

をする

シンプルコーデック

ここでは、をするために、なとをのようにできます。

```
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));
}
```

```
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);
```

オンラインでセキュアなSharedPreferencesをむ https://riptutorial.com/ja/android/topic/9887/セキュアなsharedpreferences

179: セキュアなSharedPreferences

き

プリファレンスは、 キーベースの**XML**ファイルです 。 / data / data / package_name / shared_prefs / <filename.xml>のにあります。

rootをつユーザーはこのにし、そのをすることができます。のをしたいは、なとメカニズムをできます。

はしてのためにされたものではなく、データをするためのなです。

1. public static String encryptストリング;
2. public static String decryptストリング;

パラメーター

パラメータ	
	またはする。

はにされたことはなく、データをするためのなです。

ユーザーのなどのなをするためのプリファレンスをすることはおめできません。ユーザーのパスワードなどをするには、Androidの`AccountManager`などののをするがあります。

Examples

をする

シンプルコーデック

ここでは、をするために、なとをのようにできます。

```
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));
}
```

```
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);
```

オンラインでセキュアなSharedPreferencesをむ https://riptutorial.com/ja/android/topic/9890/セキュアなsharedpreferences

180: セキュリティ

Examples

アプリケーションのをする - ざん

このテクニックでは、.apkがでされていることをするをしくし、がをち、ユーザーだけがアクセスできることをしています。このを3つのなステップにけることができます。

- のをします。
- あなたのアプリケーションのにあなたのをめみます。
- にがめまれたのとすることをします。

コードスニペットはのとおりです。

```
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;

}
```

オンラインでセキュリティをむ https://riptutorial.com/ja/android/topic/4664/セキュリティ

181: ダイアログ

パラメーター

ライン	
show;	ダイアログをします。
setContentView R.layout.yourlayout;	ダイアログのContentViewをカスタムレイアウトにします。
dismiss	ダイアログをじる

- のDialogのダイアログは、コンストラクタでされるときに`show()`をびすはありません
- Alertダイアログは、 `AlertDialog.Builder()`クラスのしいインスタンスをじてするがあります。 Builderパターンにいて、AlertDialog.Builderのすべてのメンバーをメソッドさせて、ダイアログインスタンスをすることができます。
- Alert Dialog Builderはダイアログを`show()`でき`create()` AlertDialogインスタンスで`create()`、`show()`をびすはありません`show()`

Examples

ダイアログ

```
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();
                            }
```

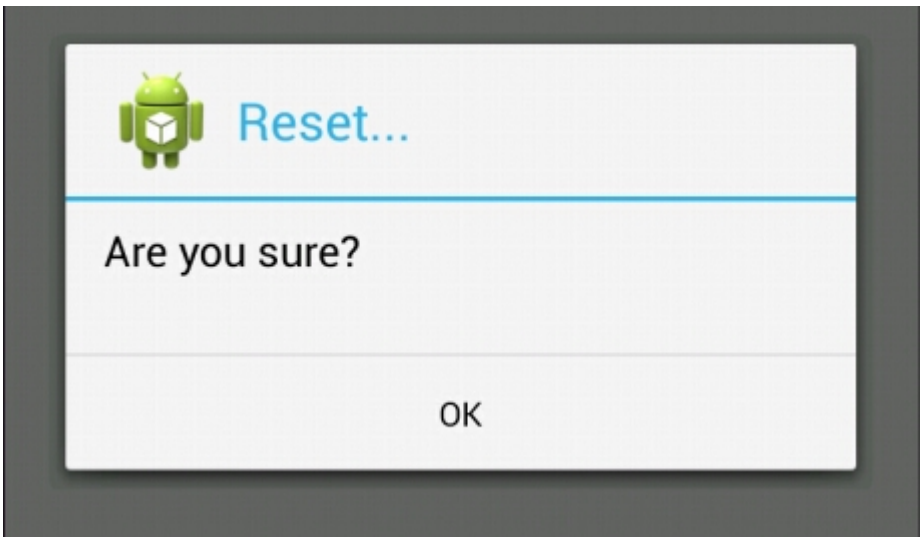
```
                        });

        AlertDialog alertDialog = alertDialogBuilder.create();
        alertDialog.show();
```

なダイアログ

```
    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.
```

、このコードはこれをします



ソース*WikiHow*

DialogFragmentのピッカー

ダィアログの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">
```

```
    <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>
```

ダィァログクラス

```
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;
    }

}
```

ダイアログをびすアクティビティ

```
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`は、アプリケーションのどこにでもダイアログをできるため、 `DatePicker`をするもなです。 `DatePicker`ウィジェットでのレイアウトをするはありません。

ダィアログをする

```
DatePickerDialog datePickerDialog = new DatePickerDialog(context, listener, year, month, day);
datePickerDialog.show();
```

のダィアログから`DataPicker`ウィジェットをして、よりくのにアクセスしたり、のをミリでしたりすることができます。

```
DatePicker datePicker = datePickerDialog.getDatePicker();
datePicker.setMinDate(System.currentTimeMillis());
```

DatePicker

`DatePicker`をすると、をできます。 `DatePicker`しいインスタンスをするときに、をできます。をしないと、のがデフォルトでされます。

たちはることができます`DatePicker`して、ユーザーに`DatePickerDialog`か、とたちのレイアウトをすることで、 `DatePicker`ウィジェット。

また、ユーザーができるのをすることもできます。

をミリでする

```
//In this case user can pick date only from future
datePicker.setMinDate(System.currentTimeMillis());
```

のをミリでする

```
//In this case user can pick date only, before following week.
datePicker.setMaxDate(System.currentTimeMillis() + TimeUnit.DAYS.toMillis(7));
```

ユーザーがしたにするをけるには、 `Listener`をするがあります。

`DatePickerDialog`をしている、 `DatePickerDialog`しいインスタンスをするときに、コンストラクタで`OnDateSetListener`をできます。

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) {

    }
}
```

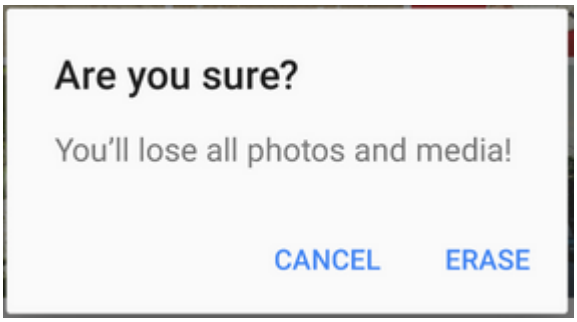
それの、 `DatePicker`ウィジェットをしてのレイアウトをするは、 のにすようにのリスナーをするがあります

Appcompatをして Material Design AlertDialogをアプリケーションにする

`AlertDialog`は`Dialog`サブクラスで、1つ、2つ、または3つのボタンをできます。このダイアログボックスでをするは、 `setMessage()`メソッドをします。

`android.app`パッケージの`AlertDialog`は、なるAndroid OSバージョンでなってされます。

Android V7 Appcompatライブラリには、にすように、サポートされているすべてのAndroid OSバージョンでMaterial Designでされる`AlertDialog`が`AlertDialog`されています。



まず、V7 Appcompatライブラリをプロジェクトにするがあります。あなたはアプリレベルのbuild.gradleファイルでこれをうことができます

```
dependencies {
    compile 'com.android.support:appcompat-v7:24.2.1'
    //........
}
```

しいクラスをインポートするようにしてください。

```
import android.support.v7.app.AlertDialog;
```

に、AlertDialogをのようにします。

```
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();
```

AlertDialogのListView

`ListView`または`RecyclerView`をにリストからすることができますが、がなく、のからするは、`AlertDialog.Builder setAdapter`をできます。

```
    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();
    }
```

おそらく、の`ListView`がないは、なをできます。

```
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();
```

EditTextをしたカスタムダイアログ

```
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();
    }
```

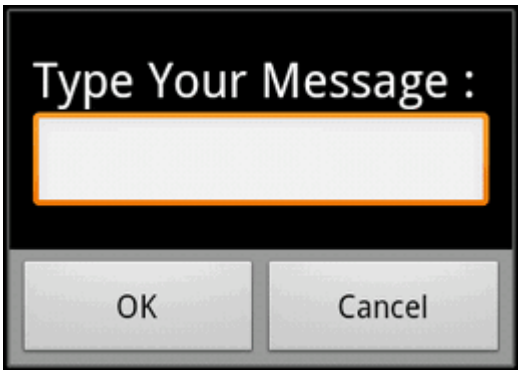
XMLファイルres / layout / alert_dialog.xml

```
<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>
```



とタイトルなしのフルスクリーンカスタムダイアログ

`styles.xml`にカスタムスタイルを

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppBaseTheme" parent="@android:style/Theme.Light.NoTitleBar.Fullscreen">
    </style>
</resources>
```

ダイアログのカスタムレイアウトをします`fullscreen.xml`

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

</RelativeLayout>
```

その、Javaファイルでは、それをアクティビティやダイアログなどにできます。

```
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();
    }
}
```

タイトルのダイアログ

AlertDialog.BuilderのsetCustomTitleメソッドをすると、ダイアログタイトルにされるのビューをできます。このメソッドのないの1つは、タイトルがいアラートダイアログをすることです。

```
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!"/>

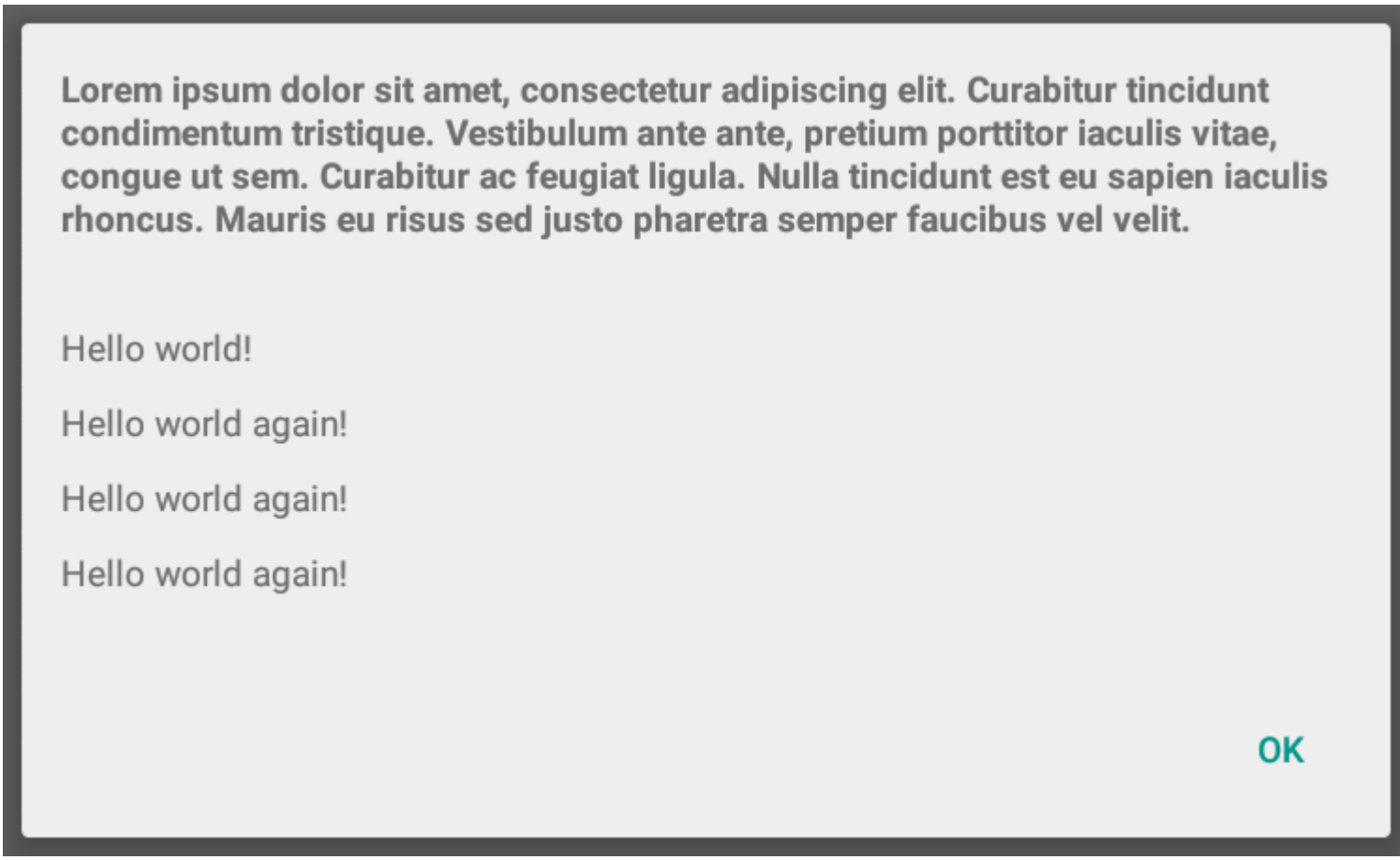
        <TextView
```

```
            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>
```



オンラインでダイアログをむ https://riptutorial.com/ja/android/topic/1225/ダイアログ

182: タイムユース

Examples

をミリに

dd / MM / yyyyのをミリにするには、このをStringのデータでびします。

```
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();
}
```

このメソッドは、ミリをTime-stampにします。Format date

```
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;
}
```

このメソッドは、されたの、、およびをミリにします。 `Timpicker`または`Datepicker`をするとにちます

```
public static long getTimeInMillis(int day, int month, int year) {
    Calendar calendar = Calendar.getInstance();
    calendar.set(year, month, day);
    return calendar.getTimeInMillis();
}
```

からミリをします

```
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;
}
```

のをします。

```
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;
}
```

Javaフォーマットのサポートをします。 パターン

あるにチェックする

このは、されたがあるであるかどうかをするのにちます。

のをするには、 ***DateUtils***クラスをできます

```
boolean isToday = DateUtils.isToday(timeInMillis);
```

が1であることをするには、

```
private static boolean isWithinWeek(final long millis) {
    return System.currentTimeMillis() - millis <= (DateUtils.WEEK_IN_MILLIS -
DateUtils.DAY_IN_MILLIS);
}
```

が1であることをするには、

```
private static boolean isWithinYear(final long millis) {
    return System.currentTimeMillis() - millis <= DateUtils.YEAR_IN_MILLIS;
}
```

をむのうちにをするには、

```
public static boolean isWithinDay(long timeInMillis, int day) {
    long diff = System.currentTimeMillis() - timeInMillis;

    float dayCount = (float) (diff / DateUtils.DAY_IN_MILLIS);

    return dayCount < day;
}
```

DateUtilsは**android.text.format.DateUtils**です。

GetCurrentRealTime

これはのデバイスをし、とデバイスのを/します

```
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;
}
```

UTCベースのタイムゾーンをします。

```
public static TimeZone getUTCTimeZone() {
    return TimeZone.getTimeZone("GMT");
}
```

オンラインでタイムユースをむ https://riptutorial.com/ja/android/topic/7138/タイムユース

183: ダガー2

- @Module
- @Component= {OtherComponent.class}、modules = {ModuleA.class、ModuleB.class}
- DaggerMyComponent.create
- DaggerMyComponent.builder。myModulenewMyModule。create

でダガーとしないでください。ダガー2のです。

Examples

アプリケーションおよびアクティビティのためのコンポーネント

アプリケーションのシングルトンオブジェクトをするの`AppModule`にするな`AppComponent`。

```
@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {

    void inject(App app);

    Context provideContext();

    Gson provideGson();
}
```

アプリケーションでするための`Gson`インスタンスなど、そのシングルトンオブジェクトをする`AppComponent`とにするモジュール。

```
@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;
    }
}
```

ダガーとシングルトンコンポーネントをセットアップするためのサブクラスされたアプリケーション。

```
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;
    }
}
```

、 `AppComponent`オブジェクトへのアクセスをるために`AppComponent`にするアクティビティスコープきコンポーネント。

```
@ActivityScope
@Component(dependencies = AppComponent.class, modules = ActivityModule.class)
public interface MainActivityComponent {

    void inject(MainActivity activity);
}
```

また、 `FragmentManager`よううななをするな`ActivityModule`

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

すべてをまとめると、たちは`Gson`ていて、たちのをすることができ、アプリにじ`Gson`をするようにすることができます

```
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);
    }
}
```

カスタムスコープ

```
@Scope
@Documented
@Retention(RUNTIME)
public @interface ActivityScope {
}
```

スコープはなるであり、にじてのものをできます。

コンストラクタインジェクション

のないクラスは、でにできます。

```
public class Engine {

    @Inject // <-- Annotate your constructor.
    public Engine() {
    }
}
```

このクラスは、 のコンポーネントによってされます。 はなく、もありません 。それのコードはありません。

は、コンストラクタのパラメータとしてされます。 Daggerは、をできるり、コンストラクタをびしてをします。

```
public class Car {

    private Engine engine;

    @Inject
    public Car(Engine engine) {
        this.engine = engine;
    }
```

```
}
```

このクラスは、このコンポ―ネントがすべてのこのは`Engine`できる 、すべてのコンポ―ネントによってされることができます。 `Engine`はコンストラクタにもできるので、 どのコンポ―ネントでも`Car`をできます。

すべてのをコンポ―ネントができるときはいつでも、コンストラクタインジェクションをできます。コンポ―ネントはをできます。

- コンストラクタインジェクションをしてできます
- コンポ―ネントのモジュ―ルがそれをすることができます
- コンポ―ネント `@Subcomponent`によってすることができます。
- するコンポ―ネントによってされるオブジェクトをすることができますコンポ―ネントの

@Componentのわりに@Subcomponentをする= {...}

```
@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);
    }
}
```

build.gradleでDagger 2をする

Gradle 2.2がリリ―スされて、android-aptプラグインのはされました。 Dagger 2をするには、のをするがあります。いバ―ジョンのGradleのは、ののをします。

Gradle> = 2.2の

```
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'
}
```

Gradle <2.2の

Dagger 2をするには、 `android-apt`プラグインをするがあります。これをルートbuild.gradleにしてください

```
buildscript {
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.0'
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}
```

に、アプリケーション・モジュールのbuild.gradleにはがまれているがあります。

```
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}"
}
```

リファレンス https : //github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

のモジュールからコンポーネントをする

Dagger 2では、のモジュールからコンポーネントをすることができます。このようにコンポーネントをすることができます。

```
@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);
}
```

`GeneralPurposeModule`と`SpecificModule`という2つのモジュールは、のようにできます。

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;
    }
}
```

フェーズでは、コンポーネントはにじてのモジュールからオブジェクトをします。

このアプローチは、 モジュールのでにです。このでは、 `Retrofit`オブジェクトネットワークのにや`PropertiesReader` ファイルのなどのコンポーネントをインスタンスするモジュールがあります。そののアプリケーションコンポーネントにして、のコントローラとサービスクラスのインスタンスをするのモジュールもあります。

オンラインでダガー2をむ https://riptutorial.com/ja/android/topic/3088/ダガー2

184: タッチイベント

Examples

ビュー—とビュー—グルー—プのタッチイベントをどのようにするか

1. ネストされたビューグルー—プの`onTouchEvents()`は、 `boolean` onInterceptTouchEventによってできます。

`OnInterceptTouchEvent`のデフォルトはfalseです。

の`onTouchEvent`は、のにされます。 `OnInterceptTouchEvent`がfalseをしたは、チェーンのにあるモーションイベントをの`OnTouchEvent`ハンドラにします。 trueをすと、はtouchイベントをします。

しかし、いくつかのに`OnTouchEvent`させ、あるものをビュー—またはおそらくのがするようにするもあります。

これはのでることができます。

2. の`OnInterceptTouchEvent`からをするの1つは、 requestDisallowInterceptTouchEventをすることです 。

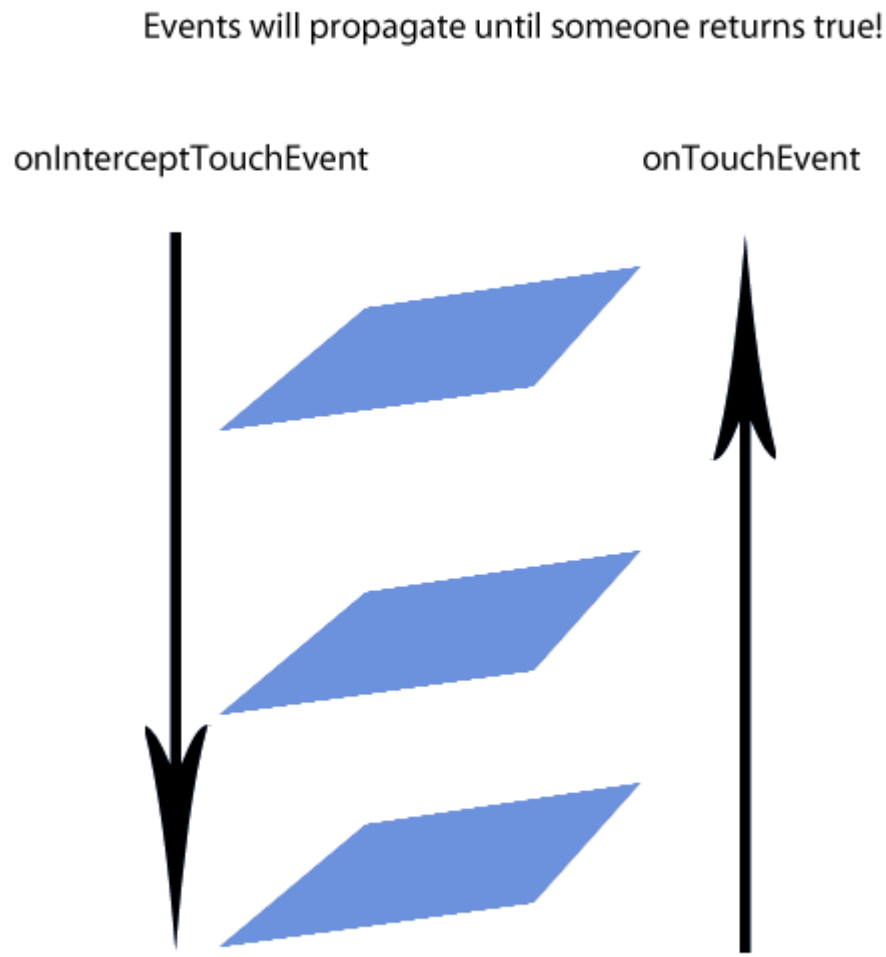
   public void requestDisallowInterceptTouchEventboolean disallowIntercept

これにより、でイベントハンドラがになっている、こののの`OnTouchEvent`をビュー—でできなくなります。

3.

`OnInterceptTouchEvent`がfalseの、の`OnTouchEvent`がされます。さまざまなタッチィベントをするにメソッドがある、になっているするイベントハンドラはOnTouchEventをにします。

このえ
タッチイベントのがどのようにするかの
`parent -> child|parent -> child|parent -> child views.`



ここからの

4. のは、の`OnInterceptTouchEvent`からさまざまなをすことです。

このでは、ViewGroupのタッチイベントのから`OnTouchEvent`し、ユーザーがスクロールしているときにの`OnTouchEvent`をインターセプトするをします。

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;
}
```

これはじリンクのいくつかのコーードで、のりののパラメーータをするをしています

4b。

```
// The hit rectangle for the ImageButton
myButton.getHitRect(delegateArea);

// Extend the touch area of the ImageButton beyond its bounds
// on the right and bottom.
delegateArea.right += 100;
delegateArea.bottom += 100;

// Instantiate a TouchDelegate.
// "delegateArea" is the bounds in local coordinates of
// the containing view to be mapped to the delegate view.
// "myButton" is the child view that should receive motion
// events.
TouchDelegate touchDelegate = new TouchDelegate(delegateArea, myButton);

// Sets the TouchDelegate on the parent view, such that touches
// within the touch delegate bounds are routed to the child.
if (View.class.isInstance(myButton.getParent())) {
    ((View) myButton.getParent()).setTouchDelegate(touchDelegate);
```

```
}
```

オンラインでタッチイベントをむ https://riptutorial.com/ja/android/topic/7167/タッチイベント

185: タッチイベントとモーションイベントのき

Android APIのなタッチ/モーションシステムの。

パラメーター

リスナー	
onTouchListener	ボタン、サーフェスなどのシングルタッチをします。
onTouchEvent	サーフェスでできるリスナSurfaceViewなどのリスナーのようにするはありませんonTouchListener
onLongTouch	onTouchにていますが、ボタン、サーフェスなどのしをちけます。

Examples

ボタン

`Button`するタッチイベントは、のようにできます。

```
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;
    }
}
```

サーフェスのタッチイベントハンドラSurfaceView、GLSurfaceViewなど

```
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;
    }
}
```

あるいはに

```
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;
    }
}
```

のマルチタッチの

```
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.
    }
}
```

オンラインでタッチイベントとモーションイベントのをむ
https://riptutorial.com/ja/android/topic/9315/タッチイベントとモーションイベントの

186: ツール

Androidには、ツールがXMLファイルにをできるようにするのXMLがあります。

URIはのとおりです。

`http://schemas.android.com/tools`であり、は`tools:`にバインドされてい`tools:`。

Examples

デザインレイアウト

これらのは、Android Studioでレイアウトをレンダリングするときにされますが、ランタイムにはしません。

に、レイアウトプレビューには`android:`ネームスペースではなくネームスペースという`tools:`をするだけで、どのAndroidフレームワークアトリビュートもできます。にされる`android:` namespaceと、する`tools:`レイアウトプレビューのをオーバーライドするのをできます。

「」のでされているように、ツールのをするだけです。

たとえば、のような`text`があります。

```
<EditText
      tools:text="My Text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
```

または、 `visibility`をしてプレビューをする

```
<LinearLayout
        android:id="@+id/ll1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:visibility="gone" />
```

レイアウトをアクティビティやフラグメントにける`context`

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity" >
```

または、のレイアウトでレイアウトプレビューをおよび`showIn`するための`showIn`

```
<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" />
```

オンラインでツールをむ https://riptutorial.com/ja/android/topic/1676/ツール

187: データのをにうSyncAdapter

き

あなたのアプリケーションのアダプタコンポーネントは、デバイスとサーバでデータをするタスクのコードをカプセルします。アダプタフレームワークは、アプリケーションでするスケジューリングとトリガについて、アダプタコンポーネントでコードをします。

はSyncAdapterにりみました。はとをしたい、それはをけるかもしれません。

Examples

サーバーからするすべてのをつアダプター。

```
<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>
```

このコードはマニフェストファイルにするがあります

のコードでは、syncserviceとconteproviderとauthenticatorserviceがあります。

アプリでは、syncadpterとauthenticator xmlファイルをするためにxmlパッケージをするがあります。 **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"/>
```

オーセンティケータ

```
import android.accounts.AbstractAccountAuthenticator;
import android.accounts.Account;
import android.accounts.AccountAuthenticatorResponse;
import android.accounts.NetworkErrorException;
import android.content.Context;
import android.os.Bundle;

public class Authenticator extends AbstractAccountAuthenticator {
    private Context mContext;
    public Authenticator(Context context) {
        super(context);
         this.mContext=context;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse accountAuthenticatorResponse,
String s) {
        return null;
    }

    @Override
    public Bundle addAccount(AccountAuthenticatorResponse accountAuthenticatorResponse, String
s, String s1, String[] strings, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle confirmCredentials(AccountAuthenticatorResponse
accountAuthenticatorResponse, Account account, Bundle bundle) throws NetworkErrorException {
        return null;
```

```
    }

    @Override
    public Bundle getAuthToken(AccountAuthenticatorResponse accountAuthenticatorResponse,
Account account, String s, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public String getAuthTokenLabel(String s) {
        return null;
    }

    @Override
    public Bundle updateCredentials(AccountAuthenticatorResponse accountAuthenticatorResponse,
Account account, String s, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle hasFeatures(AccountAuthenticatorResponse accountAuthenticatorResponse,
Account account, String[] strings) throws NetworkErrorException {
        return null;
    }
}
```

AuthenticatorService

```
public class AuthenticatorService extends Service {

    private Authenticator authenticator;

    public AuthenticatorService() {
        super();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        IBinder ret = null;
        if (intent.getAction().equals(AccountManager.ACTION_AUTHENTICATOR_INTENT)) ;
        ret = getAuthenticator().getIBinder();
        return ret;
    }

    public Authenticator getAuthenticator() {
        if (authenticator == null)
            authenticator = new Authenticator(this);
        return authenticator;
    }
}
```

IpDataDBHelper

```
 public class IpDataDBHelper extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION=1;
    private static final String DATABASE_NAME="ip.db";
    public static final String TABLE_IP_DATA="ip";
```

```
    public static final String COLUMN_ID="_id";
    public static final String COLUMN_IP="ip";
    public static final String COLUMN_COUNTRY_CODE="country_code";
    public static final String COLUMN_COUNTRY_NAME="country_name";
    public static final String COLUMN_CITY="city";
    public static final String COLUMN_LATITUDE="latitude";
    public static final String COLUMN_LONGITUDE="longitude";

    public IpDataDBHelper(Context context, String name, SQLiteDatabase.CursorFactory factory,
int version) {
        super(context, DATABASE_NAME, factory, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        String CREATE_TABLE="CREATE TABLE " + TABLE_IP_DATA + "( " + COLUMN_ID + " INTEGER
PRIMARY KEY ,"
                + COLUMN_IP + " INTEGER ," + COLUMN_COUNTRY_CODE + " INTEGER ," +
COLUMN_COUNTRY_NAME +
                " TEXT ," + COLUMN_CITY + " TEXT ," + COLUMN_LATITUDE + " INTEGER ," +
COLUMN_LONGITUDE + " INTEGER)";
        sqLiteDatabase.execSQL(CREATE_TABLE);
        Log.d("SQL",CREATE_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + TABLE_IP_DATA);
        onCreate(sqLiteDatabase);
    }

    public long AddIPData(ContentValues values)
    {
        SQLiteDatabase sqLiteDatabase =getWritableDatabase();
        long insertedRow=sqLiteDatabase.insert(TABLE_IP_DATA,null,values);
        return insertedRow;
    }

    public Cursor getAllIpData()
    {
        String[]
projection={COLUMN_ID,COLUMN_IP,COLUMN_COUNTRY_CODE,COLUMN_COUNTRY_NAME,COLUMN_CITY,COLUMN_LATITUDE,CO

        SQLiteDatabase sqLiteDatabase =getReadableDatabase();
        Cursor cursor =
sqLiteDatabase.query(TABLE_IP_DATA,projection,null,null,null,null,null);
        return cursor;
    }

    public int deleteAllIpData()
    {
        SQLiteDatabase sqLiteDatabase=getWritableDatabase();
        int rowDeleted=sqLiteDatabase.delete(TABLE_IP_DATA,null,null);
        return rowDeleted;
    }
}
```

な

```
     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();
}
```

}

オンラインでデータのをにうSyncAdapterをむ https://riptutorial.com/ja/android/topic/10774/データのをにうsyncadapter

188: データのI

き

このトピックでは、Androidでとがどのようにするかについてします。

Examples

なでパスワードをしてデータのAES

のでは、 AESをしてのデータブロックをします 。はなランダムソルト、1000ラウンドのSHA-256でられます。はCBCモードのAESをランダムIVでします。

EncryptedData salt 、 iv 、およびencryptedData クラスにされているデータは、1バイトのにできます。その、データをしたり、にしたりすることができます。

```
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;
}
```

のコードは、とをテストするをしています。

```
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();
}
```

オンラインでデータの/をむ https://riptutorial.com/ja/android/topic/3471/データの-

189: データバインディングライブラリ

セットアップ

データバインディングをするに、 `build.gradle`プラグインをにするがあります。

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

バージョン1.5.0の*Android Gradle*プラグインにデータバインディングがされました

バインディングクラス

データバインディングプラグインは、レイアウトのファイルをPascalのにし、に「Binding」をすることによって、バインディングクラスをします。したがって、 `item_detail_activity.xml`は、`ItemDetailActivityBinding`というのクラスをします。

リソース

-

Examples

なテキストフィールドバインド

Gradleモジュール**app**の

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

データ·モデル

```
public class Item {
    public String name;
    public String description;

    public Item(String name, String description) {
        this.name = name;
        this.description = description;
    }
}
```

レイアウト**XML**

のステップは、レイアウトを`<layout>`タグにラップし、 `<data>`をし、データモデルに`<variable>`をすることです。

に、 `@{model.fieldname}`をしてXMLをデータモデルのフィールドにバインドできます。ここで、`model`はので、 `fieldname`はアクセスするフィールドです。

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>
```

バインディングでしくされたXMLレイアウトファイルごとに、Android Gradleプラグインはするクラスbindingsをします。 *item_detail_activity*というのレイアウトがあるため、するされたバインディングクラスは`ItemDetailActivityBinding`とばれ`ItemDetailActivityBinding`。

このバインディングは、のようにActivityでできます。

```
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);
    }
}
```

アクセサメソッドによるバインド

モデルにプライベートメソッドがある、データバインディングライブラリではメソッドのフルネ

ームをせずにビューにアクセスすることができます。

データ・モデル

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

レイアウト **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>
```

クラスの

データ・モデル

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

レイアウト **XML**

Javaのとに、されたクラスをインポートするがあります。

```
<?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>
```

パッケージ`java.lang.*`は、システムによってにインポートされます。 `JVM` for `Java`じです

フラグメントのデータバインディング

データ·モデル

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name){
        this.name = name;
    }

}
```

レイアウト**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>
```

```
@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();
}
```

みみのデ―タバイ ンディ ング

データバイ ンディ ングは、のをサポ―トします。

	プロパテ ィ
AbsListView	android:selectedItemPosition
CalendarView	android:date
CompoundButton	android:checked
DatePicker	• android:year • android:month • android:day
EditText	android:text
NumberPicker	android:value
RadioGroup	android:checkedButton
RatingBar	android:rating
SeekBar	android:progress
TabHost	android:currentTab
TextView	android:text
TimePicker	• android:hour • android:minute
ToggleButton	android:checked
Switch	android:checked

```
<layout ...>
    <data>
```

```
        <variable type="com.example.myapp.User" name="user"/>
    </data>
    <RelativeLayout ...>
        <EditText android:text="@={user.firstName}" .../>
    </RelativeLayout>
</layout>
```

バインディング`@={}`は、バインディングになの=があることにしてください。バインディングでメソッドをすることはできません。

RecyclerViewアダプタでのデータバインディング

また、`RecyclerView`アダプタでデータバインディングをすることもできます。

データ・モデル

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

XMLレイアウト

```
<TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@{item.name}"/>
```

アダプタクラス

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

Bindingでリスナーをクリック

clickHandlerのインタフェースをする

```
public interface ClickHandler {
    public void onButtonClick(View v);
}
```

レイアウト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>
```

アクティビティのイベントをする

```
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();
    }
}
```

ラムダをしたカスタムイベント

インタフェースの

```
public interface ClickHandler {
    public void onButtonClick(User user);
}
```

Modelクラスをする

```
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;
    }
}
```

レイアウト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>
```

アクティビティコード

```
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();
    }
}
```

XMLコードではできないがJavaコードでできるビューリスナーの、カスタムバインドでバインドできます。

カスタムクラス

```
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);
    }
}
```

ハンドラクラス

```
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>
```

データバインディングのデフォルト

プレビューペインには、データバインディングのデフォルトがされます。

えば

```
android:layout_height="@{@dimen/main_layout_height, default=wrap_content}"
```

これはに`wrap_content`を`wrap_content` 、プレビューウィンドウで`wrap_content`としてします。

もうつのは

```
android:text="@{user.name, default=`Preview Text`}"
```

プレビューウィンドウに`Preview Text`がされますが、デバイス/エミュレータですると、にバイン

ドされたテキストがされます

カスタムによるデータバインディングint、boolean

には、のにづいた/のようななをするがあります。そののにしてモデルをできないか、モデルをするのはいではありません。 DataBindingは、これらのオペレーションをするためのデータをサポートします。

```
<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>
```

javaクラスからそのをします。

```
binding.setSelected(true);
```

ダイアログでのデータバインディング

```
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();
}
```

BindingAdapterのとしてウィジェットをす

レイアウト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>
```

BindingAdapterメソッド

```
@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);
}
```

オンラインでデータバインディングライブラリをむ https://riptutorial.com/ja/android/topic/111/データバインディングライブラリ

190: データの

Examples

データをする

このは、のIPまたはドメインにpingをしてデータをチェックするです。

```
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;
}
```

ConnectivityManagerをしてをする

```
    public static boolean isConnectedNetwork (Context context) {

        ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
        return cm.getActiveNetworkInfo () != null && cm.getActiveNetworkInfo
().isConnectedOrConnecting ();

    }
```

データがされているにネットワークインテントをしてタスクをする

デバイスがネットワークにすると、インテントがされます。くのアプリはこれらのインテントをチェックしませんが、アプリケーションがにするようにするには、がなときにするネットワークインテントをくことができます。ネットワークをするには、たとえば、のをします。

```
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
}
```

オンラインでデータのをむ https://riptutorial.com/ja/android/topic/8670/データの

191: テーマ、スタイル、

Examples

カスタムテーマをグローバルにする

themes.xml

```
<style name="AppTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

AndroidManifest.xmlの

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">

    <!-- Activity declarations here -->

</application>
```

な、い、アクセントのをする

テーマのカラーパレットをカスタマイズできます。

フレームワーク APIの

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>
```

Appcompatサポートライブラリ および`AppCompatActivity` をすると、

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>
```

アクティビティごとにカスタムテーマをする

themes.xml

```
<style name="MyActivityTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

AndroidManifest.xmlの

```
<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>
```

オーバーカラーカラーAPI 21+

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorEdgeEffect">@color/my_color</item>
</style>
```

リップルカラーAPI 21+

5.0

ユーザーがクリックなビューをするとリップルアニメーションがされます。

あなたのビューで`?android:colorControlHighlight`りてるアプリケーションでされているのとじリップルカラーをすることができます。テーマの`android:colorControlHighlight`をすることで、このをカスタマイズできます

このエフェクトのはできます

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorControlHighlight">@color/my_color</item>
</style>
```

または、マテリアルテーマをしている

```
<style name="AppTheme" parent="android:Theme.Material.Light">
    <item name="android:colorControlHighlight">@color/your_custom_color</item>
</style>
```

ライトステータスバーAPI 23+

このは、ステータスバーアイコンののをにすることができます。

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowLightStatusBar">true</item>
</style>
```

ナビゲーションとステータスバーAPI 19+

ナビゲーションバーのにあるはにすることができます。ここでそれをするがあります。

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentNavigation">true</item>
</style>
```

このをスタイルにすると、ステータスバーのをにすることができます

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentStatus">true</item>
</style>
```

ナビゲーションバーのAPI 21+

5.0

このはナビゲーションバーBack、Home Recentボタンをむものをするためにされます。はですが、はできます。

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:navigationBarColor">@color/my_color</item>
</style>
```

テーマの

テーマをするときには、、システムによってされるテーマをして、がのアプリケーションにわせてをします。たとえば、これは`Theme.AppCompat`テーマがされるです。

```
<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>
```

このテーマは、にされたものをいて、の`Theme.AppCompat`テーマのすべてのプロパティをつようになりました。

にはショートカットもあり、はのテーマからするときにされます。

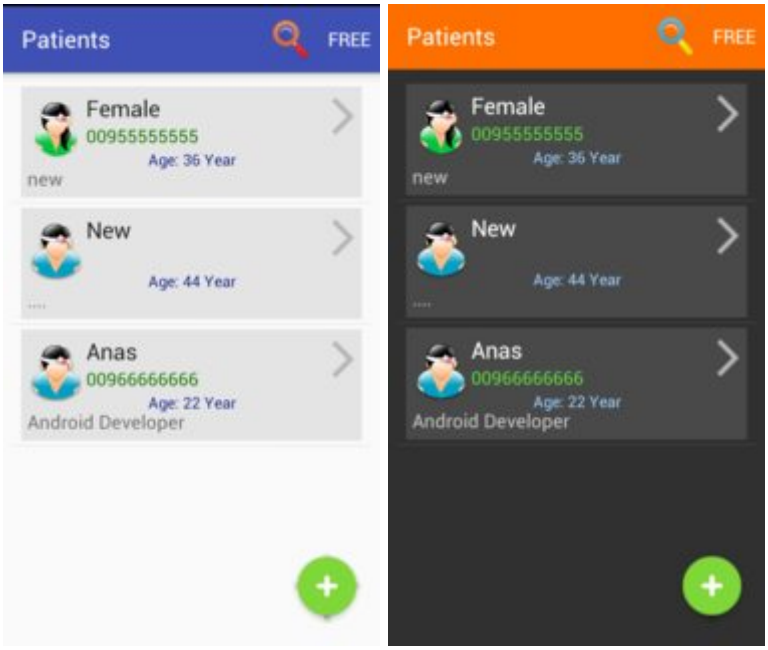
```
<style name="AppTheme.Red">
```

```
    <item name="colorAccent">@color/red</item>
</style>
```

それはに`AppTheme.`っているから`AppTheme.`そののめに、 `parent`テーマをするはなく、にされます。これは、アプリのたとえば、のアクティビティにのスタイルをするがあるにです。

1つのアプリケーションののテーマ

Androidアプリケーションでのテーマをすると、のようにすべてのテーマにカスタムをできます。



まず、のように`style.xml`テーマをする`style.xml`があります。

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">

</style>

<!--  -->
<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >

</style>
......
```

あなたのには**OneTheme**と**TwoTheme**があります。

`AndroidManifest.xml`にして、 `android:theme="@style/OneTheme"`をアプリケーションタグにすると、**OneTheme**がデフォルトのテーマになります

```
<application
        android:theme="@style/OneTheme"
        ...>
```

`attrs.xml`というのしいxmlファイルをし、のコードをします。

```
<?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) -->
```

`style.xml`って、テーマのとともにこれらのをします。

```
<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>
```

テーマのカスタムカラーをしました。このカラーをビューにしましょう。

"attr /"をしてTextViewに**custom_blue**をします。

あなたのimageViewにし、このをしてください

```
<TextView>
    android:id="@+id/txte_view"
    android:textColor="?attr/custom_blue" />
```

たちは`setTheme(R.style.TwoTheme);`だけでテーマをすることができ`setTheme(R.style.TwoTheme);`このは、この`Activity.java`ように、 `onCreate()`メソッドの`setContentView()`メソッドのにあるがあります。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setTheme(R.style.TwoTheme);
    setContentView(R.layout.main_activity);
    ....
}
```

すべてのアクティビティのテーマをにする

すべてのアクティビティのテーマをするは、MyActivityというのしいクラスをして`AppCompatActivity`クラスまたは`Activity`クラスを`setTheme(R.style.TwoTheme);`をする`setTheme(R.style.TwoTheme);` **onCreate**メソッドに

```
public class MyActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (new MySettings(this).isDarkTheme())
            setTheme(R.style.TwoTheme);
    }
}
```

に、すべてのアクティビティにして、すべてを**MyActivity**ベースクラスをします 。

```
public class MainActivity extends MyActivity {
    ....
}
```

テーマをするには、 **MyActivity**にき、 `R.style.TwoTheme`をあなたのテーマ `R.style.OneTheme` 、 `R.style.ThreeTheme` ....にしてください。

オンラインでテーマ、スタイル、をむ https://riptutorial.com/ja/android/topic/1843/テーマ-スタイル-

192: テキストビュー—のサイズ

き

ににまるようにテキストのサイズをにするTextView。

Android Oをすると、TextViewのやについてテキストのサイズをにまたはしてレイアウトにわせるようTextViewにできます。

TextViewのサイズは、コードまたはXMLでできます。

TextViewのサイズには2つのがあります。 とプリセットサイズ

Examples

Javaの

`setAutoSizeTextTypeUniformWithConfiguration()`メソッドをびします。

```
setAutoSizeTextTypeUniformWithConfiguration(int autoSizeMinTextSize, int autoSizeMaxTextSize,
int autoSizeStepGranularity, int unit)
```

XMLでは

`autoSizeMinTextSize`、 `autoSizeMaxTextSize`、および`autoSizeStepGranularity`をして、レイアウトXMLファイルのサイズのディメンションをします。

```
<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" />
```

については、GitHubのAutosizingTextViews-Demoをしてください。

プリセットサイズ

Javaの

`setAutoSizeTextTypeUniformWithPresetSizes()`メソッドをびします。

```
setAutoSizeTextTypeUniformWithPresetSizes(int[] presetSizes, int unit)
```

XMLでは

レイアウトXMLファイルで`autoSizePresetSizes`をします。

```
<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" />
```

としてリソースにアクセスするには、 *res / values / arrays.xml*ファイルでをします。

```
<array name="autosize_text_sizes">
    <item>10sp</item>
    <item>12sp</item>
    <item>20sp</item>
    <item>40sp</item>
    <item>100sp</item>
</array>
```

については、GitHubのAutosizingTextViews-Demoをしてください。

オンラインでテキストビューのサイズをむ https://riptutorial.com/ja/android/topic/9652/テキストビューのサイズ

193: テキストみげTTS

Examples

テキストベースからベース

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);
        }
    }
}
```

されるは、 `setLanguage()` メソッドに `Locale` をすることでできます。

```
tts.setLanguage(Locale.CHINESE); // Chinese language
```

サポートされるのはAndroidのレベルによってなります。 `isLanguageAvailable()` メソッドをすると、のがサポートされているかどうかをできます。

```
tts.isLanguageAvailable(Locale.CHINESE);
```

スピーチ·ピッチ·レベルは、 `setPitch()` メソッドをしてできます。デフォルトでは、ピッチは1.0です。ピッチレベルをげるには1.0よりさいをし、ピッチレベルをげるには1.0よりきいをします。

```
tts.setPitch(0.6);
```

スピーチレートは、 `setSpeechRate()` をしてできます。デフォルトのスピーチレートは1.0です。スピーチレートは、2.0にするか、0.5にしてにすることで2にすることができます。

```
tts.setSpeechRate(2.0);
```

APIをしたTextToSpeechの

TTSエンジンがしわると、にしめます。 APIレベル21では、さまざまなですことができます。

```
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;
}
```

}

オンラインでテキストみげTTSをむ https://riptutorial.com/ja/android/topic/3381/テキストみげ-tts-

194: デザインパターン

き

デザインパターンは、プログラマがアプリケーションやシステムをするによくあるをするためにできるなベストプラクティスです。

パターンは、テストみでみのパラダイムをすることでプロセスをスピードアップできます。

デザインパターンをすることで、なをきこすのあるなをぐことができ、パターンにしたコーダーやアーキテクトのコードのがします。

Examples

シングルトンクラスの

Java Singleton Pattern

シングルトンパターンをするには、さまざまなアプローチがありますが、すべてのがあります。

- プライベートコンストラクタ。クラスのインスタンスをのクラスからします。
- クラスののインスタンスであるじクラスのプライベート。
- クラスのインスタンスをすpublic staticメソッドです。これはグローバルアクセスです
- がシングルトンクラスのインスタンスをするためのポイント。

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

オブザーバーパターンはのパターンであり、くのでくされています。のはYouTubeからることができますチャンネルがきで、すべてのニュースをしてこのチャンネルからしいをたいときは、そのチャンネルをするがあります。に、このチャンネルがニュースをするたびに、あなたおよびのすべてのユーザにがされます。

オブザーバには2つのコンポーネントがあります。 1つはチャネルで、もう1つはあなたまたはのです。は、それをしているすべてのインスタンスをします。はしいイベントをさせると、これをすべてのインスタンスにします。がイベントをすると、そのイベントにするがありますたとえば、YouTubeをオンにしてしいビデオをするなど。

オブザーバパターンの

1. は、がをするをしなければならない。がイベントをさせると、イベントがしたことをにするがあります。

```
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. は、からのイベントをするメソッドをするがあります。

```
interface Subscriber {
    void doSubscribe(Channel channel);
    void doUnsubscribe(Channel channel);
    void handleEvent();  // Process the new event.
}
```

オンラインでデザインパターンをむ https://riptutorial.com/ja/android/topic/9949/デザインパターン

195: デバイスメトリック

Examples

のピクセルサイズをする

のとさをピクセルでするには、 WindowManagersのメトリックをできます。

```
// Get display metrics
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
```

これらのDisplayMetricsには、やサイズなど、デバイスにするのがされます。

```
// Get width and height in pixel
Integer heightPixels = metrics.heightPixels;
Integer widthPixels = metrics.widthPixels;
```

をする

をるために、 Windowmanagers DisplayMetricsをすることもできます。これはなです。

```
// Get density in dpi
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
int densityInDpi =  metrics.densityDpi;
```

pxをdpに、dpをpxのに

DPからピクセルへ

```
private int dpToPx(int dp)
{
    return (int) (dp * Resources.getSystem().getDisplayMetrics().density);
}
```

ピクセルから**DP**へ

```
private int pxToDp(int px)
{
    return (int) (px / Resources.getSystem().getDisplayMetrics().density);
}
```

オンラインでデバイスメトリックをむ https://riptutorial.com/ja/android/topic/4207/デバイスメトリック

196: トースト

き

Toastはさなポップアップでにするなフィードバックをし、タイムアウトにえます。メッセージになスペースだけをたし、のアクティビティはされ、インタラクティブなままです。

- トーストmakeTextコンテキストコンテキスト、CharSequenceテキスト、int
- トーストmakeTextコンテキストコンテキスト、int resId、int duration
- void setGravityint、int xOffset、int yOffset
- void show

パラメーター

パラメータ	
コンテキスト	あなたのToastをするためのコンテキスト。 `this`はにアクティビティでされ、`getActivity()`はにフラグメントでされます
テキスト	Toastにされるテキストをする CharSequenceです。 CharSequenceをするオブジェクトはすべてできます。
resId	Toastにするリソース Stringをするためにできるリソース ID。
	トーストがされるをすのフラグ。オプションは`Toast.LENGTH_SHORT`と`Toast.LENGTH_LONG`
	トーストのまたはをする。 ここのオプションをる
xOffset	トーストのオフセットをします。
yオフセット	トーストのオフセットをします。

トーストはさなポップアップでにするなフィードバックをします。メッセージになスペースだけをたし、のアクティビティはされ、インタラクティブなままです。

のToastのわりにSnackBarがあります。 SnackBarはされたビジュアルスタイルをし、ユーザーはメッセージをじたり、さらにアクションをこすことができます。については、 SnackBarのマニュアルをしてください。

https://developer.android.com/reference/android/widget/Toast.html

Examples

トーストのをする

なトーストが、のにのにってされます。このは`setGravity(int, int, int)`できます。これは、、xオフセット、およびyオフセットの3つのパラメータをけれます。

たとえば、トーストがにされるようにするは、のようにをできます。

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

トーストメッセージをする

Androidでは、ToastはユーザーにコンテキストによるフィードバックをえるためにできるなUIです。

なToastメッセージをするには、のをいます。

```
// 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();
```

または、Toastオブジェクトをせずに`Toast`インラインをするには、のをします。

```
Toast.makeText(context, "Ding! Your Toast is ready.", Toast.LENGTH_SHORT).show();
```

`show()`メソッドがUIスレッドからびされていることをしてください。のスレッドから`Toast`をしようとしているは、たとえば、 `Activity` `runOnUiThread`メソッドをできます。

そうしないと、Toastをして`UI`をしようとすると、 `RuntimeException`がスローされ、`RuntimeException`ようになります。

```
java.lang.RuntimeException: Can't create handler inside thread that has not called
Looper.prepare()
```

このをするもなは、runOnUiThreadをすることです。はのとおりです。

```
   runOnUiThread(new Runnable() {
        @Override
```

```
        public void run() {
            // Your code here
        }
    });
```

カスタムトーストの

デフォルトのToastビューをしないは、 `Toast`オブジェクトにして`setView(View)`メソッドをしてのトーストビューをできます。

まず、ToastでしたいXMLレイアウトをします。

```
<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>
```

に、ToastをするときにXMLからカスタムViewをし、 `setView`をびし`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();
```

Toastアプリケーションワイドをするスレッドセーフな

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

`MainApplication`を`manifest`にすることをれないでください。

これでトーストメッセージがされます。

```
MainApplication.getApp().showToast("Some message", Toast.LENGTH_LONG);
```

ソフトキーボードのにトーストメッセージをする

デフォルトでは、キーボードがされていてもAndroidのにToastメッセージがされます。これにより、キーボードのすぐにトーストメッセージがされます。

```
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();
}
```

トーストメッセージをするスレッドセーフなAsyncTaskの

アプリケーションをし、トーストメッセージのスレッドをにちたくないは、AsyncTasksのpost executeセクションにずしてください。

```
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();
    }

}
```

オンラインでトーストをむ https://riptutorial.com/ja/android/topic/1741/トースト

197: トーストメッセージのシングルトンクラスをする

き

トーストメッセージは、ユーザーにフィードバックをするもなです。デフォルトでは、Androidはのメッセージトーストをし、ここでメッセージとメッセージのをできます。よりカスタマイズでなトーストメッセージをするがあるは、カスタムレイアウトをしてにすることができます。さらになことに、たちがそれをするとき、Singeltonデザインパターンをすることで、カスタムトーストメッセージクラスのとがになります。

- トーストトーストContext Context
- void setDurationint duration
- void setGravityint、int xOffset、int yOffset
- void setViewビューの
- void show

パラメーター

パラメータ	
コンテキスト	トーストメッセージをするがあるコンテキスト。あなたが "this"キーワードをすか、または "getActivity"としてフラクションパスでするは、これをアクティビティでします。
る	カスタムビューをし、そのビューオブジェクトをこれにします。
	トースタのをす。すべてのがとしてGravityクラスのにされました。もなはGravity.TOP、Gravity.BOTTOM、Gravity.LEFT、Gravity.RIGHTです。
xOffset	トーストメッセージのオフセット。
yオフセット	トーストメッセージのオフセット。
	トーストショーの。 Toast.LENGTH_SHORTまたはTOAST.LENGTH_LONGのいずれかをできます

トーストメッセージは、かこっていることをユーザーにフィードバックするなです。フィードバックをするためのよりながなは、ダイアログまたはスナックバーをできます。

トーストメッセージのについては、このドキュメントをしてください。
https://developer.android.com/reference/android/widget/Toast.html

Examples

トーストマッサージののシングルトンクラスをする

トーストメッセージのためののシングルトンクラスをするはのとおりです。ユースケースごとに、、メッセージをするがあるは、のにしたでこのクラスをできます。

```
    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);
        }

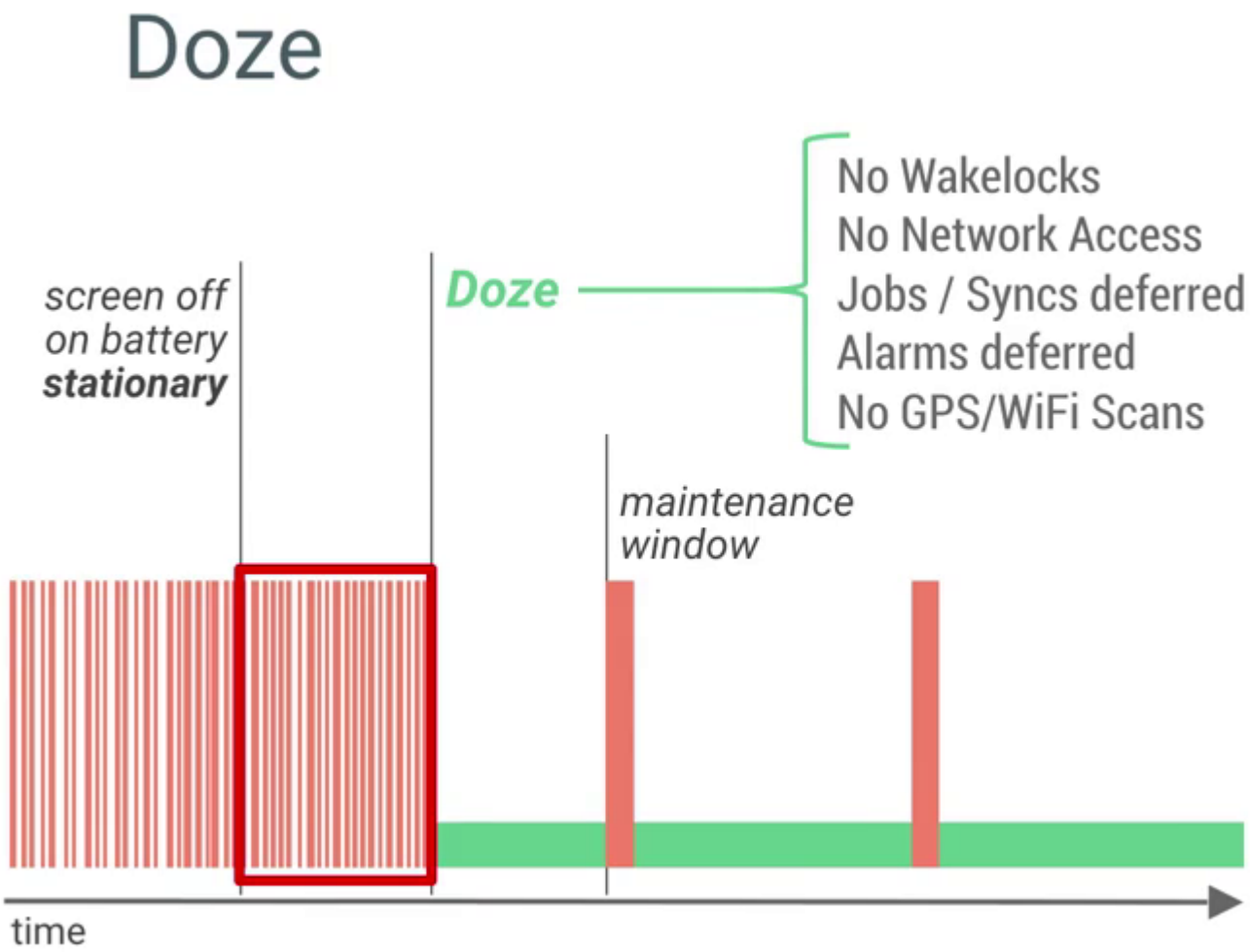
        private void showToast(View view){
            Toast toast = new Toast(context);
            toast.setGravity(Gravity.TOP,0,0); // show message in the top of the device
            toast.setDuration(Toast.LENGTH_SHORT);
            toast.setView(view);
            toast.show();
        }
    }
```

オンラインでトーストメッセージのシングルトンクラスをするをむ
https://riptutorial.com/ja/android/topic/10843/トーストメッセージのシングルトンクラスをする

198: ドーズモード

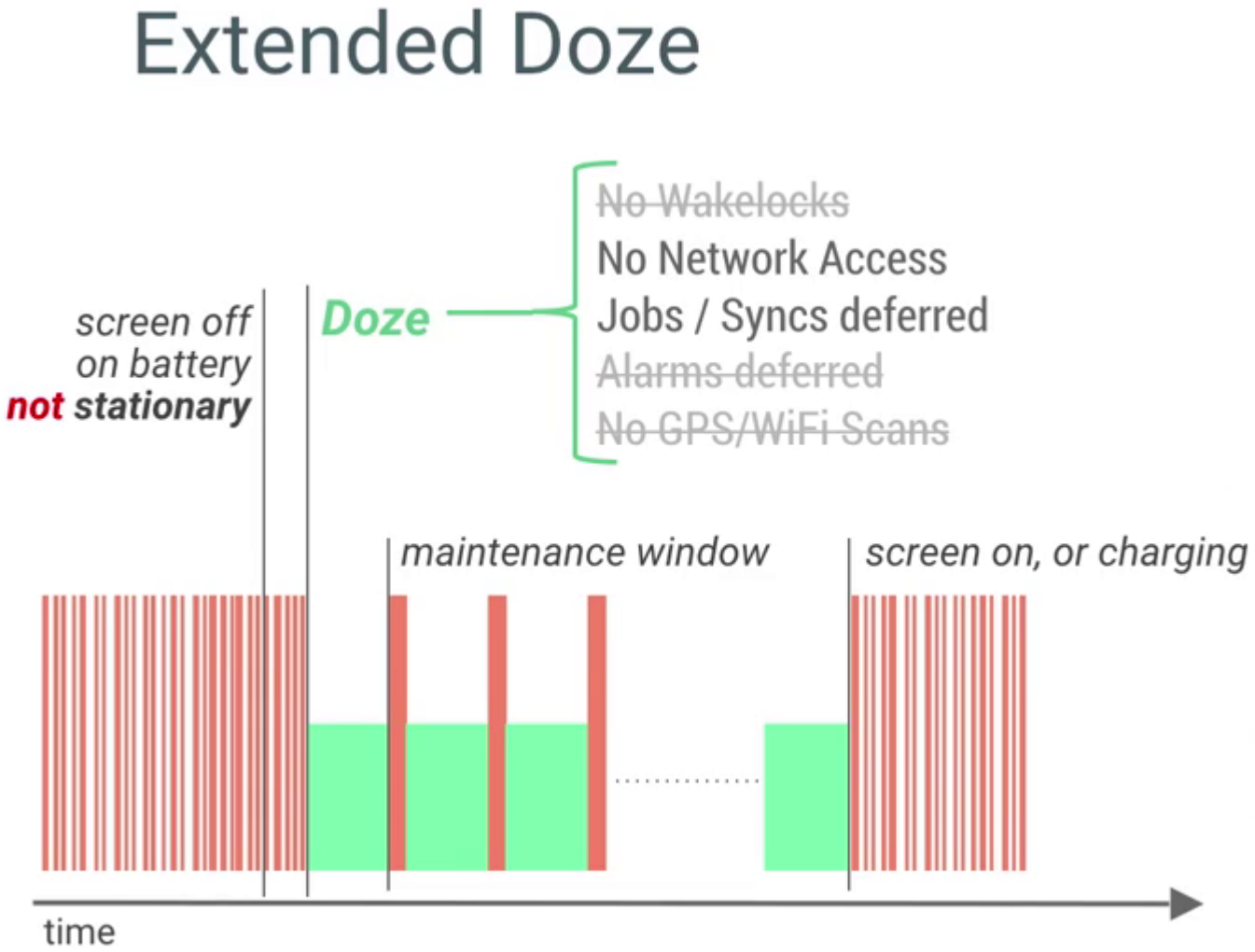
Doze Modeは、アイドルにをスリープにするのとルールです。

Android 6.0では、MarshmallowDozeモードは、がオフになってからしばらくするとアクティブになり、デバイスはしており、バッテリーでしています。



のでわかるように、Dozeモードがになると、デバイスにはウォークロック、ネットワークアクセス、ジョブ/、アラーム、GPS / Wi-Fiスキャンはしません。

Android 7.0のNougatあなたのがポケットにっているがオフで、バッテリーでしていますが、していないは、ドーズモードのもできますかだから、GoogleがExtended Doze Modeをしたは、がしてもしていないときにされるモードだ。



このでわかるように、ネットワークアクセスとジョブ/だけがになっています。 Extended Dozeは、のDozeモードにってわるものではありません。らはのかどうかによって、にきます。ここにはいがあります

	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

はのにするがあります。

- Dozeは、のいGCMGoogle Cloud Messagingメッセージユーザーがをとするのなワockロックとネットワークアクセスをするがあります。
- フォアグラウンドサービスなどはききします。

については、 https : //developer.android.com/training/monitoring-device-state/doze-standby.html をしてください。

Examples

ドーズモードをしないようにアプリをする

1. のをく
2. オープンバッテリ
3. メニューをき、「バッテリー」をします。
4. プルダウンメニューから[すべてのアプリ]をします
5. ホワイトリストにするアプリをする
6. 「しない」をする

、このアプリはされていないアプリのにされます。

アプリは`isIgnoringBatteryOptimizations()`びすことでホワイトリストにされているかどうかをできます。

プログラムによるAndroidアプリケーションのホワイトリスト

ホワイトリストにすると、アプリケーションのドーズモードがになることはありませんが、ネットワークとホールドウェイクロックをしてうことができます。

Androidアプリケーションをプログラムでホワイトリストするには、のようにします。

```
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);
}
```

のアクティビティをしたは、のコードでできます。

```
@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
        }
    }
}
```

オンラインでドーズモードをむ https://riptutorial.com/ja/android/topic/4719/ドーズモード

199: バーコードとQRコードのみり

QRCodeReaderView

Zxing

Examples

QRCodeReaderViewのZxingベース

QRCodeReaderViewは、カメラをするAndroidビューをし、プレビューにQRコードがあることをします。

zxingオープンソースのマルチフォーマット1D / 2Dバーコードライブラリをしています。

ライブラリをプロジェクトにする

build.gradleにQRCodeReaderViewをする

```
dependencies{
    compile 'com.dlazaro66.qrcodereaderview:qrcodereaderview:2.0.0'
}
```

の

- レイアウトに`QRCodeReaderView`する

```
<com.dlazaro66.qrcodereaderview.QRCodeReaderView
       android:id="@+id/qrdecoderview"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />
```

- `onQRCodeReadListener`をするアクティビティをし、それを`QrCodeReaderView`リスナとしてします。
- ライブラリをするには、カメラがです。
  https://developer.android.com/training/permissions/requesting.html

アクティビティでは、のようにできます。

```
   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();
    }
}
```

オンラインでバーコードとQRコードのみりをむ https://riptutorial.com/ja/android/topic/6067/バーコードとqrコードのみり

200: パーセルブル

き

Parcelableは、あなたでシリアルをするAndroidのインターフェイスです。これは、Serializableよりはるかにで、のJavaシリアルスキームのいくつかのをするためにされました。

パーセルにフィールドをきむは、カスタムオブジェクトをするときにパーセルからみすとじでなければならないことをえておくことがです。

パーセルなインターフェイスのサイズは1 MBにされています。つまり、1MBのをめるパーセルにれたオブジェクトまたはオブジェクトのみわせは、でることになります。これはするのがしいかもしれないので、どのようなのオブジェクトをにするかをえておいてください。らがきなツリーをっているは、データをすのをしてください。

Examples

カスタムオブジェクトをする**Parcelable**。

```
/**
 * Created by Alex Sullivan on 7/21/16.
 */
public class Foo implements Parcelable
{
    private final int myFirstVariable;
    private final String mySecondVariable;
    private final long myThirdVariable;

    public Foo(int myFirstVariable, String mySecondVariable, long myThirdVariable)
    {
        this.myFirstVariable = myFirstVariable;
        this.mySecondVariable = mySecondVariable;
        this.myThirdVariable = myThirdVariable;
    }

    // Note that you MUST read values from the parcel IN THE SAME ORDER that
    // values were WRITTEN to the parcel! This method is our own custom method
    // to instantiate our object from a Parcel. It is used in the Parcelable.Creator variable
we declare below.
    public Foo(Parcel in)
    {
        this.myFirstVariable = in.readInt();
        this.mySecondVariable = in.readString();
        this.myThirdVariable = in.readLong();
    }

    // The describe contents method can normally return 0. It's used when
    // the parceled object includes a file descriptor.
    @Override
    public int describeContents()
    {
```

```
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags)
    {
        dest.writeInt(myFirstVariable);
        dest.writeString(mySecondVariable);
        dest.writeLong(myThirdVariable);
    }

    // Note that this seemingly random field IS NOT OPTIONAL. The system will
    // look for this variable using reflection in order to instantiate your
    // parceled object when read from an Intent.
    public static final Parcelable.Creator<Foo> CREATOR = new Parcelable.Creator<Foo>()
    {
        // This method is used to actually instantiate our custom object
        // from the Parcel. Convention dictates we make a new constructor that
        // takes the parcel in as its only argument.
        public Foo createFromParcel(Parcel in)
        {
            return new Foo(in);
        }

        // This method is used to make an array of your custom object.
        // Declaring a new array with the provided size is usually enough.
        public Foo[] newArray(int size)
        {
            return new Foo[size];
        }
    };
}
```

の**Parcelable**オブジェクトをむ**Parcelable**オブジェクト

にクラスをむクラスの

```
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;
    }
}
```

はのパーセルブルクラスです。

ParcelableでEnumをする

```
/**
 * 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
}
```

にしいをしても、にされたにはしないため、これはをするたとえばにはしています。

オンラインでパーセルブルをむ https://riptutorial.com/ja/android/topic/1849/パーセルブル

201: ハードウェアボタンイベント/インテント PTT、LWPなど

き

いくつかのアンドロイドデバイスには、メーカーがカスタムボタンをしています。これにより、にAppsをハードウェアデバイスのターゲットにするときに、がこれらのボタンをするしいがかれます。

このトピックでは、インテントレシーバをしてリッスンできるインテントがいたボタンについてします。

Examples

Sonim Devices

Sonimデバイスは、さまざまなカスタムボタンをモデルによってさまざまにさせています。

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
```

ボタンの

これらのインテントをするには、Phone-Settingsでボタンをアプリにりてるがあります。 Sonimには、インストールにボタンをアプリケーションにするがあります。これをうには、をって、マニフェストにのようにパッケージのキーをめるがあります

```
<meta-data
    android:name="app_key_green_data"
    android:value="your-key-here" />
```

RugGearデバイス

PTTボタン

```
android.intent.action.PTT.down
android.intent.action.PTT.up
```

みRG730、RG740A

オンラインでハードウェアボタンイベント/インテントPTT、LWPなどをむ
https://riptutorial.com/ja/android/topic/10418/ハードウェアボタンイベント-インテント-ptt-lwpなど-

202: パスワードをにする

Examples

されたパスワードによるAESの

このでは、パスワードのにAESアルゴリズムをしています。のさは128ビットです。

SecureRandomクラスをしてsaltをしますSecureRandomは、パスワードとみわされてをします。されるクラスは、すでにAndroidパッケージjavax.cryptoおよびjava.securityします。

キーがされると、このキーをにするか、しなければなりません。をS_KEYプリファレンスにしています。に、 ENCRYPT_MODEでされると、 CipherクラスのdoFinalメソッドをしてパスワードがされます。に、されたパスワードは、バイトからにされ、のプリファレンスにされます。されたパスワードをするためにされるキーは、のでパスワードのにできます。

```
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;
    }
}
```

オンラインでパスワードをにするをむ https://riptutorial.com/ja/android/topic/9093/パスワードをにする

203: バターナイフ

き

Butterknifeは、をしてボイラープレートコードをするビューバインディングツールです。このツールは、SquareのJake Whartonによってされたもので、ビューをうに`findViewById(R.id.view)`ようなコードのりしを`findViewById(R.id.view)`するためにされています。

にするために、Butterknifeはライブラリではありません。 Butterknifeはコンパイルにコードをします。 Androidアノテーションでわれたとによくています。

バターナイフ

をしてコードをするAndroidビューのフィールドとメソッドバインディング

- フィールドの@BindViewをしてfindViewByIdびしをします。
- リストまたはののビューをグループする。アクション、セッター、またはプロパティをして、それらをにすべてします。
- @OnClickなどのメソッドにをけることによって、リスナーののクラスをします。
- フィールドのリソースをしてリソースのをします。

http : //jakewharton.github.io/butterknife/

ライセンス

2013 Jake Wharton

Apache License、Version 2.0「ライセンス」のでライセンスされています。ライセンスにわないり、このファイルをすることはできません。あなたはライセンスのコピーを

http://www.apache.org/licenses/LICENSE-2.0

またはによるがなをき、ライセンスにづいてされるソフトウェアは、またはにいかなるのまたはもなく「」でされます。ライセンスにづいておよびをするのについては、ライセンスをしてください。

Examples

あなたのプロジェクトで**ButterKnife**をする

`android-apt`プラグインをめるようにプロジェクトレベルの`build.gradle`をします。

```
buildscript {
   repositories {
```

```
      mavenCentral()
   }

   dependencies {
      classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
   }
}
```

に、モジュールレベルの`build.gradle` `android-apt`プラグインをし、ButterKnifeのをします。

```
apply plugin: 'android-apt'

android {
    ...
}

dependencies {
      compile 'com.jakewharton:butterknife:8.5.1'
      annotationProcessor 'com.jakewharton:butterknife-compiler:8.5.1'
}
```

バージョン2.2.0のしいJackコンパイラをしているは、 `android-apt`プラグインはありません。コンパイラのをするときは、aptを`annotationProcessor`にきえることができます。

あなたがそれらををれるべきではありませんButterKnifeをするためには、 `onCreate()`あなたのやの`onCreateView()`あなたのフラグメントのを

```
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;
  }

}
```

バージョンのスナップショットは、 Sonatypeのスナップショットリポジトリにあります 。

は、ライブラリプロジェクトで**ButterKnife**をするためになステップです

ライブラリプロジェクトでButterKnifeをするには、プラグインをプロジェクトレベルの `build.gradle`にします。

```
buildscript {
    dependencies {
        classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
    }
}
```

...あなたのライブラリレベル`build.gradle`にのをしてモジュールにします

```
apply plugin: 'com.android.library'
// ...
apply plugin: 'com.jakewharton.butterknife'
```

、あなたがしていることをし`R2`のわりに、 `R`すべて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
  }

}
```

ButterKnifeをしたビューのバインド

`@BindView`とバターナイフのビューIDでフィールドにをけると、レイアウトのするビューをつけて

にキャストできます。

バインディングビュー

アクティビティのビューのバインド

```
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...
  }
}
```

フラグメントのビューのバインド

```
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();
    }
}
```

ダイアログのビューのバインド

`ButterKnife.findById`をして、ビュー、アクティビティ、またはダイアログにするビューをつけることができます。ジェネリックをしてりのをし、にキャストをします。

```
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);
```

ViewHolderでのビューのバインド

```
static class ViewHolder {
    @BindView(R.id.title) TextView name;
    @BindView(R.id.job_title) TextView jobTitle;

    public ViewHolder(View view) {
      ButterKnife.bind(this, view);
    }
  }
```

バインディングリソース

にビューをするのにであるから、はまた、このようなでされたものなどのリソースバインドするButterKnifeをすることができます`strings.xml`、 `drawables.xml`、 `colors.xml`、 `dimens.xml`など

```
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);
    }

}
```

ビューリストのバインド

のビューをリストまたはにグループできます。これは、にのビューにして1つのアクションをするがあるににちます。

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

オプションのバインディング

デフォルトでは、 `@Bind`バインディングとリスナーバインディングのがです。ターゲットビューがつからないは、がスローされます。しかし、ビューがするかどうかわからないは、 `@Nullable`アノテーションをフィールドにするか、 `@Optional`アノテーションをメソッドにして、このをし、オプションのバインディングをすることができます。

```
@Nullable
@BindView(R.id.might_not_be_there) TextView mightNotBeThere;

@Optional
@OnClick(R.id.maybe_missing)
void onMaybeMissingClicked() {
  // TODO ...
}
```

ButterKnifeをしたバインディングリスナー

OnClickリスナー

```
@OnClick(R.id.login)
public void login(View view) {
  // Additional logic
}
```

リスナメソッドへのはすべてオプションです。

```
@OnClick(R.id.login)
public void login() {
   // Additional logic
}
```

のタイプはにキャストされます

```
@OnClick(R.id.submit)
public void sayHi(Button button) {
  button.setText("Hello!");
}
```

のイベントのためののバインディングののID

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

カスタムビューは、IDをしないことによって、のリスナーにバインドできます。

```
public class CustomButton extends Button {
  @OnClick
  public void onClick() {
    // TODO
  }
}
```

ButterKnifeでのビューのバインド

フラグメントは、アクティビティよりもビューのライフサイクルがなります。 onCreateViewでフラグメントをバインドするときは、onDestroyViewでビューをnullにします。バターナイフはバインドをびしてこれをうためUnbinderのインスタンスをします。なライフサイクルコールバックでunbindメソッドをびします。

```
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();
  }
}
```

> onDestroyViewでunbindをびすはありませんが、アプリがきなバックスタックをつはかなりのメモリをできるのでおめします。

Android Studio ButterKnifeプラグイン

Android ButterKnife Zelezny

> アクティビティ/フラグメント/アダプタのされたレイアウトXMLからButterKnifeインジェクションをするためのプラグイン。

your_xml_layou `(R.layout.your_xml_layou` をクリックし、 *Generate*メニューにButterknifeインジェクタオプションがまれていないことをしてください。

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

リンク Jetbrainsプラグイン Android ButterKnife Zelezny

オンラインでバターナイフをむ https://riptutorial.com/ja/android/topic/1072/バターナイフ

204: パフォーマンスの

き

Appsのパフォーマンスは、ユーザーエクスペリエンスのなです。 UIスレッドでするようないパターンをけ、でのいアプリケーションをするをびましょう。

Examples

ViewHolderパターンをしてViewルックアップをする

にListViewでは、スクロールにfindViewById()びしをすぎると、パフォーマンスのがするがあります。 ViewHolderパターンをすると、これらのをしてListViewパフォーマンスをさせることができます。

リストに1つのTextViewがまれているは、インスタンスをするViewHolderクラスをします。

```
static class ViewHolder {
    TextView myTextView;
}
```

リストをするときに、リストにViewHolderオブジェクトをします。

```
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;
}
```

このパターンをすると、 findViewById()はしいViewがされているときにのみびされ、 ListViewはよりにListViewをリサイクルできます。

オンラインでパフォーマンスのをむ https://riptutorial.com/ja/android/topic/8711/パフォーマンスの

205: ハンドラ

ハンドラは、されたのにコードをするためににできます。また、Runnableのrunメソッドから Handler.postDelayedメソッドをびすことによって、されたにコードをりしするにもです。

Examples

したのにハンドラをしてコードをする

1.5にコードをする

```
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
```

コードを**1**ごとにりしする

```
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とスレッドの

`Handler`は、 `Message`と`Runnable`をスレッドのメッセージキューにするため、のスレッドでイベントベースのをするのはです。 `Looper`をつすべてのスレッドは、メッセージをしてできます。`HandlerThread`は、そのような`Looper`をするスレッドです。たとえば、メインスレッドUIスレッドは`HandlerThread`をします。

のスレッドのハンドラの

```
Handler handler = new Handler();
```

メインスレッドのハンドラのUIスレッド

```
Handler handler = new Handler(Looper.getMainLooper());
```

のスレッドからメインスレッドに**Runnable**をする

```
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();
```

の**HandlerThread**のハンドラをし、それにイベントをする

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

ハンドラをからする

ハンドラのをするには、でされているファイルをして、ハンドラにけられたコールバックをします。

```
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);
}
```

ハンドラをしてタイマーをするjavax.swing.Timerと

これは、おきにコードをするのあるゲームなどをするにです。

```
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 ();
    }
}
```

```
Timer timer = new Timer(new Runnable() {
    public void run() {
        System.out.println("Hello");
    }
}, 1000, true)
```

このコードは "Hello"をします。

オンラインでハンドラをむ https://riptutorial.com/ja/android/topic/1425/ハンドラ

206: ピカソ

き

PicassoはAndroidのライブラリです。これはスクエアによってされされています 。のからをするプロセスをします。ライブラリは、のHTTPリクエストからイメージのキャッシングまで、プロセスのすべてのをします。くの、このきちんとしたライブラリをするためになコードはわずかです。

Picassoは、Androidのなダウンロードおよびキャッシングライブラリです。
プロジェクトにライブラリをするには、 このにいます。

ウェブサイト

- ソース
- Doc
- ログの

Examples

Androidプロジェクトに Picassoライブラリをする

から

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>
```

プレースホルダとエラー

Picassoは、ダウンロードとエラーののプレースホルダをオプションとしてサポートしています。

また、ダウンロードをするためのコールバックもします。

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

エラーのプレースホルダーがされるに、が3されます。

サイズと

```
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);
```

ピカソのアバター

ここには、 オリジナルにづいたピカソサークルトランスフォームクラスのがあり、いボーダーがされており、スタッキングのオプションのセパレータのもまれています。

```
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);

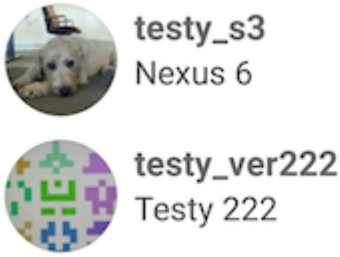
        // Optional separator for stacking:
        if (mCircleSeparator) {
            Paint paintBorderSeparator = new Paint();
            paintBorderSeparator.setStyle(Style.STROKE);
            paintBorderSeparator.setColor(Color.parseColor("#ffffff"));
            paintBorderSeparator.setAntiAlias(true);
            paintBorderSeparator.setStrokeWidth(4);
            canvas.drawCircle(r, r, r+1, paintBorderSeparator);
        }

        squaredBitmap.recycle();
        return bitmap;
    }

    @Override
    public String key() {
        return "circle";
    }
}
```

ィメ―ジをロ―ドするときに、 `this`イメ―ジをするをします `this`はActivity Contextで、 `url`はロ―ドするイメ―ジの`url`をつStringです。

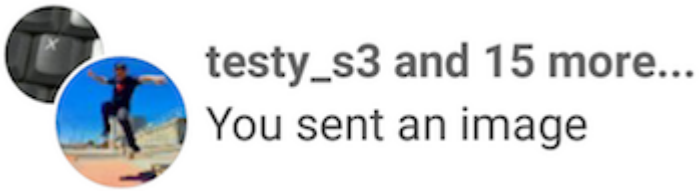
```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform())
    .into(ivAvatar);
```



セパレータでするは、のイメージのコンストラクタを`true`します。

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform(true))
    .into(ivAvatar);
```

FrameLayoutの2つのImageView



Picasaでキャッシュをにする

```
Picasso.with(context)
.load(uri)
.networkPolicy(NetworkPolicy.NO_CACHE)
.memoryPolicy(MemoryPolicy.NO_CACHE)
.placeholder(R.drawable.placeholder)
.into(imageView);
```

ストレージからをみむ

```
String filename = "image.png";
String imagePath = getExternalFilesDir() + "/" + filename;

Picasso.with(context)
    .load(new File(imagePath))
    .into(imageView);
```

ピカソをったビットマップのダウンロード

`Picasso`をしてを`Bitmap`としてダウンロードするは、のコードをにしてください。

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

ピカソをしたリクエストのキャンセル

によっては、ダウンロードがするにPicassoでのダウンロードをキャンセルするがあります。

これは、のダウンロードがするにビューがのビューにしたなど、さまざまなでするがあります。

この、 `cancelRequest()` メソッドをしてイメージのダウンロードをりすことができます。

```
ImageView imageView;

//......

Picasso.with(imageView.getContext()).cancelRequest(imageView);
```

PicasaをHtml.fromHtmlのImageGetterとしてする

PicasaをHtml.fromHtmlのImageGetterとしてする

```
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);
    }

    //-----------------------------------------------------------------//
```

```
}
}
```

いはです

```
Html.fromHtml(textToParse, new PicassoImageGetter(picasso, textViewTarget), null);
```

オフラインのディスクキャッシュをにしてから、オンラインにしてイメージをしてください

にアプリケーションモジュールのgradleビルドファイルにOkHttpをします

```
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'
```

に、アプリケーションをするクラスをします。

```
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);

    }
}
```

マニフェストファイルにのようにします。

```
<application
        android:name=".Global"
        .. >

</application>
```

の

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

のへのリンク

オンラインでピカソをむ https://riptutorial.com/ja/android/topic/2172/ピカソ

207: ビットマップキャッシュ

き

メモリなビットマップキャッシングこれは、アプリケーションがアニメーションをしてGCクリーンアップにされ、アプリケーションがユーザーにとっていようにえるににです。キャッシュは、なオブジェクトをすることをにする。オブジェクトをメモリにロードする、これをオブジェクトのキャッシュとえることができます。アンドロイドでビットマップをするのはしいです。りしするは、bimapをキャッシュするがです。

- `LruCache<String, Bitmap> mMemoryCache;//declaration of LruCache object.`
- void addBitmapToMemoryCacheString key、Bitmap bitmap{} //ビットマップをキャッシュメモリにするメソッドの
- ビットマップgetBitmapFromMemCacheString key{} //キャッシュからbimapをするメソッドの。

パラメーター

パラメータ	
キー	メモリキャッシュにビットマップをするためのキー
ビットマップ	メモリにキャッシュされるビットマップ

Examples

LRUキャッシュをしたビットマップキャッシュ

LRUキャッシュ

のコードは、をキャッシュするためのLruCacheクラスのをしています。

```
private LruCache<String, Bitmap> mMemoryCache;
```

ここで、はビットマップのキーです。

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

ビットマップをメモリキャッシュにする

```
public void addBitmapToMemoryCache(String key, Bitmap bitmap) {
if (getBitmapFromMemCache(key) == null) {
        mMemoryCache.put(key, bitmap);
    }
}
```

メモリキャッシュからビットマップをする

```
public Bitmap getBitmapFromMemCache(String key) {
    return mMemoryCache.get(key);
}
```

ビットマップをイメージビューにロードするには、 に**getBitmapFromMemCache "**パスキー**"**をします。

オンラインでビットマップキャッシュをむ https://riptutorial.com/ja/android/topic/9901/ビットマップキャッシュ

208: ビットマップをにみむ

き

このトピックは、にAndroidデバイスでビットマップをみむことにしています。

ビットマップのみみについては、そこからみまれるにがあります。ここでは、Androidデバイスでリソースからビットマップをみむについてします。えばギャラリーから。

でするでこれをします。

- `<uses-permission>` **- >** `<uses-permission>`されるタグ。
- `android:name` **- >**リクエストするパーミッションのをするための。
- `android.permission.READ_EXTERNAL_STORAGE` **- >**これはシステムです
- "android.permission.CAMERA"または "android.permission.READ_CONTACTS"

Examples

Androidデバイスからリソースからをみみます。インテントの。

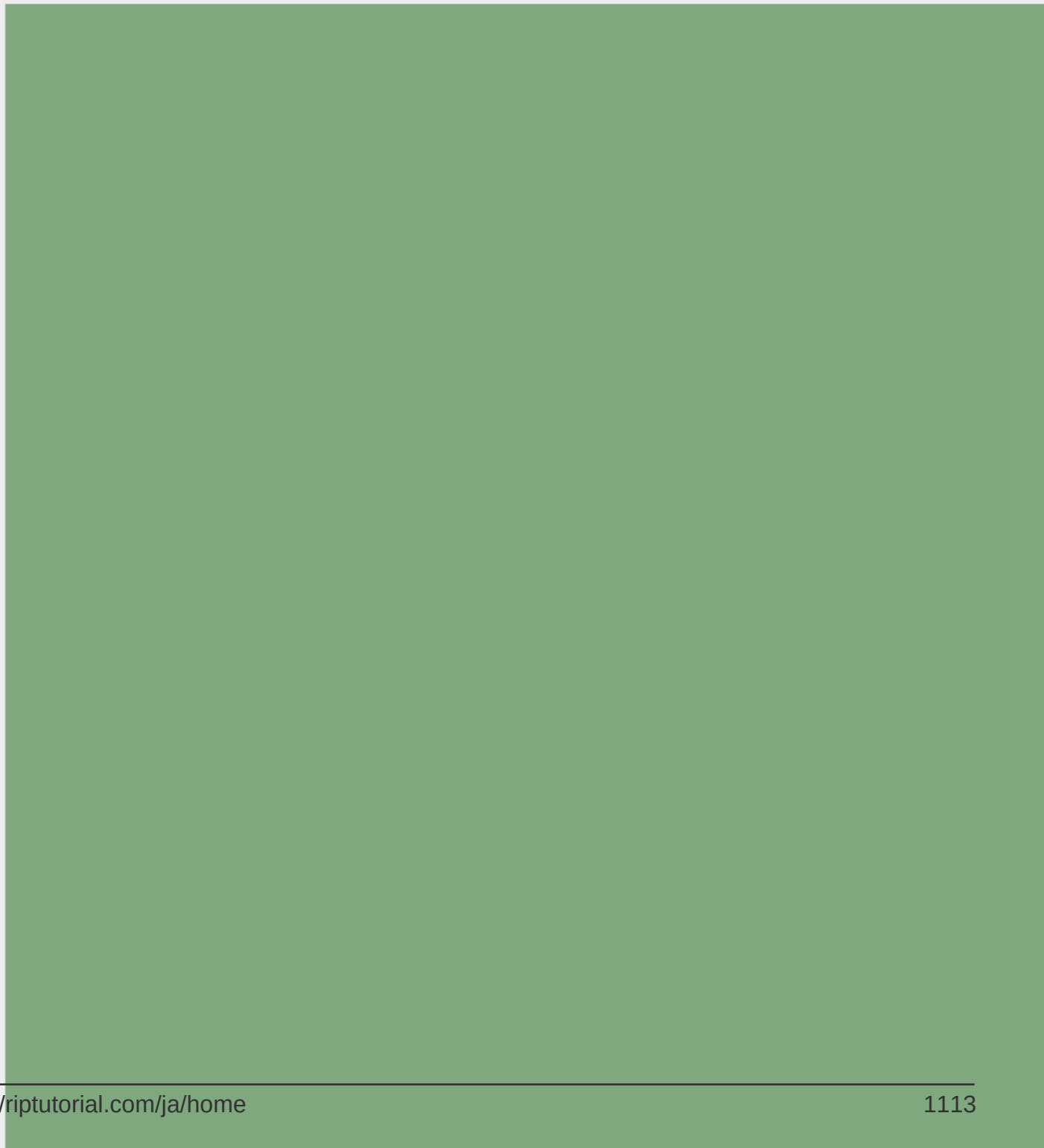
インテントをしてギャラリーからイメージをロードする。

1. にあなたはをとします

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

2. のコードをして、のようなレイアウトをします。

LoadImageFrmGallery



ビットマップをにみむ

209: ファイアベース

き

Firebaseは、がのアプリケーションをするのをするためのツールとインフラストラクチャをえた、モバイルおよびWebアプリケーションプラットフォームです。

Firebaseクラウドメッセージング、Firebase Auth、リアルタイムデータベース、Firebase Storage、Firebaseホスティング、AndroidFirebaseテストラボ、Firebaseクラッシュレポート。

Firebase -

Firebaseのにするよりくのトピックとをつけることができるのタグがあります。

そののトピック

- Firebaseリアルタイムデータベース
- Firebase App Indexing
- Firebaseクラッシュレポート
- Firebase Cloud Messaging

Examples

Firebaseユーザをする

```
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;
    }
}
```

Firebaseユーザにメールとパスワードでサインイン

```
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));
    }
}
```

Firebaseのパスワードリセットメールをする

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

Firebaseユーザのメールをする

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

パスワードをする

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

Firebaseユーザの

```
public class ReAuthenticateDialogFragment extends DialogFragment {

    @BindView(R.id.et_dialog_reauthenticate_email)
    EditText mEditTextEmail;
    @BindView(R.id.et_dialog_reauthenticate_password)
    EditText mEditTextPassword;
    private OnReauthenticateSuccessListener mOnReauthenticateSuccessListener;

    @OnClick(R.id.btn_dialog_reauthenticate)
    void onReauthenticateClick() {

        FormValidationUtils.clearErrors(mEditTextEmail, mEditTextPassword);

        if (FormValidationUtils.isBlank(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "Please enter valid email");
```

```
            return;
        }

        if (TextUtils.isEmpty(mEditTextPassword.getText())) {
            FormValidationUtils.setError(null, mEditTextPassword, "Please enter password");
            return;
        }

        reauthenticateUser(mEditTextEmail.getText().toString(),
mEditTextPassword.getText().toString());
    }

    private void reauthenticateUser(String email, String password) {
        DialogUtils.showProgressDialog(getActivity(), "Re-Authenticating", "Please wait...",
false);
        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        AuthCredential authCredential = EmailAuthProvider.getCredential(email, password);
        firebaseUser.reauthenticate(authCredential)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            mOnReauthenticateSuccessListener.onReauthenticateSuccess();
                            dismiss();
                        } else {
                            ((BaseAppCompatActivity)
getActivity()).showToast(task.getException().getMessage());
                        }
                    }
                });
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        mOnReauthenticateSuccessListener = (OnReauthenticateSuccessListener) context;
    }

    @OnClick(R.id.btn_dialog_reauthenticate_cancel)
    void onCancelClick() {
        dismiss();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.dialog_reauthenticate, container);
        ButterKnife.bind(this, view);
        return view;
    }

    @Override
    public void onResume() {
        super.onResume();
        Window window = getDialog().getWindow();
        window.setLayout(WindowManager.LayoutParams.MATCH_PARENT,
WindowManager.LayoutParams.WRAP_CONTENT);
    }

    interface OnReauthenticateSuccessListener {
```

```
        void onReauthenticateSuccess();
    }
}
```

Firebaseストレージ

このでは、のをできます。

1. Firebase Storageにする
2. "images"というのディレクトリをします。
3. ディレクトリにファイルをアップロードする
4. imagesディレクトリからファイルをダウンロードする
5. ディレクトリからファイルをする

```
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();
    }
}
```

デフォルトでは、Firebase Storageルールはをします。ユーザーがされたにのみ、はFirebase Storageでをできます。それのはできません。はこのデモでルールをにしました。は、ルールはのようになりました。

```
 service firebase.storage {
   match /b/**something**.appspot.com/o {
     match /{allPaths=**} {
       allow read, write: if request.auth != null;
     }
   }
 }
```

しかし、をスキップするようにしました。

```
service firebase.storage {
  match /b/**something**.appspot.com/o {
    match /{allPaths=**} {
      allow read, write;
    }
  }
```

```
}
```

Firebase Cloud Messaging

まず、このトピックでするにって、AndroidプロジェクトにFirebaseをするプロジェクトをセットアップするがあります。

FirebaseとFCM SDKをセットアップする

アプリレベルの`build.gradle`ファイルにFCMをする

```
dependencies {
 compile 'com.google.firebase:firebase-messaging:11.0.4'
}
```

そして、のこれはですにをしてください

```
// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

アプリのマニフェストをする

アプリのマニフェストにをします

- `FirebaseMessagingService`をするサービス。これは、バックグラウンドでアプリのをするのメッセージをするにです。
- `FirebaseInstanceIdService`をしてトークンの、ローテーション、およびをするサービス。

えば

```
    <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>
```

ここには2つのサービスのながあります。

のトークンをするには、 `FirebaseInstanceIdService`クラスをし、 `onTokenRefresh()`メソッドをオー

バーライドします。

```
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
    }
}
```

メッセージをするには、 `FirebaseMessagingService`をして`onMessageReceived`メソッドをオーバーライドするサービスをします。

```
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
    }
```

Firebaseでは、「AppVersion、フリーユーザ、ユーザ、またはのルール」のようなるいでユーザをグループし、FireBaseのトピックをしてのグループにをできます。
トピックをしてユーザーをする

```
FirebaseMessaging.getInstance().subscribeToTopic("Free");
```

FireBaseコンソールで、トピックでをする

のトピック「 Firebase Cloud Messaging」の 。

Firebaseプロジェクトをして Androidアプリにするためにな ドキュメントにづいています です。

あなたのアプリにFirebaseをする

1. FirebaseコンソールでFirebaseプロジェクトをし、[**Create New Project**]をクリックします。
2. **Android**アプリに [**Firebase**を]をクリックし、にいます。
3. プロンプトがされたら、 アプリのパッケージをします。
   アプリがしているパッケージをすることはです。 Firebaseプロジェクトにアプリをするときにのみできます。
4. に`google-services.json`ファイルをダウンロードします。いつでもこのファイルをダウンロードできます。
5. まだしていないは、 `google-services.json`ファイルをプロジェクトのモジュールフォルダは`app/`にコピーします。

のステップでは、SDKをしてFirebaseライブラリをプロジェクトにします。

SDKをする

Firebaseライブラリをのプロジェクトにするには、Android Studioプロジェクトをするためのなタスクをいくつかするがあります。 Firebaseをあなたのアプリにするに、すでにこれをっているかもしれません。

1. ルートレベルの`build.gradle`ファイルにルールをして、 **google-services**プラグインをします。

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.1.0'
    }
}
```

に、モジュールのGradleファイルは`app/build.gradle` に、Gradleプラグインをにするために、ファイルのにプラグインをします。

```
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'
```

のステップでは、Firebase SDKのを、Firebaseのさまざまなでできる 1つまたはのライブラリをしてします。

Gradle Dependency Line	サービス
com.google.firebasefirebase-core11.0.4	アナリティクス
com.google.firebasefirebase-database11.0.4	リアルタイムデータベース
com.google.firebasefirebase-storage11.0.4	ストレージ
com.google.firebasefirebase-crash11.0.4	クラッシュレポート
com.google.firebasefirebase-auth11.0.4	
com.google.firebasefirebase-messaging11.0.4	クラウドメッセージング/
com.google.firebasefirebase-config11.0.4	リモート
com.google.firebasefirebase-invites11.0.4	/ダイナミックリンク
com.google.firebasefirebase-ads11.0.4	AdMob
com.google.android.gmsplay-services-appindexing11.0.4	アプリのインデックス

Firebaseリアルタイムデータベースデータの/

のためのをしましょう

```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

したら、データベースアドレスをしてをします。えば

https://your-project.firebaseio.com/から**https://your-project.firebaseio.com/chat**

たちはAndroidデバイスからこのにデータをします。あなたはデータベースタブ、フィールド...などをするはありませんあなたはFirebaseにJavaオブジェクトをおりしますと、にされます

データベースにするすべてのをむJavaオブジェクトをします。

```
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;
    }
}
```

あなたのでは

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

をするには

```
ChatMessage msg = new ChatMessage("user1", "Hello World!");
reference.push().setValue(msg);
```

データベースでしたをするには

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

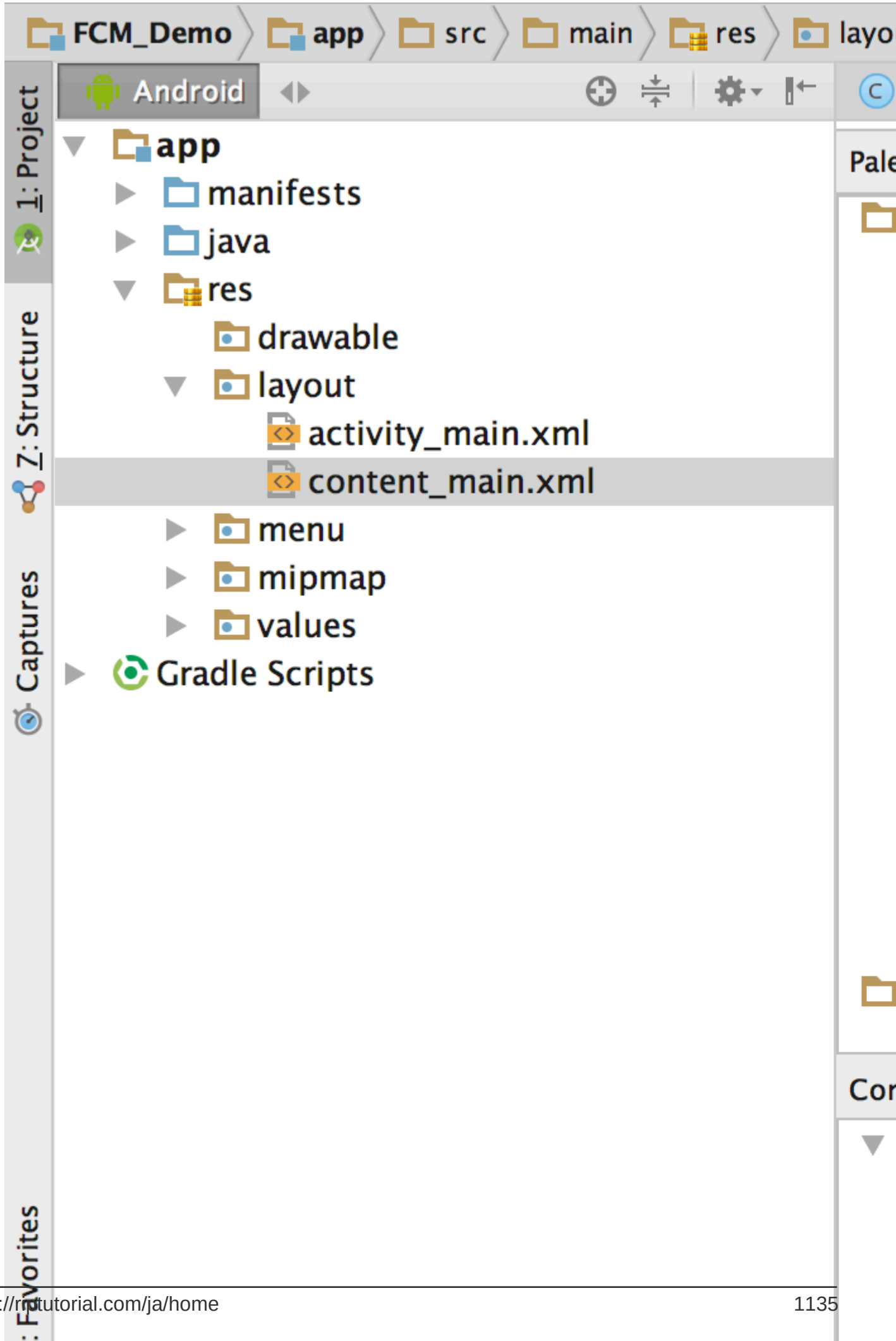


FCMベースののデモ

このでは、Firebase Cloud MessagingFCMプラットフォームのをします。 FCMはGoogle Cloud MessagingGCMのです。アプリユーザーからのC2D_MESSAGEはありません。

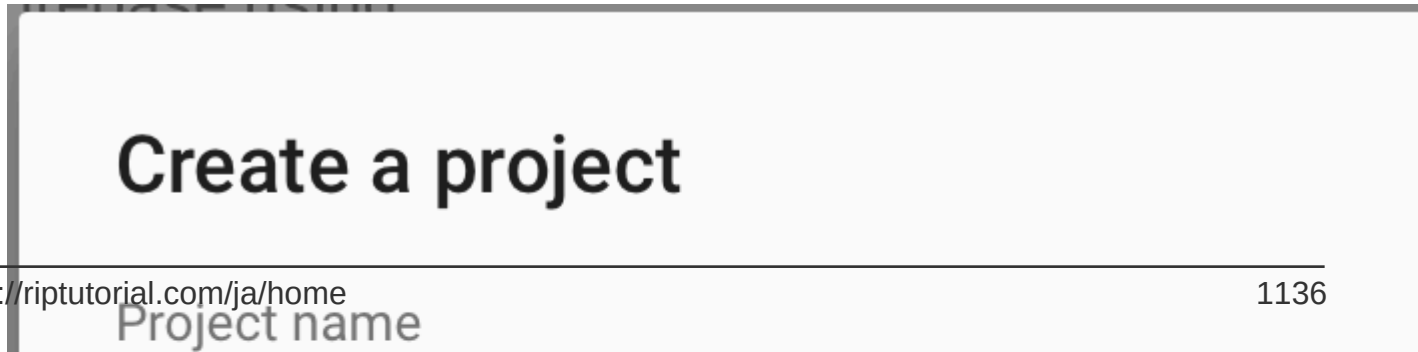
FCMをするはのとおりです。

1. Android StudioでサンプルのHello worldプロジェクトをするあなたのAndroidスタジオはののようになります。


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. のステップは、firebaseプロジェクトをすることです。 https://console.firebase.google.com にアクセスして、にできるように、じのプロジェクトをします。

にアクセスして、にできるように、じのプロジェクトをします。



3. すぐあなたがしたサンプルのアンドロイドプロジェクトにfirebaseをするです。プロジェクトのパッケージとデバッグSHA-1オプションがです。

   a。パッケージ - アンドロイドマニフェストXMLファイルからつけることができます。

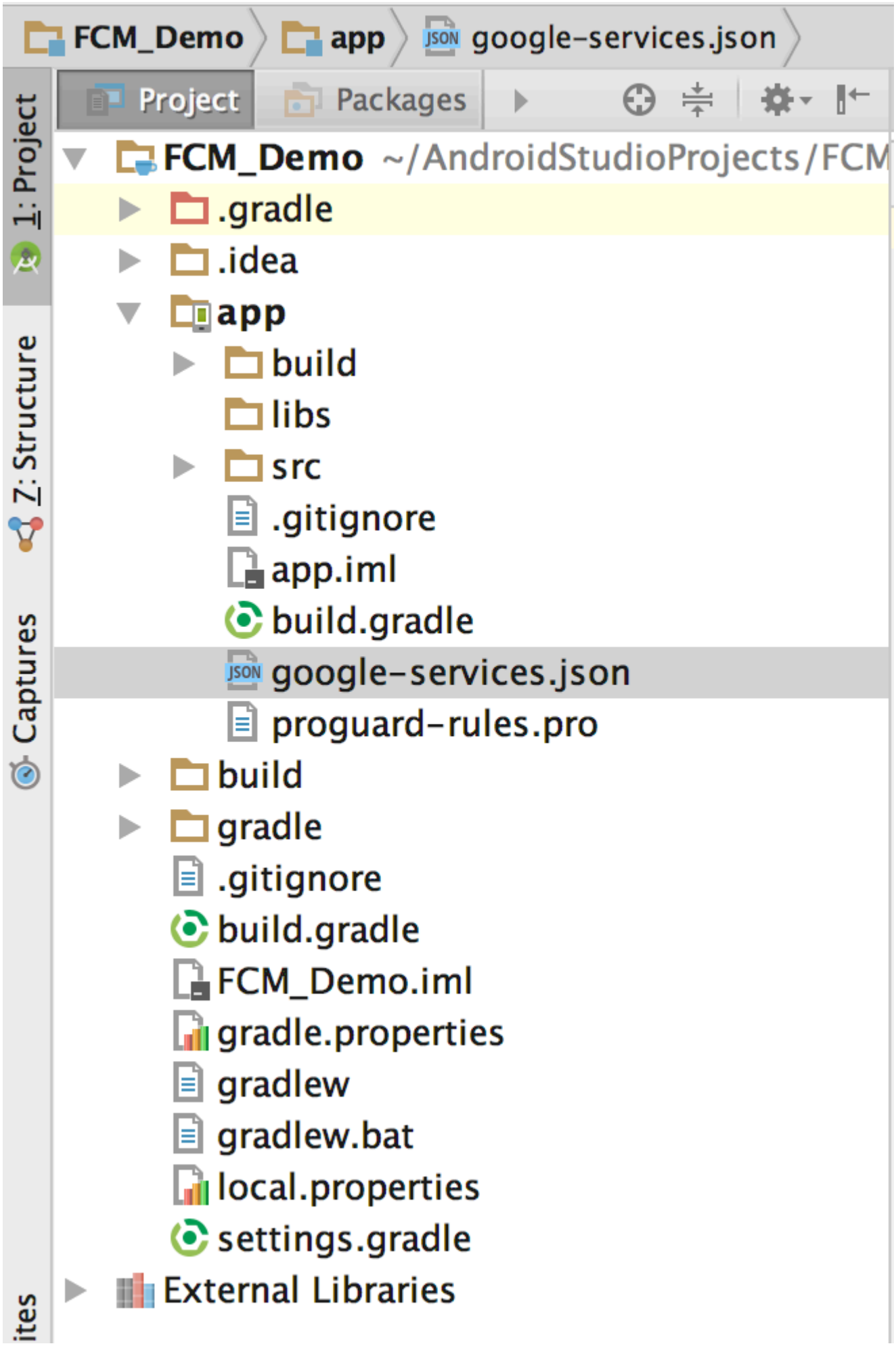
   b。デバッグSHA-1 - でのコマンドをするとつかります。

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -
keypass android
```

Create a project
Project name

このをfirebaseコンソールにし、appをfirebaseプロジェクトにします。アプリケーションのボタンをクリックすると、ブラウザはに "google-services.json"というのJSONファイルをダウンロードします。

4. ダウンロードしたgoogle-services.jsonファイルをAndroidアプリモジュールのルートディレクトリにコピーします。


FCM_Demo
app
google-services.json
Project
Packages
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
1: Project
7: Structure
Captures

5. にんでいくうちに、Firebaseコンソーールにされるにってください。 a。プロジェクトレベルbuild.gradleにのコーードをしてください

```
dependencies{ classpath 'com.google.gms:google-services:3.1.0' .....
```

b。アプリレベルbuild.gradleのにのコーードをします。

```
    //following are the dependencies to be added
    compile 'com.google.firebase:firebase-messaging:11.0.4'
    compile 'com.android.support:multidex:1.0.1'
}
// this line goes to the end of the file
apply plugin: 'com.google.gms.google-services'
```

c。 Androidのスタジオは、プロジェクトをするようにんでいます。すぐをクリックします。

6. のタスクは、2つのサーービスをすることです。 a。 intent-filterをしてFirebaseMessagingServiceをのようにするもの

```
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
```

b。 1つはFirebaseInstanceIDServiceをします。

```
<intent-filter>
    <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
</intent-filter>
```

7. FirebaseMessagingServiceコーードはのようになります。

```
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はのようになります。

```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

import com.google.firebase.iid.FirebaseInstanceIdService;

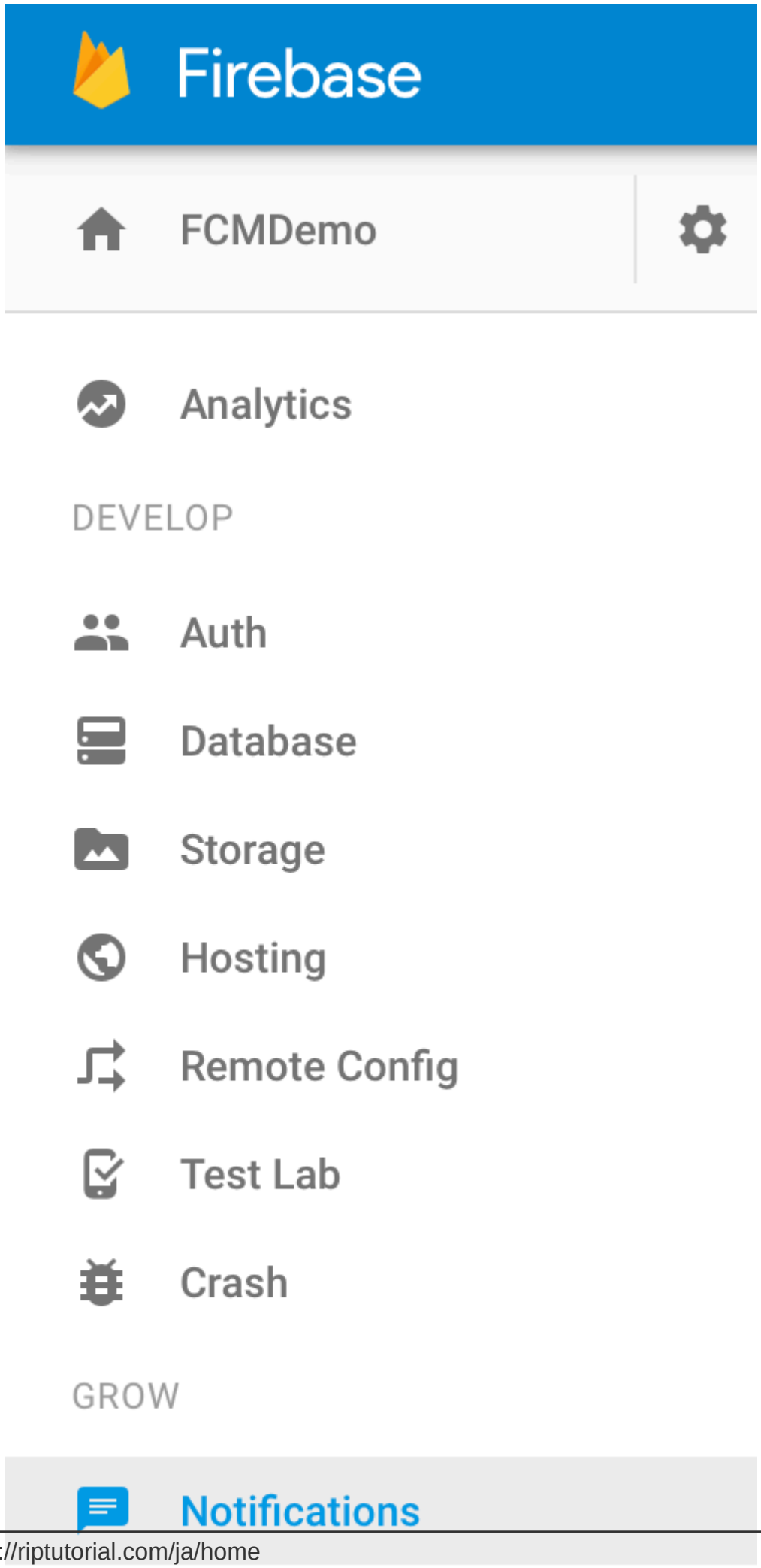
public class MyFirebaseInstanceIDService extends FirebaseInstanceIdService {
```

```
    public MyFirebaseInstanceIDService() {
    }
}
```

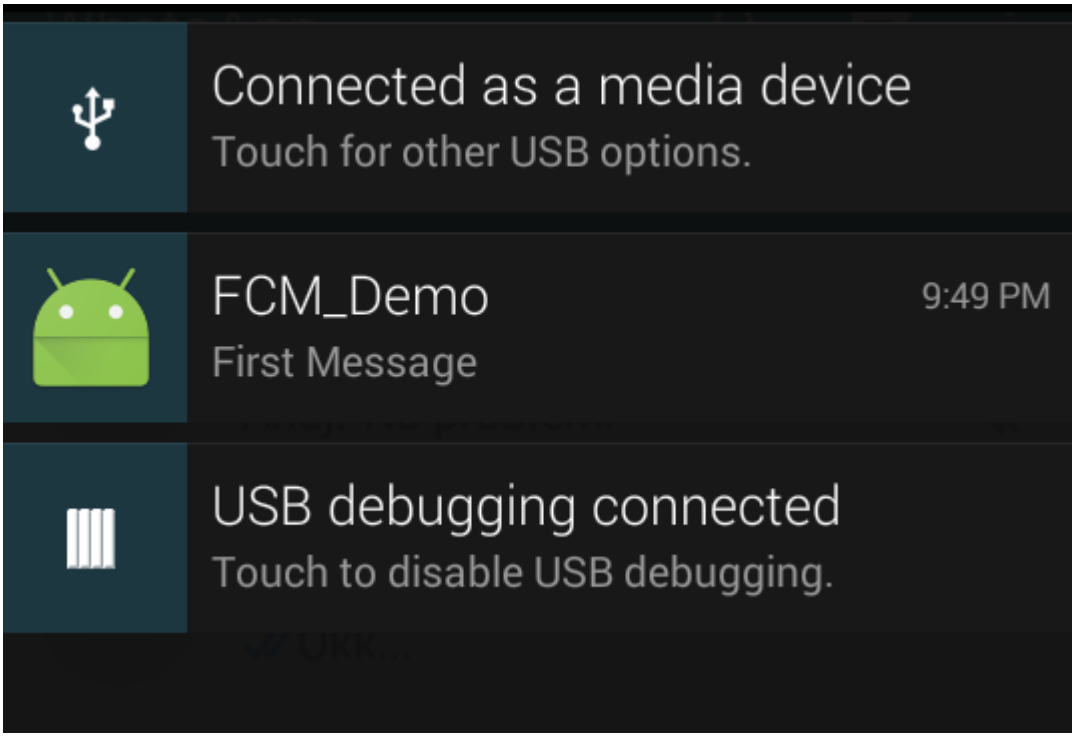
9. は、デバイストークンをします。 MainActivityのonCreateメソッドにのコードをします。

```
String token = FirebaseInstanceId.getInstance().getToken();
Log.d("FCMAPP", "Token is "+token);
```

10. アクセストークンをしたら、firebase consoleをしてをすることができます。 Androidでアプリをします。


Firebase
FCMDemo
Analytics
DEVELOP
Auth
Database
Storage
Hosting
Remote Config
Test Lab
Crash
GROW
Notifications

たちがしたデバイストークンIDをしてのデバイスにメッセージをする、またはアプリケーションをするすべてのユーザー、またはのユーザーグループにメッセージをするをします。のメッセージをすると、モバイルはのようにされます。



ありがとうございました

Firebaseログアウト

の

```
private GoogleApiClient mGoogleApiClient;
```

サインアウトボタンをすと、このコードをonCreateメソッドでくがあります。

```
 mGoogleApiClient = new GoogleApiClient.Builder(this)
            .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener
*/)
            .addApi(Auth.GOOGLE_SIGN_IN_API)
            .build();
```

サインアウトボタンのにコードをしてください。

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

オンラインでファイアベースをむ https://riptutorial.com/ja/android/topic/3843/ファイアベース

210: ファストジョン

き

Fastjsonは、JavaオブジェクトをJSONにするためにできるJavaライブラリです。これは、JSONをのJavaオブジェクトにするためにもできます。

Fastjsonの

サーバーとアンドロイドクライアントでのパフォーマンスを

な`toJSONString()`および`parseObject()`メソッドをして、JavaオブジェクトをJSONにしたり、も

のなオブジェクトをJSONとのでできるようにする

Java Genericsのなサポート

- オブジェクトテキスト
- JSONObject parseObjectテキスト
- T parseObjectString text、Class <T> clazz
- JSONArray parseArrayテキスト
- <T> List <T> parseArrayString text、Class <T> clazz
- String toJSONStringObjectオブジェクト
- String toJSONStringObject object、boolean prettyFormat
- Object toJSONオブジェクトjavaObject

Examples

FastjsonをしたJSONの

Fastjsonライブラリのサンプルをることができます

エンコード

```
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);
```

```
{"id":0,"name":"admin","users":[{"id":2,"name":"guest"},{"id":3,"name":"root"}]}
```

デコード

```
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;
    }
}
```

タイプマップのデータをJSON Stringにする

コード

```
    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);
```

```
{1:"No.1",2:"No.2",3:[{"id":2,"name":"Liu"},{"id":3,"name":"Yue"}]}
```

オンラインでファストジョンをむ https://riptutorial.com/ja/android/topic/10865/ファストジョン

211: フレスコ

き

フレスコは、Androidアプリケーションでをするためのなシステムです。

Android 4.xでは、Frescoは**Android**メモリ ashmemとばれるのなにをします。これにより、アプリケーションのがされ、ろしいOutOfMemoryErrorのがされます。

FrescoはJPEGのストリーミングもサポートしています。

アプリレベルbuild.gradleファイルのをするには

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

はこちらをください 。

Examples

フレスコ

に、「」セクションにすように、Frescoを`build.gradle`にします。

アニメーションGIFやWebPサポートなどのがなは、するFrescoアーティファクトもするがあります。

フレスコをするがあります。これを1だけうがあるため、 `Application`をすることをおめします。これのはのとおりです。

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

サーバーからリモートイメージをロードするは、アプリケーションにinterntアクセスがです。に`AndroidManifest.xml`してください

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

に、XMLレイアウトに`SimpleDraweeView`をします。フレスコは、がなるのプレースホルダ、エラー

、のなどがあるがあるため、の`wrap_content`をサポートしていません。

したがって、ディメンションまたは`match_parent` をつ`SimpleDraweeView`をすることもできます

```
<com.facebook.drawee.view.SimpleDraweeView
    android:id="@+id/my_image_view"
    android:layout_width="120dp"
    android:layout_height="120dp"
    fresco:placeholderImage="@drawable/placeholder" />
```

または、のアスペクトをします。

```
<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" />
```

に、イメージURIをJavaですることができます

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

それでおしまいネットワークイメージがされるまで、プレースホルダードロアブルがされます。

フレスコとOkHttp 3をう

まず、のフレスコGradleのにえて、あなたはあなたにOkHttp 3をするがあり`build.gradle`

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

Frescoをするときはカスタム`Application`で、OkHttpクライアントをできるようになりました

```
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);
```

DraweeControllerをしたJPEGによるフレスコのストリーミング

このでは、すでにアプリケーションにFrescoをしていることをとしています このを。

```
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.
```

オンラインでフレスコをむ https://riptutorial.com/ja/android/topic/5217/フレスコ

212: プログレスバー

ドキュメント ProgressBar

Examples

なProgressBar

なProgressBarは、のがないアニメーションをします。

になProgressBarスピンホイール

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

ProgressBarフラットバー

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

そののみみのProgressBarスタイル

```
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"
```

アクティビティでなProgressBarをするには

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setVisibility(View.VISIBLE);
progressBar.setVisibility(View.GONE);
```

ProgressBarをする

ProgressBarは、ののをのにかってします。

のProgressBar

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="10dp"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

の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"/>
```

res / drawable / progress_vertical.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/background">
        <shape>
            <corners android:radius="3dp"/>
            <solid android:color="@android:color/darker_gray"/>
        </shape>
    </item>
    <item android:id="@android:id/secondaryProgress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
                <corners android:radius="3dp"/>
                <solid android:color="@android:color/holo_blue_light"/>
            </shape>
        </clip>
    </item>
    <item android:id="@android:id/progress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
                <corners android:radius="3dp"/>
                <solid android:color="@android:color/holo_blue_dark"/>
            </shape>
        </clip>
    </item>
</layer-list>
```

リングのProgressBar

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:progressDrawable="@drawable/progress_ring"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

res / drawable / progress_ring.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/secondaryProgress">
        <shape
            android:shape="ring"
            android:useLevel="true"
            android:thicknessRatio="24"
            android:innerRadiusRatio="2.2">
            <corners android:radius="3dp"/>
            <solid android:color="#0000FF"/>
        </shape>
    </item>

    <item android:id="@android:id/progress">
        <shape
            android:shape="ring"
            android:useLevel="true"
            android:thicknessRatio="24"
            android:innerRadiusRatio="2.2">
            <corners android:radius="3dp"/>
            <solid android:color="#FFFFFF"/>
        </shape>
    </item>
</layer-list>
```

アクティビティでProgressBarをするには

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setSecondaryProgress(100);
progressBar.setProgress(10);
progressBar.setMax(100);
```

カスタマイズされたプログレスバー

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>
```

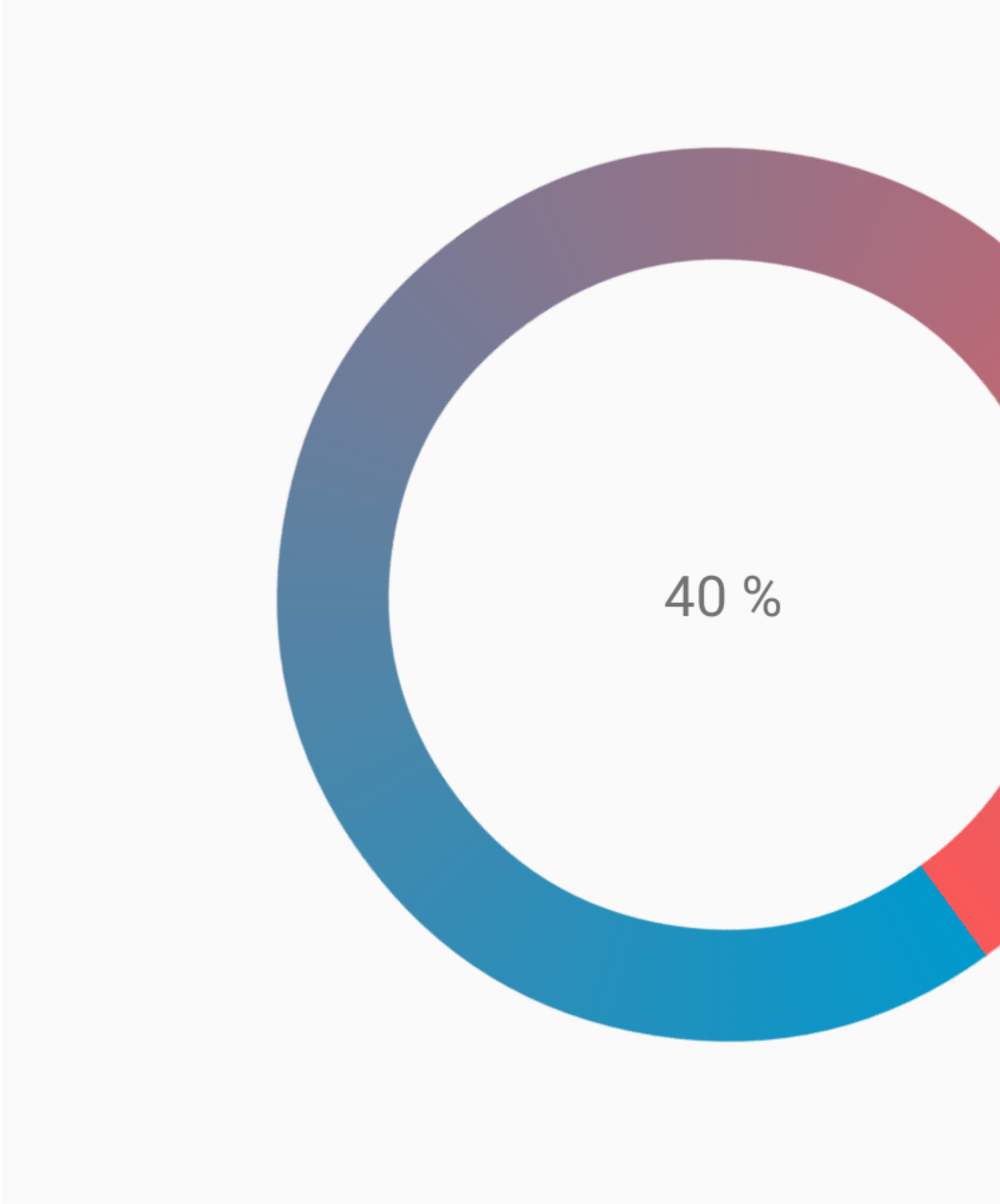
custom_progressbar_drawable.xml

```
<?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>
```

リファレンススクリーンショット



Tinting ProgressBar

AppCompatテーマをすると、 `ProgressBar`のはあなたがした`colorAccent`ます。

5.0

アクセントのをせずに`ProgressBar`をするには、アクセントのをオーバーライドする`android:theme`をします。

```
<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>
```

`ProgressBar`にをけるには、XMLファイルで`android:indeterminateTintMode`と`android:indeterminateTint`できます

```
<ProgressBar
    android:indeterminateTintMode="src_in"
    android:indeterminateTint="@color/my_color"
/>
```

マテリアルリニアプログレスバー

マテリアルドキュメントによると

> インジケータはに0から100までをたし、してがしません。
> ヘッダーやシートのにあるバーがされ、えてされます。

Linear ProgressBarをするには、xmlでするだけです。

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```



な ProgressBarをするには、 `android:indeterminate`を`true`し`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"/>
```

する

ProgressBarをするには、 `android:indeterminate`を`false`し、 `android:max`および`android:progress`をします。

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"/>
```

このコードをしてをしてください

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
```

バッファ

ProgressBarでバッファをするには、 `android:indeterminate`を`false`し、 `android:max`、 `android:progress`、 `android:secondaryProgress`をします。

```
<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"/>
```

バッファは、 `android:secondaryProgress`でされます。
このコードをしてをするだけです

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
progressBar.setSecondaryProgress(50);
```

でする

こののProgressBarをするには、 `android:indeterminate`をtrueにしてなProgressBarをします。

```
<ProgressBar
    android:id="@+id/progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="true"/>
```

に、からをするためにりえるがあるは、 `setIndeterminate()`メソッドをします。

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setIndeterminate(false);
```

カスタムダイアログの

カスタムプログレスダイアログをすることで、ダイアログをせずにUIインスタンスですることができます。

に、カスタムダイアログをします。

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;
       }
   }
}
```

カスタムのレイアウトをする

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>
```

これです。 Dialog in Codeをびしています

```
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();
```

オンラインでプログレスバーをむ https://riptutorial.com/ja/android/topic/3353/プログレスバー

213: プロジェクトSDKのバージョン

き

Androidアプリケーションは、あらゆるのデバイスでするがあります。それぞれのデバイスはAndroidでなるバージョンをしているがあります。

さて、Androidバージョンは、あなたのアプリケーションがとするすべてのをサポートしていないがあります。したがって、アプリケーションをするには、Androidバージョンのとをにいてください。

パラメーター

パラメータ	
SDKのバージョン	フィールドのSDKバージョンは、Androidリリースの SDK APIレベルです。たとえば、FroyoAndroid 2.2はAPIレベル8にしています。これらのも `Build.VERSION_CODES` されてい `Build.VERSION_CODES` 。

プロジェクトには、する4つのSDKバージョンがあります。

- `targetSdkVersion` はあなたがテストしたAndroidのバージョンです。

  フレームワークは `targetSdkVersion` をして、のをいつにするかをします。たとえば、APIレベル23をターゲティングすると、ランタイムモデルがされます。

- `minSdkVersion` は、アプリケーションがサポートするAndroidのバージョンです。このバージョンよりいバージョンのAndroidをしているユーザーは、アプリケーションをインストールしたり、Playストアでることはできません。

- `maxSdkVersion` は、アプリケーションがサポートするAndroidのバージョンです。このバージョンよりしいバージョンのAndroidをしているユーザーは、アプリケーションをインストールすることも、Playストアでることもできません。これは、ほとんどのアプリケーションがしいバージョンのAndroidでうまくかないためにされるべきではありません。

- `compileSdkVersion` は、アプリケーションがコンパイルされるAndroid SDKのバージョンです。にされているAndroidのバージョンであるがあります。これは、コードをくときにアクセスできるAPIをします。 `compileSdkVersion` が22にされているは、APIレベル23でされたメソッドをびすことはできません。

Examples

プロジェクトSDKバージョンの

メインモジュール **app** の`build.gradle`ファイルに、バージョンとターゲットバージョンをします。

```
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
    }
}
```

オンラインでプロジェクトSDKのバージョンをむ https://riptutorial.com/ja/android/topic/162/プロジェクトsdkのバージョン

214: ペイント

き

ペイントは、キャンバスびしをする、ビットマップピクセルをする、プリミティブRect、Path、Bitmap ...とに、にな4つのオブジェクトの1つです。

Examples

ペイントの

の3つのコンストラクタのいずれかをしてしいペイントをできます。

- `new Paint()`デフォルトで
- `new Paint(int flags)`
- `new Paint(Paint from)`のペイントからをコピーする

パフォーマンスのをきこすがあるので、paintオブジェクトやonDrawののオブジェクトをしてしないことをおめします。 Android Studioはおそらくあなたにしますわりに、グローバルにして、クラスコンストラクタでのようにします。

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

テキストのペイントの

テキスト

- `setTypeface(Typeface typeface)`フォントフェイスをします。 をしてください
- `setTextSize(int size)`フォントサイズをピクセルでします。
- `setColor(int color)`テキストのをめ、りのをします。 `setARGB(int a, int r, int g, int b`、

`setAlpha(int alpha)`することもできます。
- `setLetterSpacing(float size)`のをemでします。デフォルトは0で、のはテキストをきめ、のはテキストをきめます。
- `setTextAlign(Paint.Align align)`をにしてテキストのをします。 `Paint.Align.LEFT`はの`RIGHT`にし、 `RIGHT`はにし、 `CENTER`はのにします
- `setTextSkewX(float skewX)`これは、のとなすことができます。 SkewXは、テキストのボトムのオフセットをします。 は-0.25をしてください
- `setStyle(Paint.Style style)`テキストを`FILL` 、ストロークテキストを`STROKE` 、またはを`FILL_AND_STROKE`

SPまたはDPからピクセルにするには`TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, size, getResources().getDisplayMetrics())`をできます。

テキストの

- `float width = paint.measureText(String text)`のをする
- `float height = paint.ascent()`テキストのさをする
- `paint.getTextBounds(String text, int start, int end, Rect bounds`テキストのを`paint.getTextBounds(String text, int start, int end, Rect bounds`をりてる、nullにすることはできません。

```
String text = "Hello world!";
Rect bounds = new Rect();
paint.getTextBounds(text, 0, text.length(), bounds);
```

のもありますが、これらの3つはほとんどのにします。

シェイプをするためのペイントの

- `setStyle(Paint.Style style)`りつぶし`FILL` 、ストロークシェイプ`STROKE` 、または`FILL_AND_STROKE`
- `setColor(int color)`ペイントのをします。 `setARGB(int a, int r, int g, int b`、`setAlpha(int alpha)`することもできます。
- `setStrokeCap(Paint.Cap cap)` `ROUND` 、 `SQUARE` 、または`BUTT`いずれかのラインキャップをします none これをしてください。
- `setStrokeJoin(Paint.Join join)`を`MITER` `BEVEL` 、 `ROUND` 、または`BEVEL`いずれかにします。 これをてください。
- `setStrokeMiter(float miter)`マイターのをします。これにより、マイターがにするのをぎ、xピクセルにベベルにすることができます。 これをてください。
- `setStrokeWidth(float width)`ストロークのをします。 `0`は、キャンバスマトリックスとはしたヘアラインモードでします。 に1ピクセル

フラグの

コンストラクタで、または`setFlags(int flags)`を`setFlags(int flags)`して、のフラグをできます。

- `Paint.ANTI_ALIAS_FLAG`アンチエイリアスをにし、をらかにします。
- `Paint.DITHER_FLAG`ディザリングをにします。がデバイスのよりもいは、 これがしますす。
- `Paint.EMBEDDED_BITMAP_TEXT_FLAG`ビットマップフォントのをにします。
- `Paint.FAKE_BOLD_TEXT_FLAG`は、のフェイスをするわりに、のいエフェクトをしてテキストをします。いくつかのフォントはになっており、 のではされません
- `Paint.FILTER_BITMAP_FLAG`のビットマップのサンプリングにします。
- `Paint.HINTING_OFF` 、 `Paint.HINTING_ON`りえフォントヒンティングは、、これを
- `Paint.LINEAR_TEXT_FLAG`フォントのスケーリングをにし、をスケーリングする
- `Paint.SUBPIXEL_TEXT_FLAG`テキストは、サブピクセルのでされます。
- `Paint.STRIKE_THRU_TEXT_FLAG`されたテキストがストライクされます。
- `Paint.UNDERLINE_TEXT_FLAG`されるテキストにがかれます

フラグをしてのようなフラグをすることができます

```
Paint paint = new Paint();
paint.setFlags(paint.getFlags() | Paint.FLAG);   // Add flag
paint.setFlags(paint.getFlags() & ~Paint.FLAG);  // Remove flag
```

そこにないフラグをしようとしているか、すでにあるフラグをしてももされません。また、ほとんどのフラグは`set<Flag>(boolean enabled)`をして`set<Flag>(boolean enabled)`することもできます`set<Flag>(boolean enabled) setAntialias(true)`。

`paint.reset()`をして、ペイントをデフォルトにリセットすることができます。のデフォルトフラグは`EMBEDDED_BITMAP_TEXT_FLAG`です。 `new Paint(0)`をしてもされますが、

オンラインでペイントをむ https://riptutorial.com/ja/android/topic/9141/ペイント

215: ベクトルファイル

き

このがすように、ベクトルはベクトルグラフィックにづいています。ベクトルグラフィックスは、をしてグラフィックをするです。これにより、XMLベクトルグラフィックにづいてドロウアブルをできます。では、mdpi、hdpi、xhdpiなどのためになるサイズのイメージをするはありません。Vector Drawableをすると、イメージをxmlファイルとして1だけし、すべてのdpiとなるデバイスにしてイメージをするがあります。これもスペースをするものではなく、メンテナンスもです。

パラメーター

パラメータ	
`<vector>`	ベクトルドロアブルをするためにされます。
`<group>`	パスまたはサブグループのグループとをします。は、ビューポートとじでされます。そして、はスケールのにされ、してされます。
`<path>`	されるパスをします。
`<clip-path>`	パスをのクリップにします。クリップパスはのグループとそのにのみされます。

build.gradleファイルをします。

```
dependencies {
    ...
   compile 'com.android.support:appcompat-v7:23.2.1'
}
```

Gradleプラグインの **v2.0**をしているは、のコードをしてください。

```
// Gradle Plugin 2.0+
 android {
   defaultConfig {
     vectorDrawables.useSupportLibrary = true
    }
 }
```

Gradleプラグインの **v1.5**をしているは、のコードをしてください。

```
// Gradle Plugin 1.5
 android {
   defaultConfig {
     generatedDensities = []
  }

  // This is handled for you by the 2.0+ Gradle Plugin
  aaptOptions {
    additionalParameters "--no-version-vectors"
  }
 }
```

しくは、 Androidサポートライブラリ23.2リリースノートをおみください。

*AppCompat*をしていても 、Vector DrawablesはいAndroidバージョンではアプリのでしません 。たとえば、アプリケーションではなくシステムによってされるため、ベクターオブジェクトをアイコンとしてすことはできません。については、 このをしてください。

Examples

VectorDrawableの

AppCompatでにしているベクトルのをにします。

res / drawable / ic_search.xml

```
<vector xmlns:android="..."
        android:width="24dp"
        android:height="24dp"
        android:viewportWidth="24.0"
        android:viewportHeight="24.0"
        android:tint="?attr/colorControlNormal">

    <path
        android:pathData="..."
        android:fillColor="@android:color/white"/>

</vector>
```

このDrawableをすると、 `ImageView`のはのようになります。

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@drawable/ic_search"/>
```

にすることもできます

```
ImageView iv = (ImageView) findViewById(...);
iv.setImageResource(R.drawable.ic_search);
```

じとびしが`ImageButton`でも`ImageButton`ます。

VectorDrawable xmlの

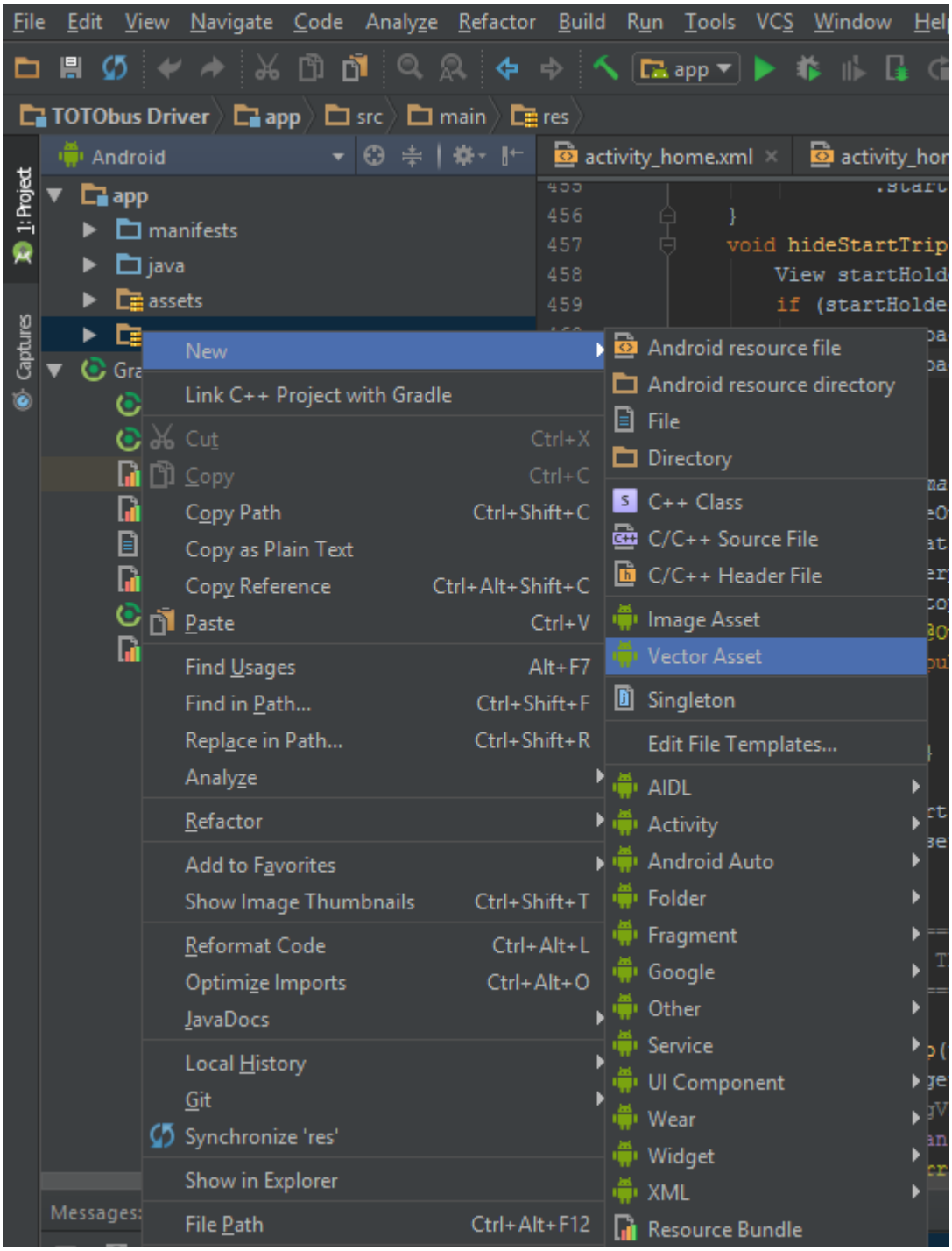
に、この**vectordrawable.xml**ファイルのな`VectorDrawable`をします。

```
<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>
```

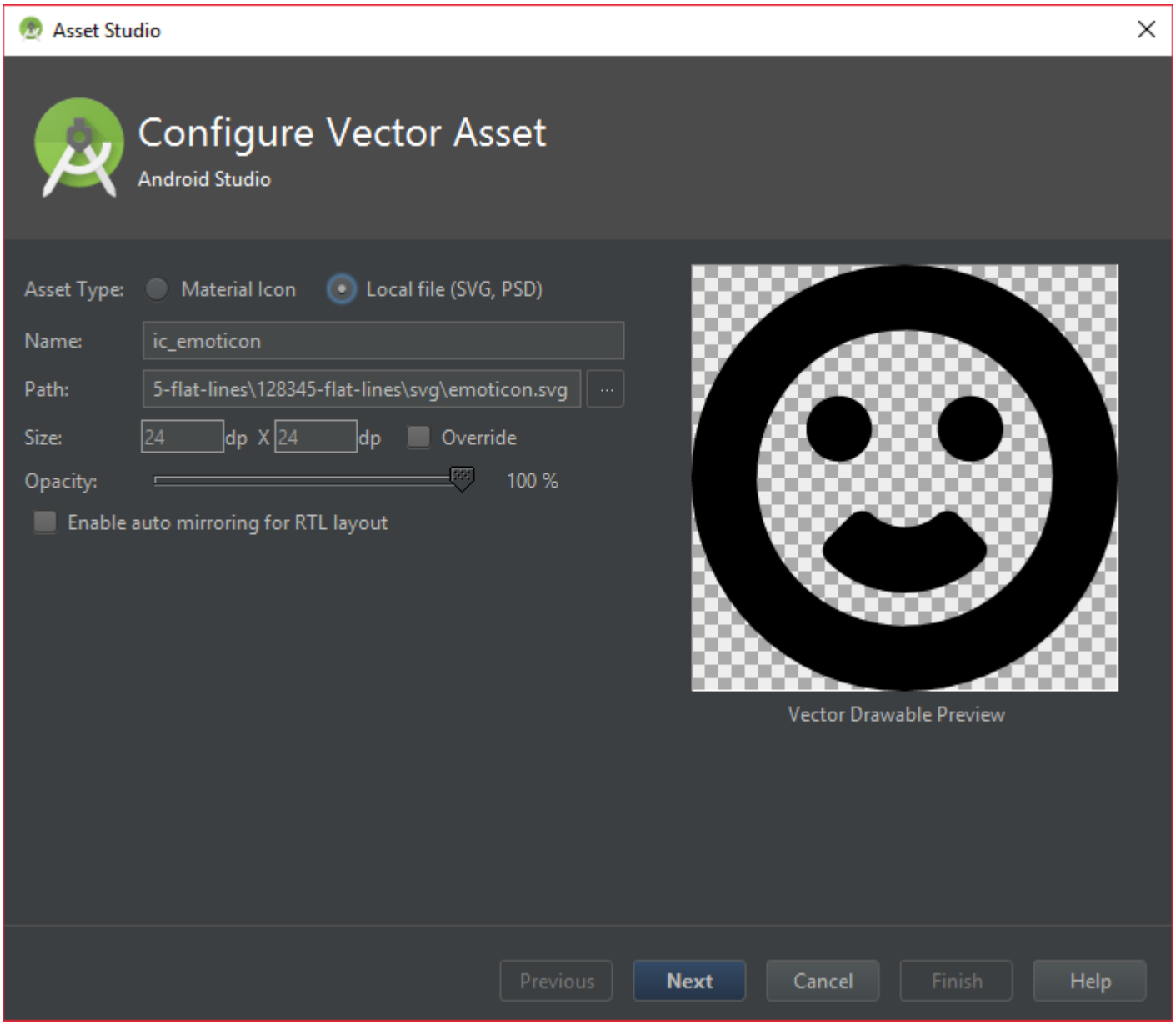
VectorDrawableとしてのSVGファイルのみ

Android Studioで*SVG*ファイルを`VectorDrawable`としてみむには、のをします。

`res`フォルダをクリックし、 > ベクター をします。



[ローカルファイル]オプションをし、.svgファイルをします。あなたのみにオプションをし、のヒット。しました。



オンラインでベクトルファイルをむ https://riptutorial.com/ja/android/topic/8194/ベクトルファイル

216: ベトナムをのにする

Examples

```
String myStr = convert("Lê Minh Thoại là người Việt Nam");
```

```
"Le Minh Thoai la nguoi Viet Nam"
```

ChuyểnchuỗiTiếngthunhchuỗikhôngdấu

```
  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;
}
```

オンラインでベトナムをのにするをむ https://riptutorial.com/ja/android/topic/10946/ベトナムをのにする

217: ボタン

- <ボタン... />
- androidonClick = "methodname"
- button.setOnClickListenerしいOnClickListener{...};
- パブリッククラスclassnameはView.OnLongClickListenerをします

Examples

インラインonClickListener

ボタンがあるとしますプログラムですることも、findViewbyIdなどをしてビュ―からバインドすることもできます

```
Button btnOK = (...)
```

は、クラスをし、インラインにします。

```
btnOk.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Do stuff here...
    }
});
```

レイアウトをしてクリックアクションをする

レイアウトでボタンをするときは、 `android:onClick`をして、コードのメソッドをしてクリックをできます。

ボタン

```
<Button
    android:width="120dp"
    android:height="wrap_content"
    android:text="Click me"
    android:onClick="handleClick" />
```

に、あなたのアクティビティで、 `handleClick`メソッドをします。

```
public void handleClick(View v) {
    // Do whatever.
}
```

XMLの1つのビュ―にしてじクリックイベントをする

レイアウトでViewをするときは、androidonClickをして、するアクティビティまたはフラグメントのメソッドをして、クリックイベントをできます。

XMLレイアウト

```
<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" />
```

アクティビティ/フラグメントコード

コードで、をけたメソッドをします。ここでvはタッチされたビューになり、このメソッドをびすビューごとにかをいます。

```
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;
    }
}
```

にじて、ビューごとになるメソッドをすることもできますこの、IDをするはありません。

いクリックイベントをく

いクリックをまえてするには、ボタンになリスナーをするがあります。

```
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);
```

リスナーの

それをいつうべきですか

- インラインリスナーのコードがきすぎてメソッド/クラスがくみにくい
- あなたは、あなたのアプリのさまざまなビューでじアクションをしたい

これをするには、 View APIでリスナーの1つをするクラスをするがあります 。

たとえば、のをくクリックするとヘルプがされます。

```
public class HelpLongClickListener implements View.OnLongClickListener
{
    public HelpLongClickListener() {
    }

    @Override
    public void onLongClick(View v) {
        // show help toast or popup
    }
}
```

に、それをするには、 `Activity`にまたはがです。

```
HelpLongClickListener helpListener = new HelpLongClickListener(...);

button1.setOnClickListener(helpListener);
button2.setOnClickListener(helpListener);
label.setOnClickListener(helpListener);
button1.setOnClickListener(helpListener);
```

されたクラスのリスナーをすることにはが1つあります。クラスフィールドにアクセスすることはできないため、をしたりゲッターをしないり、コンストラクタをしてデータコンテキスト、ビューをすがあります。

カスタムクリックリスナーをしてのクリックを

ボタンがにされないように 1に2クリックすると、フローがされないとながするがあります、カスタム***SingleClickListener***をできます。

このClickListenerは、のをしきいたとえば、1000msとしてします。
ボタンをクリックすると、したのにトリガーがされたかどうかをするチェックがされ、トリガーされていないはトリガーされます。

```
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);

}
```

クラスでは、SingleClickListenerはボタンにけられています

```
myButton = (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(new SingleClickListener() {
    @Override
    public void performClick(View view) {
        // do stuff
    }
});
```

ボタンスタイルのカスタマイズ

ボタンのをカスタマイズするはたくさんあります。このでは、いくつかのオプションをします。

オプション**0ThemeOverlay**をするもなです

スタイルファイルにしいスタイルをする

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>
```

に、ボタンをするレイアウトMainActivityなど

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>
```

オプション**1**のボタンスタイルをする

values / styles.xmlで、ボタンのしいスタイルをします。

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>
```

に、ボタンをするレイアウトMainActivityなど

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>
```

オプション**2**ボタンのごとにドロアブルをりてる

「mybuttondrawable.xml」というドロウアブル･フォルダーにxmlファイルをして、ボタンのドロウアブル･リソースをします。

drawable / mybutton.xml

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:state_enabled="false"
        android:drawable="@drawable/mybutton_disabled" />
    <item
        android:state_pressed="true"
        android:state_enabled="true"
        android:drawable="@drawable/mybutton_pressed" />
    <item
        android:state_focused="true"
        android:state_enabled="true"
        android:drawable="@drawable/mybutton_focused" />
    <item
        android:state_enabled="true"
        android:drawable="@drawable/mybutton_enabled" />
</selector>
```

これらのドロウアブルのそれぞれは、あなたがし、ドロウアブルフォルダにされたイメージえば、mybutton_disabled.pngまたはxmlファイルです。えば

drawable / mybutton_disabled.xml

```
<?xml version="1.0" encoding="utf-8"?>

    <shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle">
        <gradient
            android:startColor="#F2F2F2"
            android:centerColor="#A4A4A4"
            android:endColor="#F2F2F2"
            android:angle="90"/>
        <padding android:left="7dp"
            android:top="7dp"
            android:right="7dp"
            android:bottom="7dp" />
        <stroke
            android:width="2dip"
            android:color="#FFFFFF" />
```

```
        <corners android:radius= "8dp" />
    </shape>
```

に、ボタンをするレイアウトMainActivityなど

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>
```

オプション**3**ボタンスタイルをアプリケーションテーマにする

あなたのアプリケーションテーマのvalues / styles.xmlでデフォルトのアンドロイドボタンスタイルをきすることができます。

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>
```

オプション4プログラムでデフォルトのボタンスタイルにをねる

アクティビティのボタンをつけ、カラーフィルタをするだけです

```
Button mybutton = (Button) findViewById(R.id.mybutton);
mybutton.getBackground().setColorFilter(anycolor, PorterDuff.Mode.MULTIPLY)
```

ここではさまざまなブレンドモードをできます 。

オンラインでボタンをむ https://riptutorial.com/ja/android/topic/5607/ボタン

218: ボトムシート

き

ボトムシートは、のからにスライドするシートです。

のシートは、のからスライドして、よりくのコンテンツをします。
それらはv23.2.0バージョンのAndroidサポートライブラリにされました。

Examples

GoogleマップのようなBottomSheetBehavior

2.1.x

このは、サポートライブラリ23.4.0。+にしています。

BottomSheetBehaviorのはのとおりです。

1. ボトムシートのきにするアニメーションをえた2つのツールバー。
2. 「モーダルツールバー」にづいたときにれるFABにスライドしてされるもの。
3. ボトムシートのにあるで、らかのがあります。
4. ツールバーのタイトルTextViewは、ボトムシートにするとされます。
5. のステータスバーは、をまたはフルカラーにすることができます。
6. 「アンカー」のカスタムボトムシート。

さあ、ひとつずつチェックしてみましょう

ツールバー
Google Mapsでそのビューをくと、できるにツールバーがされます。これは、がGoogle Mapsとまったくじようにしていないのツールバーですとにかく、 `ToolBar`が`AppBarLayout`にあり、`AppBarLayout`ドラッグをするとになり、BottomSheetが`COLLAPSED`になったときにびされます。
それをするには、のことがです。

- `Behavior`をし、それを`AppBarLayout.ScrollingViewBehavior`から`AppBarLayout.ScrollingViewBehavior`
- き`layoutDependsOn`と`onDependentViewChanged`を。あなたはボトムシートのきをきます。
- アニメーションをしてAppBarLayout / ToolBarをにするをいくつかします。

これは、のツールバーまたはActionBarでったです。

```
@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_mediumAnimTime));
    mToolbarAnimation.y(mInitialY).start();
}
```

なはここになファイルがあります

2のツーールバーまたは「モーダル」ツーールバー
じメソッドをオーバーライドするがありますが、ここではよりくのビヘイビアをするがあります
。

- アニメーションでツールバーを/する
- ステータスバーの/をする
- ツールバーのBottomSheetタイトルの/
- ボトムシートをじるか、りたたまれたにする

このコードはしなので、はリンクをします

FAB

これもカスタムビヘイビアですが、 `FloatingActionButton.Behavior`から
`FloatingActionButton.Behavior`ます。で`onDependentViewChanged`あなたはそれがあなたがそれをしたいところで「オフセット」またはポイントにするとています。のは、2のツーールバーのくにあるときににしたいので、FABCoordinatorLayoutをべ、ToolBarをむAppBarLayoutをし、OffSetのような`OffSet`をします

```
@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;
}
```

なカスタムFABリンク

をつ**BottomSheet**のにある
のものと、これはカスタムです。こののの「な」ものは、BottomSheetにされたをし、デフォル

トののようなのをけるさなアルゴリズムです。

```
@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;
}
```

をつのなファイル

はに **Custom BottomSheet Behavior**
3つのステップをするには、まずデフォルトのBottomSheetBehaviorに`STATE_DRAGGING`, `STATE_SETTLING`, `STATE_EXPANDED`, `STATE_COLLAPSED`, `STATE_HIDDEN` 5つのがあり、Googleマップのではりたたみとののをするがあることをしておくがあります `STATE_ANCHOR_POINT`
は、デフォルトのbottomSheetBehaviorをさせずにしようとしたので、すべてのコードをりけてコピーし、なものをしました。
がしていることをするには、のステップにってください

1. Javaクラスをし、 `CoordinatorLayout.Behavior<V>`からします`CoordinatorLayout.Behavior<V>`
2. ペーストコードをデフォルトの`BottomSheetBehavior`ファイルからしいものにコピーします。
3. のコードを`clampViewPositionVertical`して、 `clampViewPositionVertical`メソッドをします。

```
@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. しいをする

   パブリックstatic final int STATE_ANCHOR_POINT = X;

5. のメソッドをします `onLayoutChild`、 `onStopNestedScroll`、 `BottomSheetBehavior<V> from(V view)`および`setState` オプション

```
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;
}

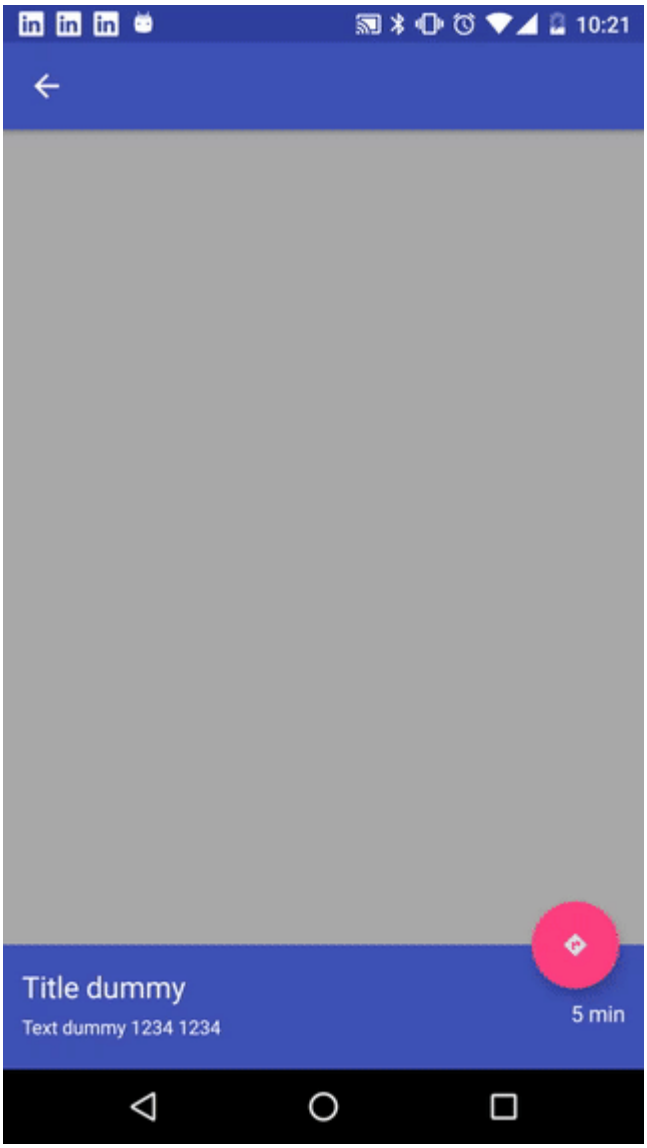
public final void setState(@State int state) {
    if (state == mState) {
        return;
    }
    if (mViewRef == null) {
        // The view is not laid out yet; modify mState and let onLayoutChild handle it later
        /**
         * New behavior (added: state == STATE_ANCHOR_POINT ||)
         */
        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;
}
```

すべてのカスタムビヘイビアをできるプロジェクトにリンクする

そして、それはそれがどのようにえるかです
[



]

クイックセットアップ

のでアプリケーションのbuild.gradleファイルにのがされていることをしてください。

```
compile 'com.android.support:design:25.3.1'
```

に、のオプションをしてボトムシートをできます。

- `CoordinatorLayout`でする`BottomSheetBehavior`
- `BottomSheetDialog`はボトムシートのビヘイビアをつダイアログです
- `BottomSheetDialogFragment`は`DialogFragment`であり、ダイアログのわりに`BottomSheetDialog`をします。

なボトムシート

あなたは`CoordinatorLayout`ビューに`BottomSheetBehavior`をけるPersistent Bottom Sheetをることができます

```
<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>
```

に、コードでのようにをすることができます。

```
 // The View with the BottomSheetBehavior
 View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
 BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

setStateメソッドをして、 BottomSheetBehaviorのをできます。

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

のいずれかのをできます。

- `STATE_COLLAPSED` このりたたまれたはデフォルトであり、レイアウトのだけがにってされます。さは、 `app:behavior_peekHeight`デフォルトは0でできます。
- `STATE_EXPANDED` ボトムシートのにされた。ボトムシートがされますさが`CoordinatorLayout`をむ。または`CoordinatorLayout`がりつふされている
- `STATE_HIDDEN` デフォルトではになっており `app:behavior_hideable`でになっています、これをにするとのシートをにスワイプしてのシートをにすことができます

のコールバックをしたいは、 `BottomSheetCallback`をすることができます

```
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
   }
 });
```

BottomSheetDialogFragmentをつモーダルボトムシート

BottomSheetDialogFragmentをしてモーダルボトムシートをすることができます。

BottomSheetDialogFragmentはモーダルボトムシートです。
これは、 DialogFragmentバージョンで、フローティングダイアログのわりにBottomSheetDialogをしてボトムシートをします。

をするだけです

```
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);
    }
}
```

に、のコードをしてフラグメントをします。

```
MyBottomSheetDialogFragment mySheetDialog = new MyBottomSheetDialogFragment();
FragmentManager fm = getSupportFragmentManager();
mySheetDialog.show(fm, "modalSheetDialog");
```

このフラグメントはBottomSheetDialogをしBottomSheetDialog。

BottomSheetDialogをしたモーダルボトムシート

BottomSheetDialogはボトムシートとしてスタイルされたダイアログです

ちょうどする

```
//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();
```

この、BottomSheetビヘイビアをアタッチするはありません。

デフォルトでExpandedモードでBottomSheet DialogFragmentをきます。

BottomSheet DialogFragmentはデフォルトでSTATE_COLLAPSEDきます。これは、のコードテンプレートのけをりて、 STATE_EXPANDEDまでにき、デバイスのをSTATE_EXPANDEDすることができます。

@NonNull @Override publicダイアログonCreateDialogBundle 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;
}
```

ダイアログアニメーションはわずかにちますが、DialogFragmentをフルスクリーンでくはにうまくいきます。

オンラインでボトムシートをむ https://riptutorial.com/ja/android/topic/5702/ボトムシート

219: ボトムナビゲーションビュー

き

Bottom Navigation Viewはしばらくガイドラインにっていましたが、アプリにするのはではありませんでした。

のアプリケーションはのソリューションをしていますが、のはサードパーティのオープンソースライブラリにしてをしています。

デザインサポートライブラリではこのボトムナビゲーションバーがされましたので、どのようにできるかをてみましょう。

アプリケーションのためののナビゲーションバーをします。これは、のボトムナビゲーションのです。

リンク

- ジャバドック

Examples

な

`BottomNavigationView`をするには、のをします。

1. を `build.gradle` する

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

2. レイアウトに `BottomNavigationView` をします

```
<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. ビューをするメニューをします 。

```
<!-- 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. クリックイベントのリスナーをアタッチします。

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

BottomNavigation-Demoでデモコードをチェックアウト

BottomNavigationViewのカスタマイズ

はあなたが***BottomNavigationView***いをっているとしています。

このでは、 **BottomNavigationView**セレクタをするをします。 UIでアイコンやテキストをすることができます。

bottom_navigation_view_selector.xmlようにします。

```
<?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>
```

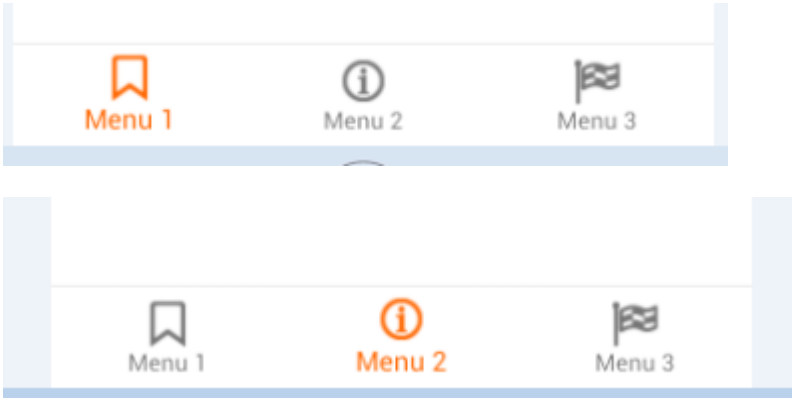
レイアウトファイルの**BottomNavigationView**にのをする

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

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

のでは、`app:itemIconTint`と`app:itemTextColor`じselector `bottom_navigation_view_selector`をして、テキストとアイコンのをじにちました。しかし、デザインにテキストとアイコンのがなる、2つのなるセレクタをしてすることができます。

はのようになります



/の

/メニュ—のセレクタをします。

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" />
```

3つのメニュ—をする

このははにはです。のところ、ShiftModeというをにするはありません。

そのようなをします。

```
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);
    }
}
```

これにより、が3 nosをえたときのメニューのシフトがになります。

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

Proguardの ののproguardファイルもしてください。そうしないとしません。

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

あるいは、クラスをし、そこからこのメソッドにアクセスすることもできます。 のをここに

これはReflectionベースの**HOTFIX**です。Googleのサポートライブラリがびしでされると、これをしてください。

オンラインでボトムナビゲーションビューをむ https://riptutorial.com/ja/android/topic/7565/ボトムナビゲーションビュー

220: ボレー

き

VolleyはAndroidのHTTPライブラリで、Googleがネットワーキングコールをもっとにするためにしたものです。デフォルトでは、すべてのVolleyネットワークコールはでわれ、バックグラウンドスレッドのすべてをし、コールバックをしてフォアグラウンドでをします。ネットワークのデータをすることは、どのアプリでもされるもなタスクの1つで、VolleyライブラリはAndroidアプリのをにするためにされました。

- RequestQueueキュー= Volley.newRequestQueueコンテキスト; //キューをする
- リクエストのリクエスト=しいSomeKindOfRequestClassRequest.Method、String url、Response.Listener、Response.ErrorListener; //らかののリクエストをセットアップします。リクエストのごとになとがされます
- queue.addリクエスト; //をキューにします。がしたまたはらかのでしたは、なリスナーがびされます。

インストール

Googleのなソースコードからをすることができます 。しばらくの、それがのでした。または、サードパーティののいずれかをします。しかし、Googleはついにjcenterののmavenパッケージをリリースしました。

アプリケーションレベルの`build.gradle`ファイルで、これをリストにします。

```
dependencies {
    ...
    compile 'com.android.volley:volley:1.0.0'
}
```

アプリのマニフェストに`INTERNET`アクセスがされていることをします。

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

Googleはこのライブラリについてになをしておらず、もれていません。しかし、なものはのにあります

https://developer.android.com/training/volley/index.html

GitHubにはのがホストされていますが、にこれをホストするのにしたがあるはずです。

https://pablobaxter.github.io/volley-docs/

Examples

GETメソッドをしたStringRequest

```
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);
```

リクエストをキャンセルする

```
// 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);
```

NetworkImageViewにカスタムのデザインタイムをする

Volley `NetworkImageView`がの`ImageView`するいくつかののがあります。ただし、これらのはコードでのみできます。は、XMLレイアウトファイルからをして`NetworkImageView`インスタンスにするクラスをするのです。

`~/res/xml`ディレクトリに、 `attrx.xml`というのファイルをします。

```
<resources>
    <declare-styleable name="MoreNetworkImageView">
        <attr name="defaultImageResId" format="reference"/>
        <attr name="errorImageResId" format="reference"/>
    </declare-styleable>
</resources>
```

しいクラスファイルをプロジェクトにします。

```
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);
        }
    }
}
```

カスタムのをすレイアウトファイルの

```
<?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>
```

JSONをリクエストする

```
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);
```

リクエストにカスタムヘッダーをする[]

あなたのボレーリクエストにカスタムヘッダーをするがあるは、ヘッダーがプライベートにされるため、にこれをうことはできません。

わりに、 `Request.class getHeaders()` メソッドをのようにオーバーライドするがあります。

```
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;
    }
};
```

パラメータの

- `REQUEST_METHOD` - `Request.Method.*`のいずれか。
- `REQUEST_URL` - リクエストをするなURL。
- `REQUEST_BODY` - されるPOST-Bodyをむ`JSONObject` またはnull。
- `RESP_LISTENER` - に`onResponse(T data)`メソッドがびされる`Response.Listener<?>`オブジェクト `Response.Listener<?>`。
- `ERR_LISTENER` - したリクエストにして`onErrorResponse(VolleyError e)`メソッドがびされる `Response.ErrorListener`オブジェクト。

カスタムリクエストをするは、ヘッダーをすることもできます。

```
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;
    }
    ...
}
```

ボレーエラーをするためのヘルパークラス

```
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);
        }
```

POSTメソッドによるStringRequestをしたリモートサーバー

このでは、Androidアプリケーションからするpostリクエストをするためのサーバーがあるとします。

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

HTTPリクエストにVolleyをする

アプリレベルbuild.gradleにgradleをする

```
compile 'com.android.volley:volley:1.0.0'
```

また、アプリのマニフェストにandroid.permission.INTERNETをします。

アプリケーションでVolley RequestQueueインスタンスシングルトンをする

```
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;
}
}
```

これで、getInstanceメソッドをしてvolleyインスタンスをし、
`InitApplication.getInstance().addToQueue(request);`をしてキュ―にしいリクエストをでき
`InitApplication.getInstance().addToQueue(request);`

サ―バ―からJsonObjectをするなはのとおりです

```
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));
```

Volleyのタイムアウトをするには、 `RetryPolicy`をするがあります。ポリシ―は、ネットワ―クなどのでをできないにされます。

Volleyは、リクエストにして`RetryPolicy`をするなをします。デフォルトでは、Volleyはすべてのに

してすべてのソケットとタイムアウトを5にします。 `RetryPolicy`は、タイムアウトがしたときにのをするのロジックをするのあるインター―フェイスです。

コンストラクタには、の3つのパラメ―タがあります。

- `initialTimeoutMs` - のたびに、ソケットタイムアウトをミリでします。
- `maxNumRetries` - がされた。
- `backoffMultiplier` - のたびにソケットにされたをするためにされる。

json request with volleyのサ―バからのブ―ル

あなたはカスタムクラスを1つにすることができます

```
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;
        }
    }
}
```

あなたのでこれをしてください

```
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();
    }
```

リクエストとして**JSONArray**をする

volleyにみまれているデフォルトのリクエストでは、 `POST`リクエストでリクエストとして`JSONArray`をすことはできません。わりに、 `JSON`オブジェクトをパラメータとしてすことしかできません。

ただし、リクエストコンストラクタにパラメータとして`JSON`オブジェクトをすわりに、`Request.class getBody()`メソッドをオーバーライドするがあります。 3のパラメータとして`null`もすがあり`null`。

```
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;
        }
    }
};
```

パラメータの

- `REQUEST_URL` - リクエストをするなURL。
- `RESP_LISTENER` - に`onResponse(T data)`メソッドがびされる`Response.Listener<?>`オブジェクト `Response.Listener<?>`。
- `ERR_LISTENER` - したリクエストにして`onErrorResponse(VolleyError e)`メソッドがびされる `Response.ErrorListener`オブジェクト。

オンラインでボレーをむ https://riptutorial.com/ja/android/topic/2800/ボレー

221: マニフェストファイル

き

Manifestはに "AndroidManifest.xml"というのファイルで、アプリケーションのルートディレクトリにあります。これは、アプリケーション、アイコン、Javaパッケージ、バージョン、アクティビティ、サービス、アプリケーション、そののをします。

Examples

コンポーネント

マニフェストのなタスクは、アプリケーションのコンポーネントについてシステムにすることです。たとえば、マニフェストファイルはのようにアクティビティをできます。

```
<?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>
```

`<application>`では、 `android:icon`は、アプリケーションをするアイコンのリソースをします。

このでは、 `android:name`でアクティビティサブクラスのクラスをし、androidlabelで、アクティビティのユーザーがするラベルとしてするをします。

このようにすべてのappコンポーネントをするがあります

- `<activity>`

- `<service>`

- のための`<receiver>`

- コンテンツプロバイダーの`<provider>`

ソースにはまれていても、マニフェストでしていないアクティビティ、サービス、およびコンテンツプロバイダは、システムからはえないため、してすることはできません。ただし、ブロードキャストレシーバは、マニフェストでするか、コードでにし `BroadcastReceiver`オブジェクトとして、 `registerReceiver()`びしてシステムにすることができます。

アプリケーションのマニフェストファイルをするのについては、AndroidManifest.xmlファイルの

ドキュメントをしてください。

マニフェストファイルのアクセスをする

APIのされたにアクセスするため、またはのアプリケーションとやりとりするためにアプリケーションがとするは、 `AndroidManifest.xml`ファイルでするがあります。これは、 `<uses-permission />`タグを`<uses-permission />`しています。

```
<uses-permission android:name="string"
   android:maxSdkVersion="integer"/>
```

android:nameこれはなのです

android:maxSdkVersionこのをアプリにするのAPIレベル。このをすることはオプションで、アプリがとするがのAPIレベルでになったにのみするがあります。

サンプル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>
```

* についてもしてください。

オンラインでマニフェストファイルをむ https://riptutorial.com/ja/android/topic/1848/マニフェストファイル

222: メニュー

- inflater.inflatemenu.your_xml_file、menu;

パラメータ

パラメータ	
`inflate(int menuRes, Menu menu)`	されたXMLリソースからメニューをさせます。
`getMenuInflater ()`	このコンテキストをつ`MenuInflater`をします。
`onCreateOptionsMenu (Menu menu)`	アクティビティのオプションメニューのをします。メニューをメニューにするがあります。
`onOptionsItemSelected (MenuItem item)`	このメソッドは、オプションメニューのがされるたびにびされます

メニューについてもっとるには、 これをんでください。それがにてばい

Examples

りきオプション メニュー

Androidにはデフォルトのオプションメニューがあり、いくつかのオプションがあります。よりくのオプションをするがあるは、これらのオプションをグループしてにすることがにかなっています。オプションは、それらのにりすなわちをくことによってグループすることができます。ディバイダをにするために、のテーマをできます。

```
<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>
```

テーマをすることで、りをメニューにすることができます。

メニューにカスタムフォントをする

```
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);
}
```

にアクティビティで

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    applyFontToMenu(menu,this);
    return true;
}
```

アクティビティでのメニューの

のメニューをするには、プロジェクトの`res/menu/`ディレクトリにXMLファイルをし、のをむメニューをします。

- `<menu>` すべてのメニューをするメニューをします。
- `<item>` メニューのをするMenuItemをします。また、サブメニューをするためにネストされたをすることもできます。

ステップ1

のxmlファイルをのようにします。

`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>
```

ステップ2

オプションメニューをするには、あなたのアクティビティで`onCreateOptionsMenu()`をオーバーライドします。

このでは、メニューリソースXMLファイルの`res/menu/main_menu.xml`されています

```
 @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }
```

ユーザがオプションメニューからアイテムをすると、システムはあなたのアクティビティのオーバーライドされた`onOptionsItemSelected()`メソッドをびします。

- このメソッドは、されたMenuItemをします。
- メニューメニューリソースの`res/menu/main_menu.xml`の`android:id attribute`でされている`getItemId()`ののIDをす`getItemId()`びすことによってをでき`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);
    }
}
```

ラッピング

`Activity`コードはのようになります。

```
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);
        }
    }
}
```

のメニューがどのようにされるかのスクリーンショット



オンラインでメニューをむ https://riptutorial.com/ja/android/topic/2028/メニュー

223: メモリリーク

Examples

のメモリリークとその

1.あなたのをする

たとえば、Toastは1つではなくくのアクティビティでることができるので、トーストのために `getApplicationContext()` をし、アクティビティがしてもサービスをしけることができるため、のようなサービスをできます。

```
Intent myService = new Intent(getApplicationContext(), MyService.class);
```

どのようなコンテキストがであるかのなガイドとして、このをしてください。

	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	

ここでのにするオリジナル 。

2. Contextへの

なメモリリークのいは、 `View` へのをすることです。 `View` は、 `Context` へのがあります。つまり、ビューのいアクティビティは、アプリがするまでガベージコレクションされません。をさせると、あなたのアプリは2メモリにされます。

ビュー、コンテキスト、またはそののがにないことをしてください。

3.にサービスをしていることをします。

たとえば、GoogleロケーションサービスapiをするintentServiceがあります。そして、は `googleApiClient.disconnect();` をびすことをれていました `googleApiClient.disconnect();`

```
//Disconnect from API onDestroy()
if (googleApiClient.isConnected()) {
    LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient,
GoogleLocationService.this);
    googleApiClient.disconnect();
}
```

4.とビットマップのをする

Squareのライブラリ**Picasso**をしている、 `.fit()` をしないでメモリがリークしていたことがわかりました。これは、メモリフットプリントを50MBから19MBににしました。

```
Picasso.with(ActivityExample.this)                    //Activity context
                .load(object.getImageUrl())
                .fit()                                 //This avoided the OutOfMemoryError
                .centerCrop()                          //makes image to not stretch
                .into(imageView);
```

5.ブロードキャストレシーバをしているは、をします。

6. `java.util.Observer` オブザーバー・パターンをしているは、のようにします。

`deleteObserver(observer);` をずしてください `deleteObserver(observer);`

AsyncTaskでアクティビティをらさないようにする

> の AsyncTaskがっているくのとしをれてここでするメモリリークから。したがって、このAPIをにしていないは、このAPIにしてください。くのThread、EventBus、RxAndroidなどがあります。

`AsyncTask` よくあるいの1つは、ホスト `Activity` または `Fragment` へのいをすることです。

```
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!
  }
}
```

これは、 `AsyncTask` が `Activity` よりもすぐに `AsyncTask` ため、タスクがされているのがしたなど、になります。

これをうしいは、タスクををキャプチャしない`static`クラスにし、ホストのいをすることです。`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
    }
  }
}
```

アクティビティのコールバック

クラスをするたびに、そのクラスへののがされます。だからあなたがくとき

```
public class LeakyActivity extends Activity
{

...

    foo.registerCallback(new BarCallback()
    {
        @Override
        public void onBar()
        {
            // do something
        }
    });
}
```

にはLeakyActivityインスタンスへのをfooにしています。ユーザーがLeakyActivityからすると、このによってLeakyActivityインスタンスがガベージコレクションされなくなります。これはなリークです。アクティビティはビューへのをするため、メモリのきなオブジェクトです。

このれをける

もちろん、のコールバックをアクティビティですることはけることができます。アクティビティライフサイクルにして、すべてのコールバックのをすることもできます。そのようです

```
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);
    }
}
```

クラスのアクティビティコンテキスト

くの、Androidクラスのいくつかをいやすいユーティリティクラスにラップしたいとうでしょう。これらのユーティリティクラスでは、Android OSやアプリケーションのリソースにアクセスするためのコンテキストがになることがよくあります。これのなは、SharedPreferencesクラスのラッパーです。 Androidsのにアクセスするには、のようにするがあります。

```
context.getSharedPreferences(prefsName, mode);
```

それで、のクラスをしたくなるかもしれません

```
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);
    }
```

```
}
```

、アクティビティコンテキストで`init()`をびすと、LeakySharedPrefsWrapperはアクティビティへのをし、ガベージコレクションされないようにします。

ける

ヘルパーをびすときは、 `context.getApplicationContext();`をしてアプリケーションコンテキストをできます`context.getApplicationContext();`

ヘルパーをするときは、されたコンテキストからアプリケーションコンテキストをできますアプリケーションコンテキストでgetApplicationContextをびすとアプリケーションコンテキストがされます。ラッパーへのはです

```
public static void init(Context context)
{
    sContext = context.getApplicationContext();
}
```

アプリケーションのコンテキストがユースケースにしていない、ユーティリティにContextパラメータをめることができます。これらのコンテキストパラメータへのをけるべきです。この、ソリューションはのようになります。

```
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);
}
```

LeakCanaryライブラリでメモリリークをする

LeakCanaryは、オープンソースのJavaライブラリで、デバッグビルドのメモリリークをします。

`build.gradle`をするだけです

```
 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'
 }
```

に、 `Application`クラスで

```
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);
  }
}
```

LeakCanaryは、 デバッグビルドでアクティビティのメモリリークがされたときに、にをします。

リリースコードには、 `leakcanary-android-no-op`にする2つののクラスのLeakCanaryへのはまれません。

リスナーによるアクティビティのをける

アクティビティでリスナーをまたはするは、リスナーがされているオブジェクトのライフサイクルににしてください。

ユーザーがログインまたはログアウトしたときにをつさまざまな/があるアプリケーションをえてみましょう。これをう1つのは、ユーザーのがしたときにをけるためにできる`UserController`シングルトンインスタンスをつ`UserController`です。

```
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();
    }
}
```

に、 `SignInActivity` 2つのアクティビティがあります。

```
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();
    }
}
```

そして`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();
    }
}
```

このでは、ユーザがログインしてからびログアウトする`MainActivity`に、 `MainActivity`インスタンスがリークされるということが`MainActivity`ます。 リークは、 `UserController#listeners`アクティビティへのがあるためにし`UserController#listeners`。

のクラスをリスナーとしてしても、アクティビティはまだリークします。

```
...
this.userController.registerUserStateChangeListener(new UserController.StateListener() {
    @Override
    public void userLoggedIn() {
        showUserAccount();
    }

    @Override
    public void userLoggedOut() {
        finish();
    }
});
...
```

のクラスにはクラスこのはアクティビティへののがあるため、アクティビティはまだリークします。このため、クラスのインスタンスメソッドをクラスからびすことができます。、クラスへのをたないクラスののはクラスです。

つまり、なクラスのすべてのインスタンスは、それらをしたクラスのインスタンスへののをします。

これをするには、 `UserController#listeners`からリスナーをするメソッドをするか、または`WeakReference`をしてリスナーのをするのいずれかをして、これをするが2つあります。

1リスナーをする

`removeUserStateChangeListener(StateListener listener)`というしいメソッドをし`removeUserStateChangeListener(StateListener listener)`。

```
public class UserController {

    ...

    public void registerUserStateChangeListener(StateListener listener) {
        listeners.add(listener);
    }

    public void removeUserStateChangeListener(StateListener listener) {
        listeners.remove(listener);
```

```
    }

    ...
}
```

に、アクティビティの`onDestroy`メソッドでこのメソッドをびします。

```
public class MainActivity extends Activity implements UserController.StateListener{
    ...

    @Override
    protected void onDestroy() {
        super.onDestroy();
        userController.removeUserStateChangeListener(this);
    }
}
```

このにより、ユーザーのログインとログアウトに`MainActivity`のインスタンスが`MainActivity`はなくなりました。ただし、ドキュメントがでないは、 `UserController`をしているのが、アクティビティがされたときにリスナーのをするがあることをれてしまうがあります。

の2の

まず、いがであるかをしてめましょう。いは、そのがすように、オブジェクトへのいをします。いであるのインスタンスフィールドとして、はガベージコレクタGCがオブジェクトをするのをめません。のでは、 `UserController`が`WeakReference`をリスナーをするためにした、 `MainActivity`がされたで`MainActivity`をガベージ`MainActivity`することができます。

すると、いは、にもこのオブジェクトへのいがない、にみ、それをすることをGCにえます。

`WeakReference`リストをしてリスナーをするように`UserController`をしましょう

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

このでは、Listenerがされているかどうかはありません。なぜなら、`UserController`はいずれのリスナーにしてもいをしていないからです。しかし、このをくのはです。したがって、`WeakCollection`というクラスをしましょう。

```
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);
        }
    }
}
```

`WeakCollection<T>`うわりに`UserController`をきしてみましょう。

```
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();
        }
    }

    ...
}
```

のコーードにすように、 `WeakCollection<T>`は、のリストのわりに`WeakReference`をするのになコーードをすべてします。これをすべてにするには`UserController#removeUserStateChangeListener(StateListener)`へのびしが`UserController#removeUserStateChangeListener(StateListener)`た、リスナーとそれがしているすべてのオブジェクトはリークしません。

クラス、ハンドラ、タイマータスク、スレッドでメモリリークをける

アンドロイドでは、すべてのがプロジェクトでなくとも1`Anonymous Class` Runnableをします。`Anonymous Class`はアクティビティへのがあります。タスクをすると、アクティビティ—はタスクがするまでされません。
Exampleは、ハンドラとAnonymous `Runnable`クラスをします。 `Runnable`がするにアクティビティをすると、メモリがリ—クします。

```
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...
```

どうやってするの

1. `Anonymous Class`しないでください。または、 `Static class`がで、 `WeakReference`をアクティビティやビュ—などにすがあります。 `Thread`は`Anonymous Class`とじです。
2. アクティビティがされたときに`Handler` 、 `Timer`キャンセルします。

オンラインでメモリリ—クをむ https://riptutorial.com/ja/android/topic/2687/メモリリ—ク

224: モシ

き

MoshiはAndroidとJavaのJSONライブラリです。 JSONをJavaオブジェクトに、JavaをJSONににできます。

れずに、にREADMEをんでください

Examples

JSONをJavaにする

```
String json = ...;

Moshi moshi = new Moshi.Builder().build();
JsonAdapter<BlackjackHand> jsonAdapter = moshi.adapter(BlackjackHand.class);

BlackjackHand blackjackHand = jsonAdapter.fromJson(json);
System.out.println(blackjackHand);
```

JavaオブジェクトをJSONとしてシリアルする

```
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);
```

アダプタ

Moshiには、Javaのコアデ―タをみきするためのみみサポ―トがあります。

- プリミティブint、float、char ...とボックスされたInteger、Float、Character ...。
-
- コレクション
- リスト
- セット
- マップ

これは、モデルクラスをフィ―ルドごとにきすことによってサポ―トします。のでは、Moshiはのクラスをします。

```
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"
    }
  ]
}
```

オンラインでモシをむ https://riptutorial.com/ja/android/topic/8744/モシ

225: ユニバーサルローダ

けれなURIの

```
"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

ユニバーサルイメージローダの

1. *build.gradle*ファイルにのをします。

```
compile 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
```

2. *AndroidManifest.xml*ファイルにのをします。

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

3. Universal Image Loaderをします。ののにこれをうがあります

```
ImageLoaderConfiguration config = new ImageLoaderConfiguration.Builder(this)
    // ...
    .build();
ImageLoader.getInstance().init(config);
```

なオプションは、 ここでつけることができます 。

な

1. イメージをロードしてビットマップにデコードし、 `ImageView` または`ImageAware`インターフェイスをするそののビューにビットマップをします。

```
ImageLoader.getInstance().displayImage(imageUri, imageView);
```

2. イメージをロードし、ビットマップにデコードし、ビットマップをコールバックにします。

```
ImageLoader.getInstance().loadImage(imageUri, new SimpleImageLoadingListener() {
    @Override
    public void onLoadingComplete(String imageUri, View view, Bitmap loadedImage) {
```

```
        // Do whatever you want with the bitmap.
    }
});
```

3. イメージをロードし、ビットマップにデコードし、ビットマップをしてします。

```
Bitmap bmp = ImageLoader.getInstance().loadImageSync(imageUri);
```

オンラインでユニバーサルローダをむ https://riptutorial.com/ja/android/topic/2760/ユニバーサルローダ

226: ライブラリダガー2アプリケーションのき

Dagger 2は、 GitHubでしたように 、にするコンパイルののアプローチです。 Dagger 1.xのアプローチをのにくDagger 2.xは、すべてのをし、の`ObjectGraph` / `Injector`をユーザーの`@Component`インターフェイスに`@Component`コードのさをさ`@Component`ます。

1. アプリケーションのライブラリMaven、gradle、Javaプロジェクト
2. Draggerの
3. なリンクドキュメントとデモ
4. Draggerコンポーネントをしてする

Dagger 2 API

Dagger 2はいくつかのなアノテーションをしています。

メソッドがをするクラスの**@Module**

@Moduleクラスのメソッドの@Provides

@Injectでをするコンストラクタ、フィールド、またはメソッド

@Componentはモジュールとののブリッジインターフェースです

なリンク

GitHub https **:** //github.com/google/dagger

UserGuideGoogle https **:** //google.github.io/dagger/users-guide.html

https **:** //google.github.io/dagger/resources.html

Vogellaチュートリアル http **://www.vogella.com/tutorials/Dagger/article.html**

Codepathチュートリアル https **:** //github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Examples

オブジェクトの**@Module**クラスと**@Singleton**アノテーションをする

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

すべてのプロバイダまたはメソッドには`@Provides`アノテーションがで、クラスには`@Module`アノテーションがです。 `@Singleton`アノテーションは、オブジェクトのインスタンスが1つだけすることをします。

オブジェクトのをする

さまざまなモデルのプロバイダがあるので、それらをリクエストするがあります。 `Vehicle`が`Motor`とするのとじように、 `Vehicle`コンストラクタで`@Inject`アノテーションをのようにするがあります。

```
@Inject
public Vehicle(Motor motor){
    this.motor = motor;
}
```

`@Inject`アノテーションをすると、コンストラクタ、フィールド、またはメソッドのをリクエストできます。このでは、コンストラクタにをしています。

@Modulesと@Injectの

のプロバイダとのの`@Module`をしてそれらをし、そしてクラス`@Inject`いてわれる`@Component`インターフェースであります

```
import javax.inject.Singleton;
import dagger.Component;

@Singleton
@Component(modules = {VehicleModule.class})
public interface VehicleComponent {
    Vehicle provideVehicle();
}
```

`@Component`アノテーションでは、どのモジュールをするかをするがあります。このでは、このでされている`VehicleModule`がされています 。よりくのモジュールをするがあるは、カンマをりとして

してモジュールをします。

@Component インターフェイスをしたオブジェクトの

すべてのができたら、このインターフェースのインスタンスをし、なオブジェクトをするメソッドをびさなければなりません。

```
VehicleComponent component = Dagger_VehicleComponent.builder().vehicleModule(new
VehicleModule()).build();
vehicle = component.provideVehicle();
Toast.makeText(this, String.valueOf(vehicle.getSpeed()), Toast.LENGTH_SHORT).show();
```

`@Component`アノテーションをつインターフェイスのしいオブジェクトをしようとすると、`Dagger_<NameOfTheComponentInterface>` このは`Dagger_VehicleComponent` をしてインターフェイスをし、ビルダーメソッドをしてそののすべてのモジュールをびすがあります。

オンラインでライブラリダガー2アプリケーションのをむ
https://riptutorial.com/ja/android/topic/9079/ライブラリダガー2-アプリケーションの

227: ライブラリをMavenリポジトリにする

Examples

Mavenに.aarファイルをする

Mavenのリポジトリにするには、gradleの "maven-publish"プラグインをできます。

ライブラリモジュールの`build.gradle`ファイルにプラグインをするがあります。

```
apply plugin: 'maven-publish'
```

パブリケーションとそのアイデンティティを`build.gradle`ファイルにもするがあります。このアイデンティティは、されたpomファイルにされます。、このパブリケーションをインポートするにします。また、パブリッシュするアーティファクトをするがあります。たとえば、ライブラリをしたにされた.aarファイル。

```
publishing {
    publications {
        myPulication(MavenPublication) {
            groupId 'com.example.project'
            version '1.0.2'
            artifactId 'myProject'
            artifact("$buildDir/outputs/aar/myProject.aar")
        }
    }
}
```

リポジトリのURLをするもあります

```
publishing{
    repositories {
        maven {
            url "http://www.myrepository.com"
        }
    }
}
```

ここになライブラリ`build.gradle`ファイルがあります

```
apply plugin: 'com.android.library'
apply plugin: 'maven-publish'

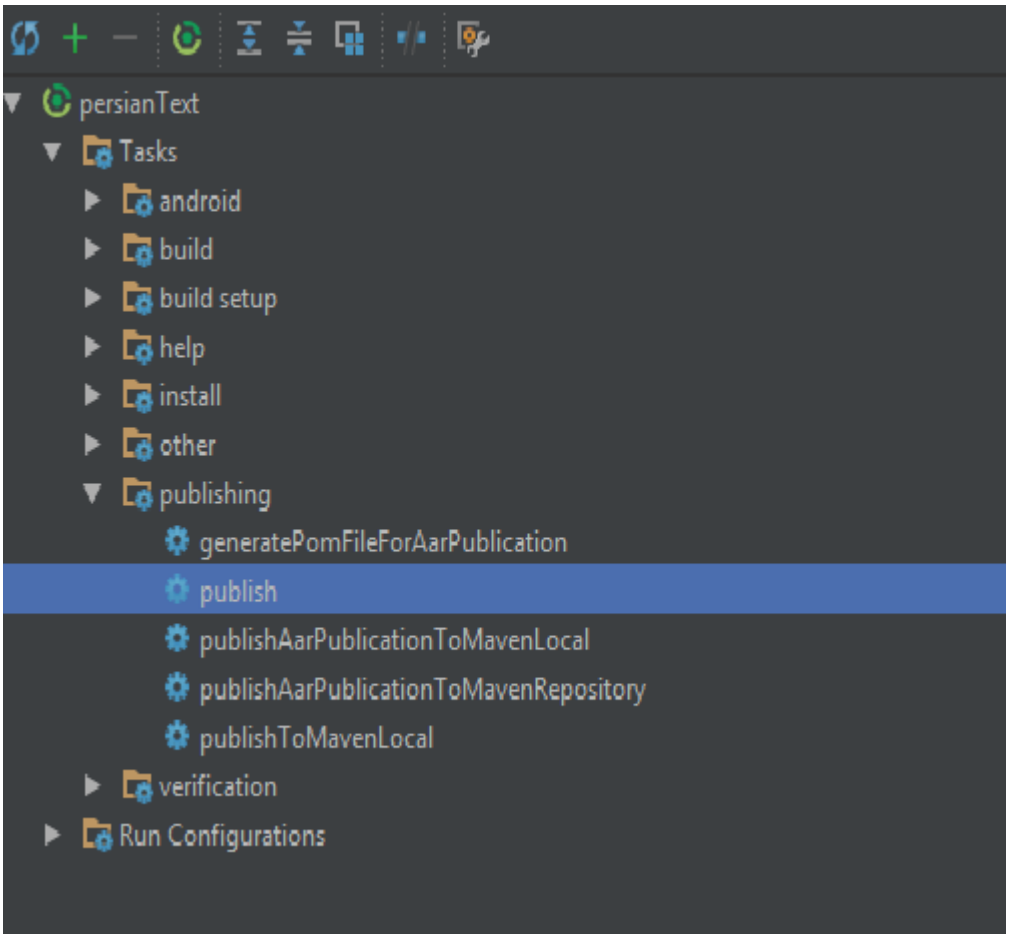
buildscript {
   ...
}
android {
    ...
}
publishing {
```

```
    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"
        }
    }
}
```

パブリッシュのために、gradle consoleコマンドをすることができます

    グラデルパブリッシュ

またはgradleタスクパネルからできます



オンラインでライブラリをMavenリポジトリにするをむ
https://riptutorial.com/ja/android/topic/9359/ライブラリをmavenリポジトリにする

228: リストビュー

き

ListViewはやデータベースのようなデータソースからいくつかのをグループし、それらをスクロールリストにするビューグループです。データは、Adapterクラスをしてリストビューにバインドされます。

`ListView`は、スクロールなのリストをするビューグループです。
リストアイテムは、やデータベースクエリなどのソースからコンテンツをし、アイテムのをリストにされたビューにする`Adapter`をして、リストににされます。

レイアウトのコンテンツがであるか、またはにされていないは、 `AdapterView`をサブクラスするレイアウトをして、にビューをレイアウトに`AdapterView`ことができます。 `AdapterView`クラスのサブクラスは、 `Adapter`をしてレイアウトにデータをバインドします。

`ListView`をするに、 `RecyclerView`もチェックするがあります。

Examples

CursorAdapterをしたフィルタリング

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

たとえば、ユーザーが`EditText`するたびにクエリがされるとします。

```
    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) {

        }
    });
```

カスタムArrayAdapter

デフォルトでは、ArrayAdapterクラスはの`toString()`をびし、そのをTextViewにすることによって、のをします。

のなビューをするにはArrayAdapterクラスをしてgetView `getView()`メソッドをオーバーライドし、になViewのタイプをす`getView()`。

えば

```
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);
         }
    }
}
```

ArrayAdapterをしたなListView

デフォルトでは、 `ArrayAdapter`は、の`toString()`をびし、そのを`TextView`することによって、のビューをします。

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, myStringArray);
```

ここで、 `android.R.layout.simple_list_item_1`は、のの`TextView`をむレイアウト`TextView`。

に、 `ListView` `setAdapter()`をびします。

```
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

たとえば、ImageViewsなどのにTextViewsのものをする、または`toString()`のデータをいくつかしてビューをたすは`getView(int, View, ViewGroup)`をオーバーライドして`getView(int, View, ViewGroup)`のタイプをします。 このをしてください 。

オンラインでリストビューをむ https://riptutorial.com/ja/android/topic/4226/リストビュー

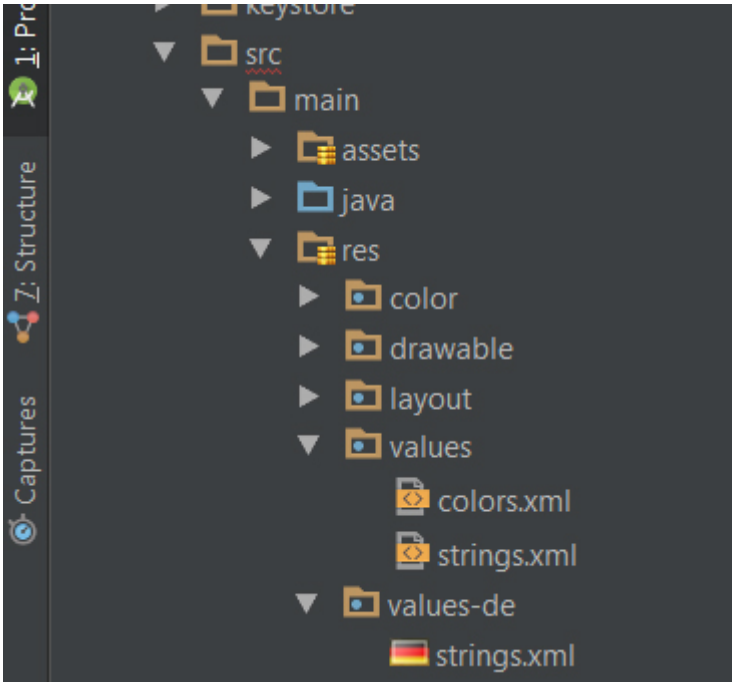
229: リソース

Examples

をする

は、サポートするごとになるstrings.xmlをすることで、することができます。

ISOコードをとしてしいvaluesディレクトリをして、しいをします。たとえば、ドイツセットをすると、はのようになります。



システムがされたをすと、のxmlがにチェックされ、つからなければ、デフォルトのstrings.xmlファイルのがされます。キーはでじままで、だけがされます。

コンテンツの

/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>
```

をする

は、リソースファイル`strings.xml`されます。それらは、 `<string>` XMLをしてされます。

strings.xmlのは、をにすることです。isoコードごとにstrings.xmlをすることができます。したがって、システムが 'app_name'をすると、のにするxmlファイルがにチェックされ、つからなければ、デフォルトのstrings.xmlファイルのエントリがされます。つまり、のをローカライズするだけでなく、のをローカライズすることもできます。

/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>
```

がXMLリソースファイルでされると、アプリケーションののでできます。

アプリケーションのXMLプロジェクトファイルは、 `@string/string_name`をして`<string>`をできます。たとえば、アプリケーションのマニフェスト /manifests/AndroidManifest.xmlファイルには、Android Studioのデフォルトでのがまれています

```
android:label="@string/app_name"
```

これはアンドロイドに、アプリケーションがランチャーにインストールまたはされたときにアプリのとしてする`<string>` app_name `<string>`リソースをします。

アンドロイドのXMLファイルから`<string>`リソースをするのは、レイアウトファイルにあります。たとえば、のは、にした`hello_world`をするTextViewをしています。

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world"/>
```

また、アプリケーションのJavaから`<string>`リソースにアクセスすることもできます。 Activityクラスのからじ`hello_world`をびすには、のようにします。

```
String helloWorld = getString(R.string.hello_world);
```

をする

リソースファイルにのきみをするには

res / values / filename.xml

```
<string-array name="string_array_name">
    <item>text_string</item>
    <item>@string/string_id</item>
</string-array>
```

えば

res / values / arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="string_array_example">
        <item>@string/app_name</item>
        <item>@string/hello_world</item>
    </string-array>
</resources>
```

のようなJavaからそれをしてください

```
String[] strings = getResources().getStringArray(R.array.string_array_example;
Log.i("TAG",Arrays.toString(strings)));
```

```
I/TAG: [HelloWorld, Hello World!]
```

ディメンションをする

ディメンションは、リソースファイル`dimens.xml`されます。それらは`<dimen>`をしてされます。

res / values / dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="small_padding">5dp</dimen>
    <dimen name="medium_padding">10dp</dimen>
    <dimen name="large_padding">20dp</dimen>

    <dimen name="small_font">14sp</dimen>
    <dimen name="medium_font">16sp</dimen>
    <dimen name="large_font">20sp</dimen>
</resources>
```

なるをすることができます

- **sp**スケールにしないピクセル。フォントの。
- **dp**にしないピクセル。のすべてのために。
- **pt**ポイント
- **px**ピクセル
- **mm**ミリメートル
- イムインチ

ディメンションは`@dimen/name_of_the_dimension`というでXMLでできるようになりました。

えば

```
<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>
```

をする

は、 `integers.xml`というのリソ―スファイルにされますが、ファイルはにできます。は、のファイルにすように、 `<integer>`をしてされます。

res / values / integers.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer name="max">100</integer>
</resources>
```

のにすように、 `@integer/name_of_the_integer`というで`@integer/name_of_the_integer` XMLでできるようになりました。

```
<ProgressBar
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:max="@integer/max"/>
```

をする

リソ―スファイルにのきみをするには

res / values / filename.xml

```
<integer-array name="integer_array_name">
    <item>integer_value</item>
    <item>@integer/integer_id</item>
</integer-array>
```

えば

res / values / arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer-array name="fibo">
        <item>@integer/zero</item>
```

```
        <item>@integer/one</item>
        <item>@integer/one</item>
        <item>@integer/two</item>
        <item>@integer/three</item>
        <item>@integer/five</item>
    </integer-array>
</resources>
```

のようなJavaからそれをしてください

```
int[] values = getResources().getIntArray(R.array.fibo);
Log.i("TAG",Arrays.toString(values)));
```

```
I/TAG: [0, 1, 1, 2, 3, 5]
```

をする

は、、 `/res/values/`フォルダーの`colors.xml`というのリソースファイルにされます。

それらは`<color>`でされます

```
<?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>
```

は、のいずれかののカラーチャネル`(0 - FF)` 16のカラーでされます。

- `#RGB`
- `#ARGB`
- `#RRGGBB`
- `#AARRGGBB`

- A - アルファチャンネル - 0はにで、FFはです
- R - チャンネル
- G - のチャンネル
- B - のチャンネル

されたはXMLでのでできます`@color/name_of_the_color`

えば

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/blackOverlay">
```

コードでのの

これらのでは、 `this`がアクティビティであるとし`this`ます。コンテキストもそのでできます。

1.6

```
int color = ContextCompat.getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

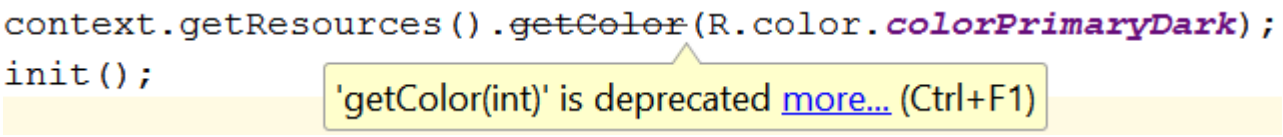
6.0

```
int color = this.getResources().getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

の`colorPrimary`では、 `colorPrimaryDark`と`colorAccent`をして、 `styles.xml`カスタムAndroidテーマをするにするマテリアルデザインのをしています。 Android Studioでしいプロジェクトをするとにされます。

「」のなしでリソースをする

Android API 23をすると、ににそのようながえます



このようなは、Android APIのリソースのにするなによってします。
の

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

すべきです。しかし、 `android.support.v4`ライブラリにはのがあります。

`build.gradle`ファイルにのをします。

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

サポートライブラリのすべてのメソッドがです

```
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));
```

さらにサポートライブラリからのよりくのメソッドをすることができます

```
ViewCompat.setElevation(textView, 1F);
```

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

メニューリソースをし、アクティビティ/フラグメントでする

res / menuにメニューをする

```
<?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>
```

のそののオプションについては、をしてください。 メニューリソース

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);
}
```

ビュ―のにのメソッドをびすには、 `getActivity().invalidateOptionsMenu();`びし
`getActivity().invalidateOptionsMenu();`

`Fragment`もう1つのコ―ルがです

```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    setHasOptionsMenu(true);
    super.onCreateView(inflater, container, savedInstanceState);
}
```

strings.xmlのフォ―マット

strings.xmlファイルにをすることで、のもです。のは、をにレイアウトにアタッチするのとはなり、のようなコ―ドでするがあることです。

```
<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);
```

のでは、
1 $ s
"はのからります。
「1」は1のパラメ―タをし、
'$'はパラメ―タとののセパレ―タとしてされ、
's'はをします 'd'はにされます

`getString()`は`Context`または`Resources`メソッドです。 つまり、 `Activity`インスタンスでできます `getActivity().getString()`そうでないは、 `getActivity().getString()`または`getContext().getString()` それぞれできます。

のリストをする

カラ―リストはカラ―としてできますが、されているビュ―のにじてわります。

するには、 `res/color/foo.xml`リソ―スファイルをします

```
<?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>
```

はされたでされ、されたがビュ―ののとするのがされます。ですから、ステ―トセレクタをしな

くても、にキャッチオールをすることをおめします。

は、カラーリテラルをすることも、のでされたをすることもできます。

ストリング・プレリュールの

のとのをするために、 *strings.xml*ファイルにをし、のにすようになるをリストすることができます。

```
<?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>
```

このは、のにすように、 `Resources`クラスの`getQuantityString()`メソッドをしてJavaコードからアクセスできます。

```
getResources().getQuantityString(R.plurals.hello_people, 3, 3);
```

ここでは、のパラメータ`R.plurals.hello_people`がリソースです。 2のパラメータこのでは`3` をして、しい`quantity`をします。 3のパラメータこのでは`3` は、`%d`にされるです。

なアルファベットはのとおりです。

```
few
many
one
other
two
zero
```

ないすべてのロケールは、すべてのサポートしていることにすることがである`quantity`。たとえば、には`one`のがありません。には`other`のものとじであるため、には`zero`はありません。サポートされていない`quantity`インスタンスは、IDEによってLintのとしてフラグがてられますが、されるとなエラーはしません。

リソースでされたオブジェクトのをインポートする

カスタムオブジェクトをし、アプリケーションのリソースでするがあるがあります。このようなオブジェクトは、 `Java Integer` 、 `Float` 、 `String`などでできます。

アプリケーションリソースでされたオブジェクトをインポートするのをにします。オブジェクト`Category` 3つのプロパティをします

- ID
-
-

この`POJO`は`categories.xml`ファイルにし`categories.xml`。 はカテゴリにしてじプロパティがされています。

1. あなたのためのモデルをする

```
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. **`res/values`**フォルダにファイルをします。

`categories.xml`

3. リソースからなるモデルをする

```
<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. リソースファイルにをする

```
<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. それらをインポートするをします。

```
@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パッチ

9パッチはなで、ばすことができるはなにのマ―カ―でされます。

ここにきなチュ―トリアルがあります 。
とてもくても、それはまだがあり、くのが9パッチ･ギアをくするのにちました。

ながら、、そのペ―ジはしばらくのされていますはもうしています。

したがって、のいサ―バ―でアンドロイドけにそのペ―ジのコピ―をするがあります。

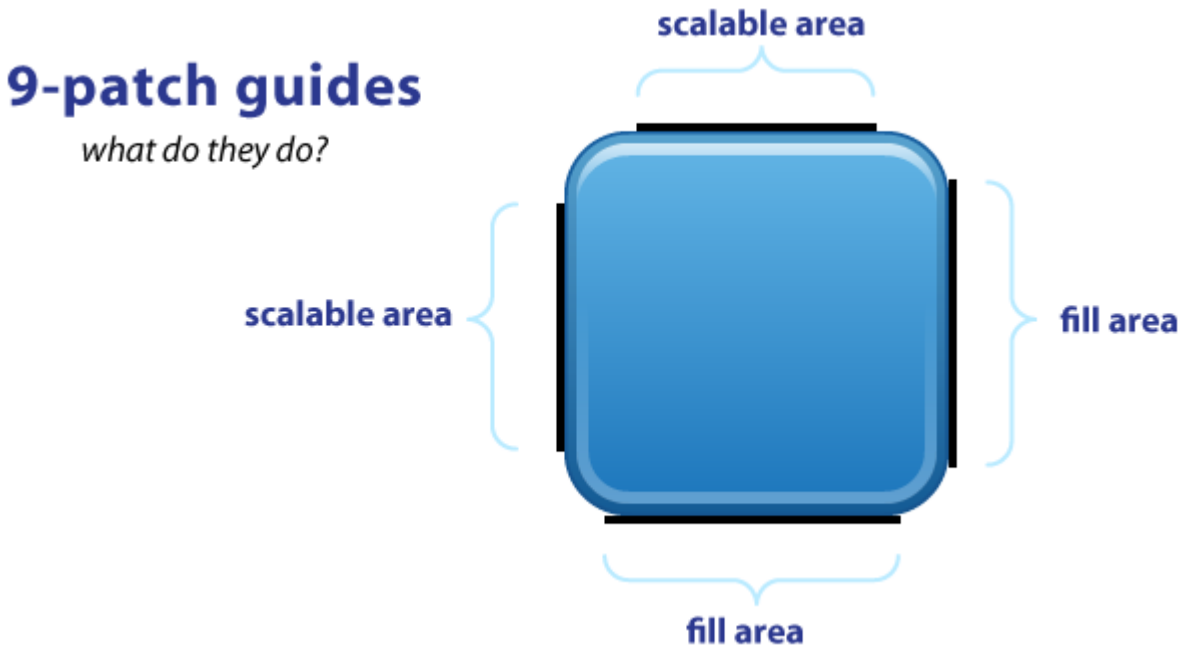
ここにあります。

アンドロイドUIの9つのパッチのなガイド2011518

がのAndroidアプリをしているに、9パッチ9.pngがしていて、されていないことがしました。しばらくすると、ついにそれがどのようにいているのかをいめ、のがそれをするのをけるためにかをにげることにめました。

に、9パッチは、9スライスまたはスケ―ル9のなをするためにpngのをします。ガイドは、のにかれた1ピクセルので、のとりしをします。ファイルname.9.pngにをけることによって、Androidは9.pngをし、いガイドをしてビットマップを/してめくします。

なガイドマップはのとおりです。



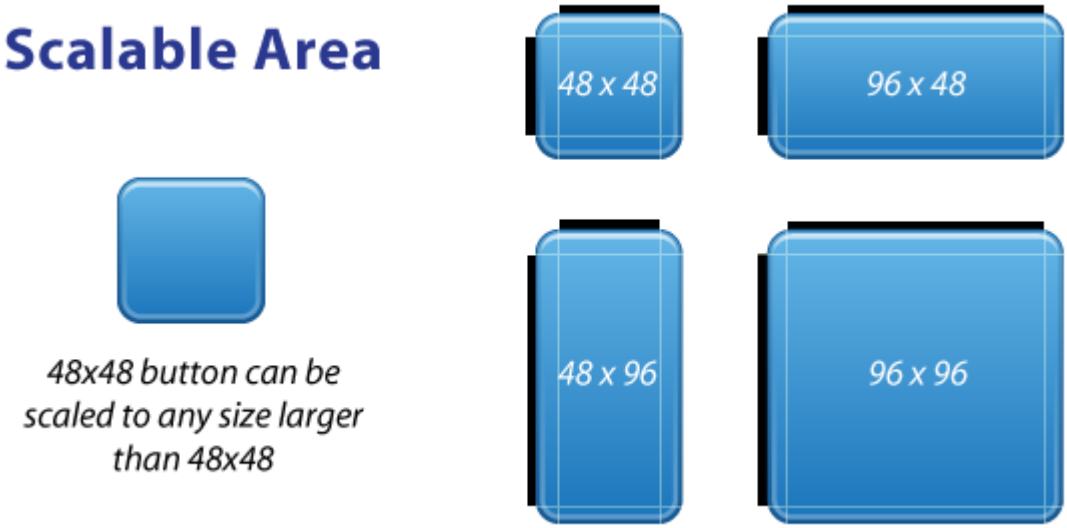
ごのとおり、のにガイドがあります。 TOPとLEFTのガイドはを/するためのガイドです9スライス。また、RIGHTとBOTTOMガイドはりつぶしをします。

のガイドラインはからりられ/されます。アプリにはされません。ガイドは1ピクセルでなければならないので、48×48ボタンがなは、には50×50になります。 1ピクセルよりもいものは、あなたののとしてります。 のでは、をめるために4ピクセルのガイドがありますが、には1ピクセルのみであるがあります。

あなたのガイドはくられていなければなりません000000。000001やアルファのわずかないでさえ、にしなくなります。このはらかではない*、それはかにするはい。に。あなたはっている。

また、1ピクセルのアウトラインのりのはにでなければならないことにしてください。これにはイメ―ジの4つのコ―ナ―がまれます。それらはにでなければなりません。これは、あなたがづくよりもきなになるがあります。たとえば、Photoshopでを/すると、アンチエイリアスされたピクセルがされます。このピクセルにはほとんどえないピクセルもまれているため、するがあります*。Photoshopで/するがあるは、[イメ―ジサイズの]プルダウンメニュ―サイズポップアップメニュ―のにあるの[]をして、ガイドにシャ―プなエッジをします。

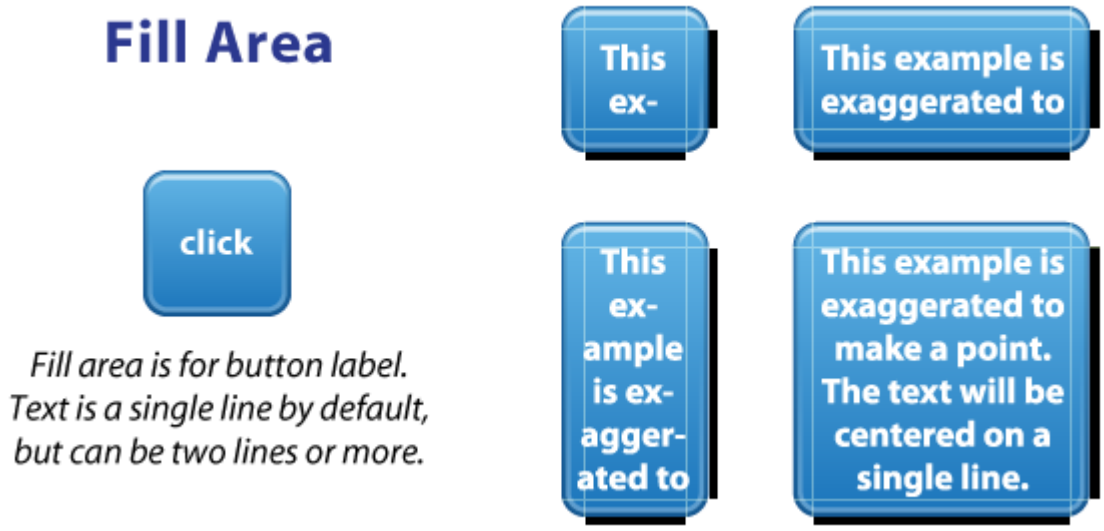
*updated 1/2012これはにはのキットの「」です。は、にれた9パッチではなく、のすべてのやリソ―スがしてしまうため、それがれます。



TOPおよびLEFTガイドは、のスケーラブルなをするためにされます。LEFTはスケーリングのさ、TOPはスケーリングのです。としてボタンイメージをすると、ボタンがいでとにび、コーナーなどのすべてがじサイズのままになることをします。このボタンをすると、のサイズにでき、なをできるボタンをできます。

9パッチのはされないことにすることがです。だからできるだけさくすることがです。

また、スケールのにをすこともできます。たとえば、でなエッジをつボタンがあるは、LEFTガイドのにいくつかのピクセルをしておくことができます。ののはスケールされず、のだけで、シャープなはアンチエイリアスやファジーにならないでしょう。

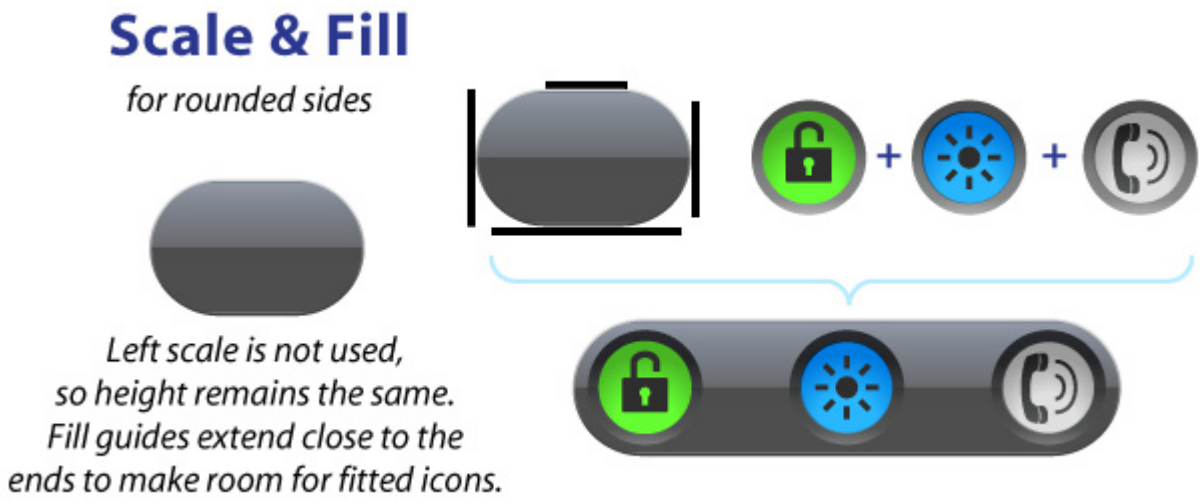


りガイドはオプションで、テキストラベルのようなもののためのをするをします。りつぶしは、テキスト、アイコン、またはそののものをくためにイメージにどれくらいのスペースがあるかをします。 9パッチはボタンだけでなく、にもできます。

のボタンとラベルのは、にりつぶしのアイデアをするためにされています - ラベルはにではありません。にえば、ボタンラベルは1のテキストであるため、Androidではのラベルをどのようにう

のかしていません。

に、LinearLayoutなどのとにみをびたなど、スケールガイドとりつぶしガイドがどのようになるかをすデモンストレーションです



このでは、LEFTガイドはされていませんが、まだガイドがです。はにされません。それはにTOPガイドにづいてにスケールされます。りつぶしガイドをると、とのガイドは、のしたエッジをたすをえてされています。これにより、いボタンをのエッジのくにして、きつくフィットしたにすることができます。

そうです。あなたがそれをると、9パッチはにです。スケールをうにはなではありませんが、りつぶしとマルチラインのスケールガイドは、の9スライスやスケール9よりもがあります。それをしてみるとすぐにそれをするでしょう。

のアルファレベル

16

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        |
    -------------------------------
```

に45をする。

```
<color name="red_with_alpha_45">#73FF0000</color>
```

の16 - FF0000

に73のをすることができます - 73FF0000

strings.xmlファイルの

リソースは、オプションのテキストのスタイリングとをしてアプリケーションのテキストをします。アプリケーションにをできる3のリソースがあります。

```
XML resource that provides a single string.
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="string_name">text_string</string>
</resources>
```

レイアウトでこのをするには

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/string_name" />
```

```
XML resource that provides an array of strings.
```

```
    <resources>
<string-array name="planets_array">
    <item>Mercury</item>
    <item>Venus</item>
    <item>Earth</item>
    <item>Mars</item>
</string-array>
```

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

の

```
XML resource that carries different strings for pluralization.
```

```
<?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>
```

```
int count = getNumberOfsongsAvailable();
Resources res = getResources();
String songsFound = res.getQuantityString(R.plurals.plural_name, count, count);
```

オンラインでリソースをむ https://riptutorial.com/ja/android/topic/108/リソース

230: リリースのためにAndroidアプリにする

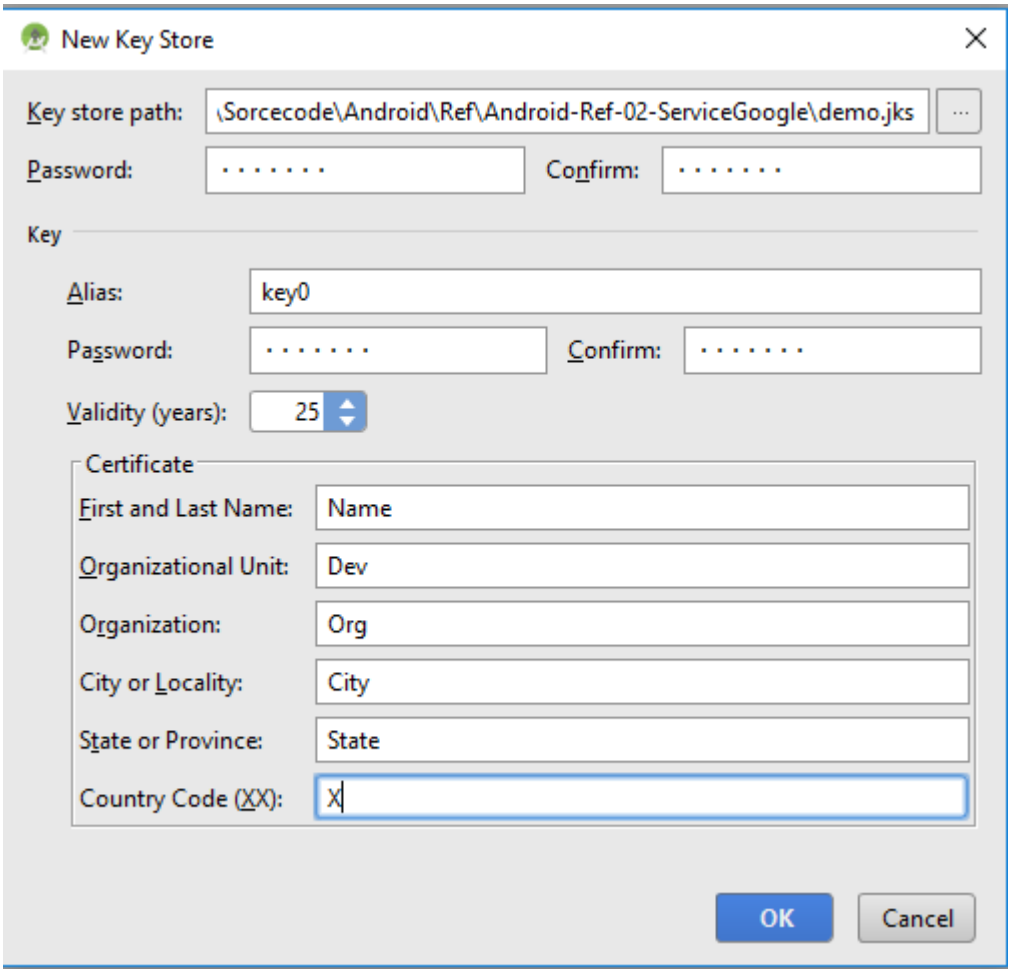
き

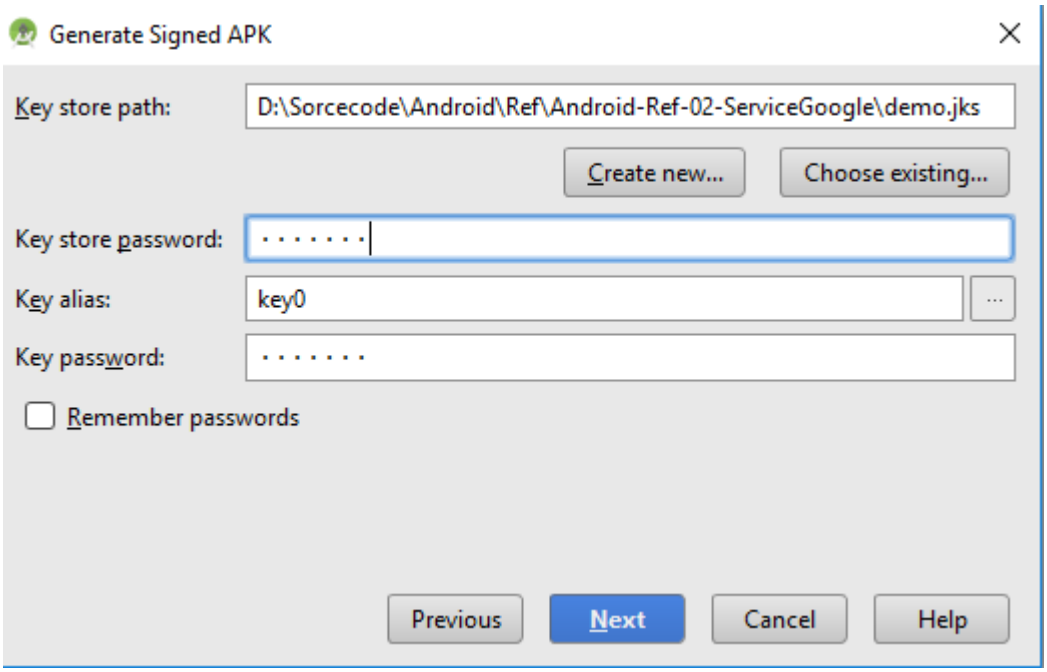
Androidでは、すべてのAPKにリリースのがです。

Examples

あなたのアプリにする

1. メニューバーで、[ビルド]> [きAPKを]をクリックします。
2. リリースしたいモジュールをドロップダウンからし、「へ」をクリックします。
3. しいキーストアをするには、「」をクリックします。なをし、[New Key Store]で[OK]をします。





4. しいキーストアをしたばかりのは、[Signed APKウィザード]フィールドにはすでにがされています。それのは、[へ]をクリックします。

5. のウィンドウで、みのAPKのをし、ビルドタイプをして[]をクリックします。

コンフィグレーションで**build.gradle**をコンフィグレーションする

をして、 `build.gradle`ファイルのapkにすることができます。

できるはのとおりです。

- `storeFile` キーストアファイル
- `storePassword` キーストアのパスワード
- `keyAlias` キーエイリアス
- `keyPassword` キーエイリアスのパスワード

をするには、 `signingConfigs`ブロックをするがあります。

```
android {
    signingConfigs {

        myConfig {
            storeFile file("myFile.keystore")
            storePassword "xxxx"
            keyAlias "xxxx"
            keyPassword "xxxx"
        }
    }
    //....
}
```

に、1つのビルドタイプにりてることができます。

```
android {

    buildTypes {
        release {
            signingConfig signingConfigs.myConfig
        }
    }
}
```

オンラインでリリースのためにAndroidアプリにするをむ
https://riptutorial.com/ja/android/topic/9721/リリースのためにandroidアプリにする

231: ルーパー

き

`Looper`はスレッドにメッセージループをするためにされるAndroidクラスです。、スレッドにけられているスレッドはありません。

Androidでもな`Looper`はメインループで、にメインスレッドともばれます。このインスタンスはアプリケーションのもので、 `Looper.getMainLooper()`を`Looper.getMainLooper()`にアクセスできます。

`Looper`がのスレッドにけられているは、 `Looper.myLooper()`してできます。

Examples

シンプルな**LooperThread**をする

ドキュメントでされる`Looper`スレッドののなは、 `Looper.prepare()`と`Looper.loop()`をし、これらのびしののループに`Handler`をけます。

```
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();
    }
}
```

HandlerThreadでループをする

`HandlerThread`をして、 `Looper`スレッドをすることができます。このループをして、それとするための`Handler`をすることができます。

```
HandlerThread thread = new HandlerThread("thread-name");
thread.start();
Handler handler = new Handler(thread.getLooper());
```

オンラインでルーパーをむ https://riptutorial.com/ja/android/topic/10593/ルーパー

232: レイアウト

き

レイアウトは、アクティビティやウィジェットなどのユーザインタフェースのビジュアルをします。

レイアウトはXMLでされ、そこにされるスクリーンもまれます。コードをアプリケーションにして、にXMLでされたオブジェクトをむスクリーンオブジェクトのをすることができます。

- アンドロイド=「」||||||りつぶし|||りつぶし||りつぶし|クリップ_|クリップ|
- アンドロイドlayout_gravity = "top | bottom | left | right | center_vertical | fill_vertical | center_horizontal | fill_horizontal | center | fill | clip_vertical | clip_horizontal | start | end"

LayoutParamsおよびLayout_ Attributes

Layout Attributes

+CoordinatorLayout

layout_behavior

+FrameLayout

layout_gravity

+LinearLayout

layout_weight

+RelativeLayout

layout_above layout_below

layout_alignLeft/Top/Right/Bottom

layout_alignParentLeft/etc

layout_toLeftOf/etc

layout_alignBaseline

layout_centerInParent

+AbsoluteLayout NO

Please don't

Linear



orientation="horiz

vs

orientation="vert



ーマンスの

Androidでのパフォーマンスにするこのでしたように、 `RelativeLayout`では、にレンダリングするには2つのレイアウトパスがです。なビューの、これはパフォーマンスになをぼします。 `RelativeLayouts`をネストすると、このはさらにします。すべての`RelativeLayout`によってレイアウトパスのがえるためです。

Examples

LinearLayout

LinearLayoutは、そのを1または1にする`ViewGroup`です。は、メソッド`setOrientation()`びすか、xml`android:orientation`をしてできます。

1. `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>
```

これはどのようにえるかをスクリーンショットでします



2. `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`は、 `android:layout_weight`をして々のにウェイトをりてることもサポートしています。

RelativeLayout

`RelativeLayout`は、ビューをにする`ViewGroup`です。デフォルトでは、すべてのビューはレイアウト

のにされるため、 RelativeLayout.LayoutParamsからなさまざまなレイアウトプロパティをして、ビュー―のをするがあります。レイアウトプロパティのは、のRelativeLayoutをにしたレイアウトのをにするブ―ルか、レイアウトのビュー―をするのビュー―をするIDです。

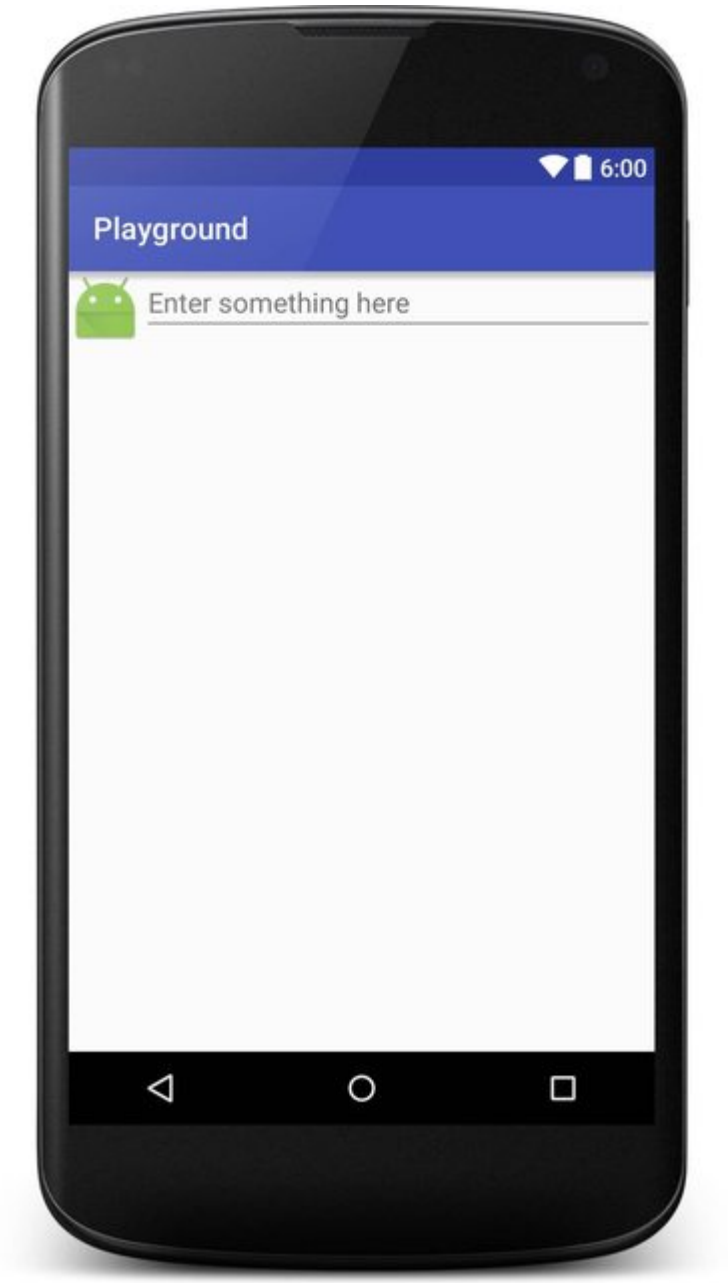
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <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>
```

これはどのようにえるかをスクリ―ンショットでします



とレイアウトの

androidlayout_gravity

- `android:layout_gravity`は、ののをするためにされ`View Layout`の`View`。
- LinearLayoutとFrameLayoutでサポート

アンドロイド

- `android:gravity`は、のコンテンツのをするためにされます `TextView`のテキストなど。

```
<?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"/>

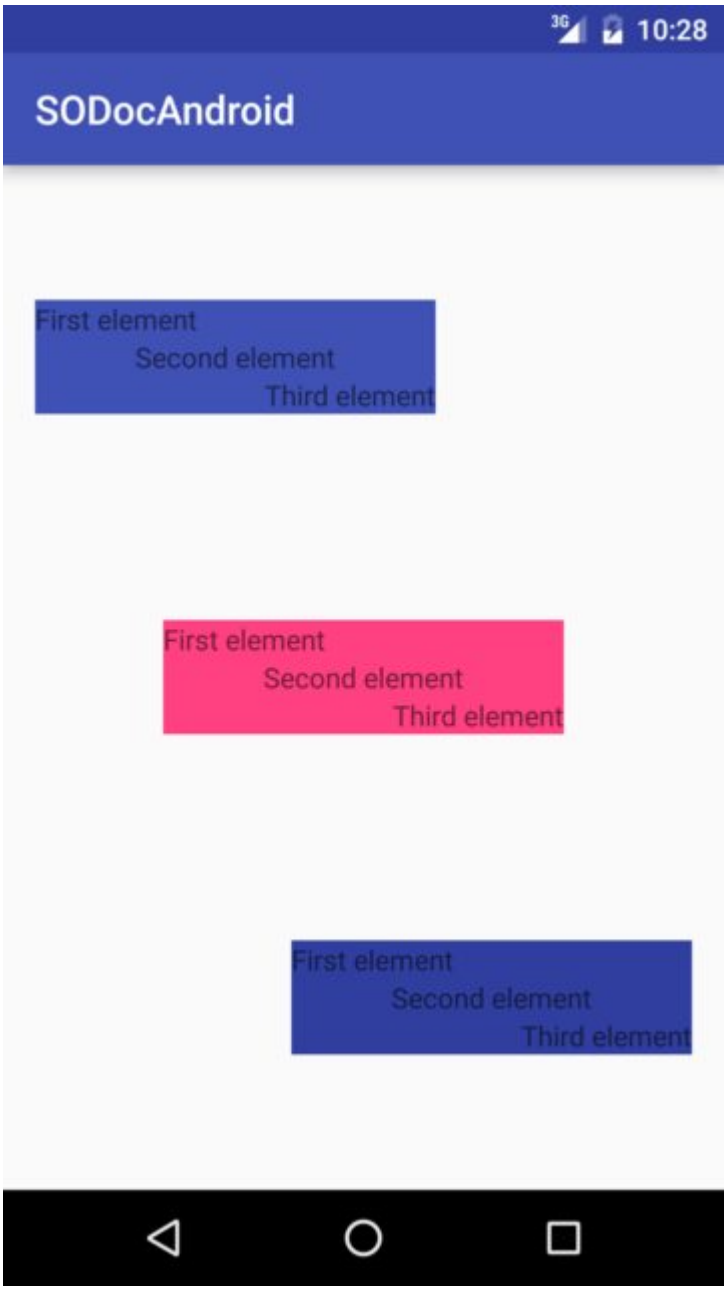
        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/second"
            android:background="@color/colorPrimaryDark"
            android:gravity="center"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/third"
            android:background="@color/colorPrimaryDark"
            android:gravity="right"/>

    </LinearLayout>

</LinearLayout>
```

のようにレンダリングされます



グリッドレイアウト

GridLayoutは、がするように、ビューをグリッドにするためにされるレイアウトです。GridLayoutは、それをとにします。のでわかるように、および/ `columnCount`のは、プロパティ`columnCount`および`rowCount`によってされ`rowCount`。このレイアウトにビューをすると、のがのに、2のが2のに、3のが2のにされます。

```
<?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"
```

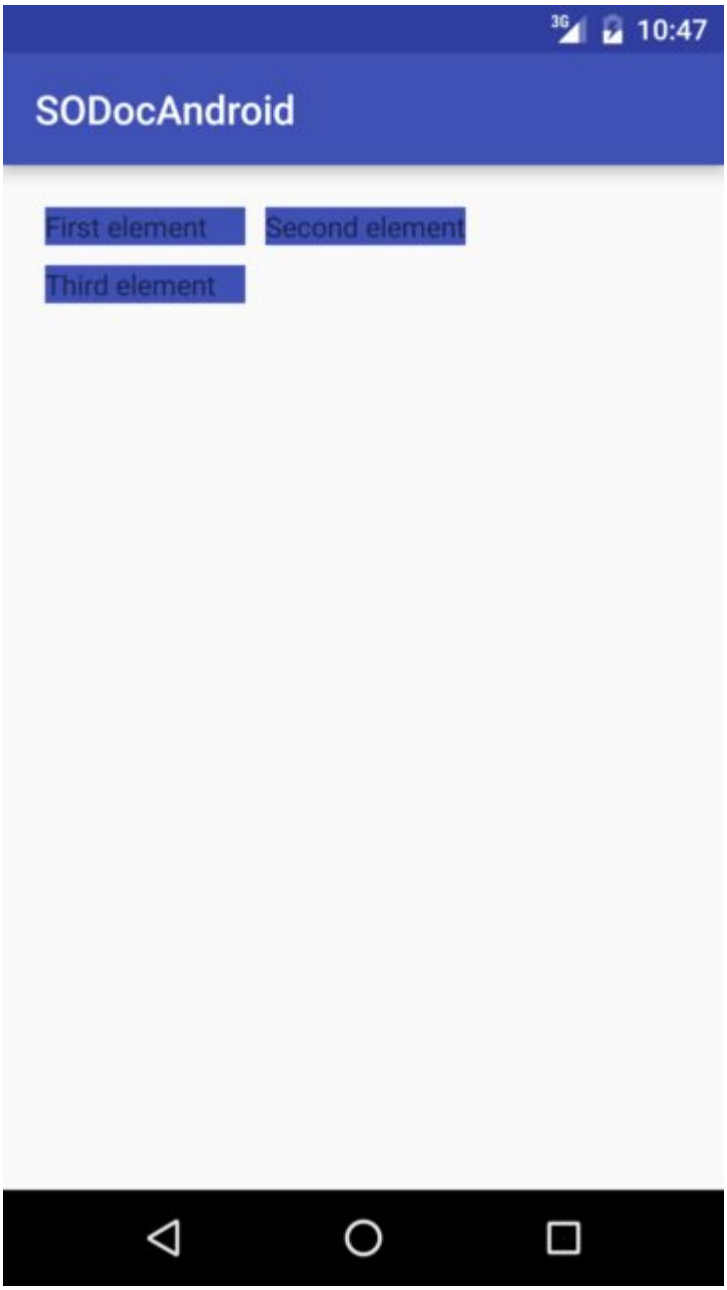
```
    android:rowCount="2">

    <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>
```



パーセンテージレイアウト

2.3

パーセントサポートライブラリが`PercentFrameLayout`と`PercentRelativeLayout`、なサイズのパーセンテージでとをするなをする2つのViewGroupsを。

Percent Support Libraryをするには、にをします。

```
compile 'com.android.support:percent:25.3.1'
```

をにりつぶし、のしかにしないようにするには、のようにします。

```
<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>
```

のようなコードをして、パーセンテージを々のXMLファイルにすることもできます。

```
<fraction name="margin_start_percent">25%</fraction>
```

`@fraction/margin_start_percent`ってそれらのレイアウトをしてください。

また、 `app:layout_aspectRatio`をしてカスタムアスペクトをするもまれています。
これにより、のみなどののだけをすることができ、43または169、またはの11のいずれであっても、したアスペクトにづいてにさがされますアスペクト。

えば

```
 <ImageView
    app:layout_widthPercent="100%"
    app:layout_aspectRatio="178%"
    android:scaleType="centerCrop"
    android:src="@drawable/header_background"/>
```

FrameLayout

`FrameLayout`は、をするためにのをブロックするようにされています。ただし、FrameLayoutにのをし、 androidlayout_gravityをしてにをりてることで、FrameLayoutのをできます。

、 `FrameLayout`は1つのビューをするためにされます。なでは、 `Activity`で`Fragments`をさせたり、ビューをオーバーラップさせたり、ビューにフォアグラウンドをしたりするためのプレースホルダをしています。

```
<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>
```

それはのようになります



コーディネーターのレイアウト

2.3

CoordinatorLayoutは、 FrameLayoutていFrameLayoutが、なをえたコンテナです。ドキュメントでは、Super-powered FrameLayoutとばれています。

CoordinatorLayout.BehaviorをCoordinatorLayout.Behaviorののにアタッチすることで、タッチイベント、ウィンドウインセット、、レイアウト、およびネストされたスクロールをインターセプトできます。

これをするには、まず、サポートライブラリのをgradleファイルにするがあります。

```
compile 'com.android.support:design:25.3.1'
```

のライブラリのはここにあります

CoordinatorLayoutなは、 FloatingActionButtonをしてビューをすることです。こののケースでは、SwipeRefreshLayoutとそのにFloatingActionButtonをつRecyclerViewをします。これをうはのとおりです。

```
<?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>
```

FloatingActionButtonがapp:layout_anchor="@id/coord_layout"してCoordinatorLayoutにされていることにapp:layout_anchor="@id/coord_layout"

コーディネータのレイアウトスクロール

2.3-2.3.2

ネストされたスクロールをサポートするレイアウト NestedScrollViewやRecyclerViewなどをNestedScrollView、 NestedScrollView CoordinatorLayoutをしてマテリアルデザインのスクロールエフェクトをできます。

このの

- `app:layout_scrollFlags="scroll|enterAlways"`はツールバーのプロパティでされます
- `app:layout_behavior="@string/appbar_scrolling_view_behavior"`がViewPagerのプロパティでされています
- RecyclerViewは、ViewPagerのフラグメントでされます

アクティビティでされるレイアウトXMLファイルはのとおりです。

```
<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">

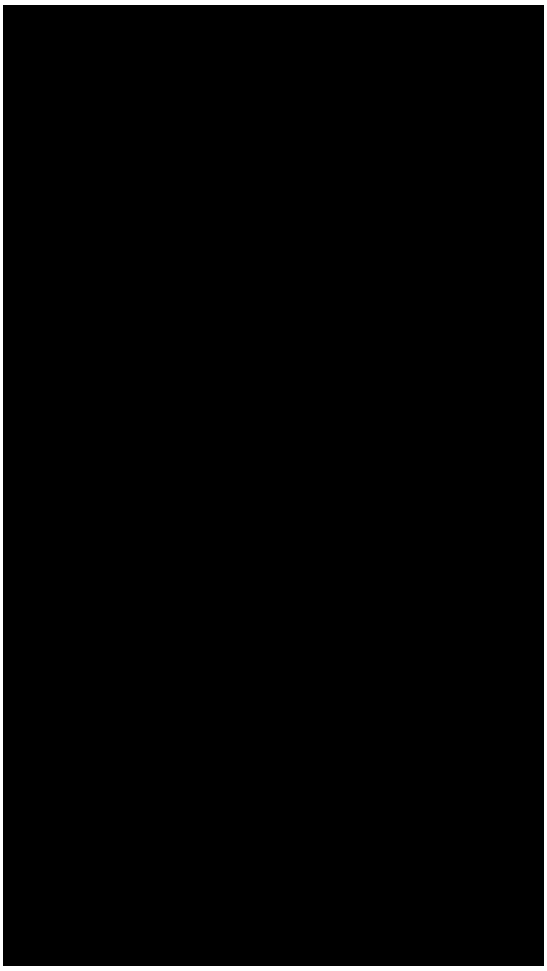

    <android.support.design.widget.AppBarLayout
        android:id="@+id/appBarLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:elevation="6dp">
    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        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>
```



を

LinearLayoutでもよくされるの1つは、そのビュー—のみです。 Weightは、LinearLayoutののビュー—として、ビュー—がするスペ—スのきさをします。

みは、のコンポ—ネントとしてあるコンポ—ネントにのをえたいにされます。

キ—のプロパティ

- weightSumは、すべてのビュー—のみのです。 weightSumしないと、システムはすべてのみのをweightSumでします。
- layout_weightは、ウィジェットがするのうちのスペ—スのをします。

コ—ド

```
<?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">
```

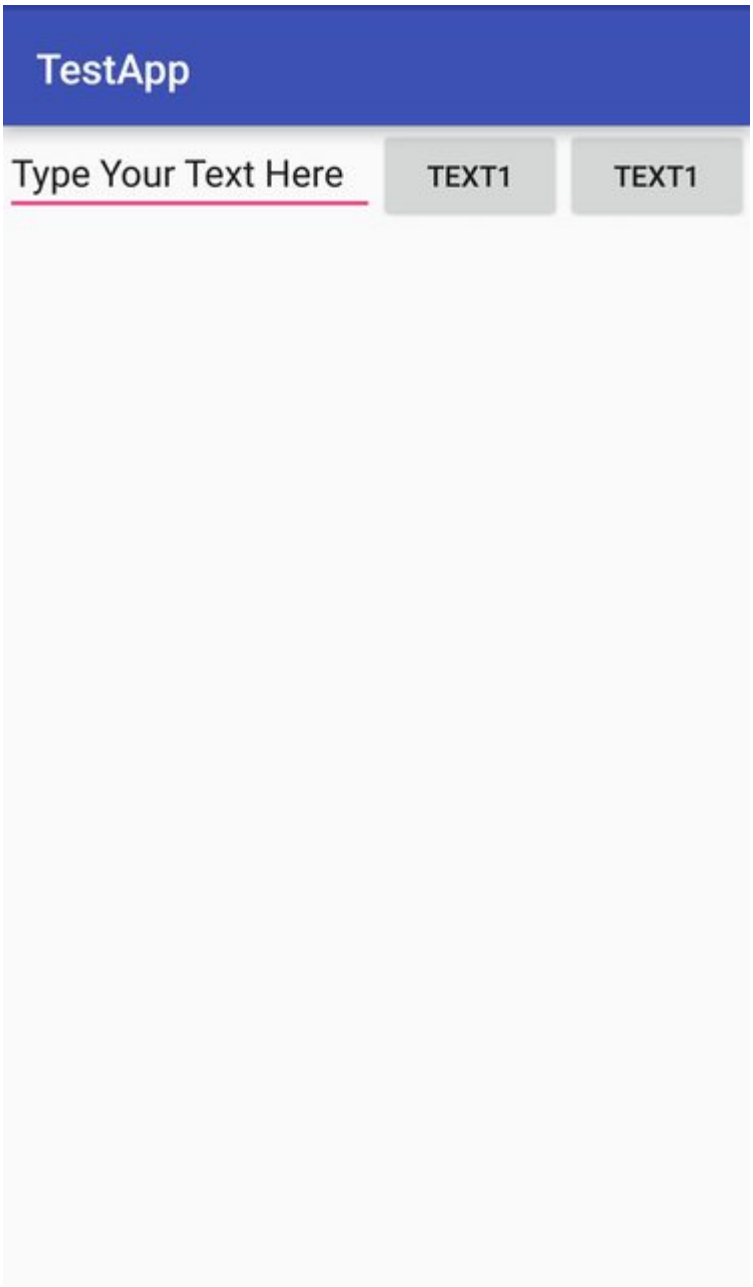
```
    <EditText
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Type Your Text Here" />

    <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>
```

はのとおりです。



、デバイスのサイズがきくても、EditTextはのスペースの2/4をめます。したがって、あなたのアプリのはすべてのでしてえます。

layout_widthは、ウィジェットスペースがにされている0dp 、 layout_widthまま0dp。ウィジェットをにする、 layout_heightはlayout_heightにされ0dp。これはにシステムがによってされるので、システムはまたはさをそれぞれしないため、コードのをめるためにわれます。わりにwrap_contentをすると、システムはweightをするにwidth / heightをにして、のサイクルをきこします。

プログラムによるLinearLayoutの

```
- LinearLayout(horizontal)
   - ImageView
   - LinearLayout(vertical)
     - TextView
     - TextView
```

コード

```
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

すべてののViewGroup たとえば、 LinearLayout 、 RelativeLayout 、 CoordinatorLayoutなどは、そのプロパティにするをするがあります。 ViewGroupそのがされるについて。このは、ラッパークラスViewGroup.LayoutParamsオブジェクトにされます。

のレイアウトタイプにのパラメータをめるには、 ViewGroupsはViewGroup.LayoutParamsクラスのサブクラスをします。

えば

- LinearLayout LinearLayout.LayoutParams
- RelativeLayout RelativeLayout.LayoutParams
- CoordinatorLayoutはCoordinatorLayout.LayoutParams
- ...

ほとんどのViewGroupsはのmarginsをするをしているため、 ViewGroup.LayoutParamsサブクラスViewGroup.LayoutParamsせず、わりにViewGroup.MarginLayoutParamsをサブクラスViewGroup.MarginLayoutParamsます ViewGroup.LayoutParamsサブクラス。

LayoutParams in xml

LayoutParamsオブジェクトは、したレイアウトxmlファイルにづいてされます。

```
<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>
```

layout_でまるすべてのパラメータは、みレイアウトがどのようにlayout_するかをします。レイアウトがすると、これらのパラメータはなLayoutParamsオブジェクトにラップされ、でLayoutによってViewGroupののViewをにするためにされます。 ViewのはViewし、 ViewによってされViewる。

TextView

- layout_width、 layout_heightとlayout_gravityにされるLinearLayout.LayoutParamsオブジェクトによってされるLinearLayout
- gravity、 text、およびtextColorはTextViewでされます

ImageView

- layout_width、 layout_heightとlayout_weightにされるLinearLayout.LayoutParamsオブジェクトによってされるLinearLayout
- background、 scaleType、 srcはImageViewでされます

LayoutParamsオブジェクトの

getLayoutParamsは、のLayoutParamsオブジェクトをすることをにするView'sメソッドです。

LayoutParamsオブジェクトは、むViewGroupしているため、 LayoutParamsにViewがけられているにのみ、このメソッドはnullのをViewGroupます。このオブジェクトはにしないかもしれないことにをつけておくがあります。に、 View'sコンストラクタView'sにそれをくことにるべきではありません。

```
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();
        }
    }

    //...
}
```

LayoutParamsオブジェクトをつことにしたいは、わりにonAttachedToWindowメソッドをするがあります。

```
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();
        }
    }

    //...
}
```

キャスト**LayoutParams**オブジェクト

のViewGroupのをするがあるかもしれませんたとえば、 RelativeLayoutルールをプログラムでしたいなど。そのためには、 ViewGroup.LayoutParamsオブジェクトをしくキャストするをるがあります。

にのViewGroupであるView LayoutParamsオブジェクトをすると、これはちょっとするかもしれません

。

```
<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>
```

の`LayoutParams`オブジェクトをむのにしている`ViewGroup`。

なキャスティング

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
FrameLayout.LayoutParams par = (FrameLayout.LayoutParams) innerLayout.getLayoutParams();
                                        // INCORRECT! This will produce ClassCastException
```

しいキャスティング

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
LinearLayout.LayoutParams par = (LinearLayout.LayoutParams) innerLayout.getLayoutParams();
                                       // CORRECT! the enclosing layout is a LinearLayout
```

オンラインでレイアウトをむ https://riptutorial.com/ja/android/topic/94/レイアウト

233: レトロフィット2

き

のRetrofitのページでは、

Androidおよび***Java***のされた***REST***クライアント。

Retrofitをすると、REST APIがJavaインターフェイスにわります。をしてHTTPをし、URLパラメータのとクエリパラメータのサポートはデフォルトでされています。さらに、マルチパートリクエストおよびファイルアップロードのためのをします。

ライブラリの

から

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

シンプルな**GET**リクエスト

JSONオブジェクトまたはJSONでするAPIにしてGETリクエストをうをします。にうことは、モジュールのGSONファイルにGSONとGSON Converterのをすることです。

「」セクションでされているように、レトロフィットライブラリのをします。

される**JSON**オブジェクトの

```
{
    "deviceId": "56V56C14SF5B4SF",
    "name": "Steven",
```

```
    "eventCount": 0
}
```

JSONの

```
[
    {
        "deviceId": "56V56C14SF5B4SF",
        "name": "Steven",
        "eventCount": 0
    },
    {
        "deviceId": "35A80SF3QDV7M9F",
        "name": "John",
        "eventCount": 2
    }
]
```

するモデルクラスの

```
public class Device
{
    @SerializedName("deviceId")
    public String id;

    @SerializedName("name")
    public String name;

    @SerializedName("eventCount")
    public int eventCount;
}
```

@SerializedNameここでのからです`GSON`ライブラリとたちがすることができます`serialize`と`deserialize`にこのクラスを`JSON`キーとしてシリアルされたをして。これで、にサーバーからデータをフェッチするAPIのインターフェイスをできます。

```
public interface DeviceAPI
{
    @GET("device/{deviceId}")
    Call<Device> getDevice (@Path("deviceId") String deviceID);

    @GET("devices")
    Call<List<Device>> getDevices();
}
```

かなりコンパクトなにたくさんありますので、それをしましょう

- `@GET`は`@GET`からされ、GETリクエストをしていることをライブラリにえます。
- カッコのパスは、GETリクエストがヒットするエンドポイントですベースURLをしにします。
- はにパスのをきえてをすことができます。
- しているは`getDevice`とばれ、なデバイスIDをとしてとります。
- `@PATH`アノテーションは、このがパスの "deviceId"プレースホルダをきえるがあることを

Retrofitにします。
- このは、 `Device`の`Call`オブジェクトをします。

ラッパークラスの

は、Retrofitコードをうまくまとめたままにするために、APIのしラッパークラスをします。

```
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);
    }
}
```

このクラスは、GSONインスタンスをしてJSONレスポンスをし、ベースURLとGSONConverterでRetrofitインスタンスをし、APIのインスタンスをします。

APIのびし

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

これは、々のAPIインターフェースをして`Call<Device>`オブジェクトをし、それぞれ`Call<List<Device>>`をします。 `enqueue`びすと`enqueue`はバックグラウンドスレッドでそのびしをい、ここでしているコールバックにをします。

プリミティブオブジェクト *String*、Integer、Boolean 、 *Double*など のJSONのは、JSONのとています。しかし、のモデルクラスはありません。 `Call<List<String>>`りのを`Call<List<String>>`としてつなど、 `Call<List<String>>`のをできます。

Retrofit2にロ グをする

は、インタセプターをしてできます。なレベルは、NONE、BASIC、HEADERS、BODYです。ここのGithubプロジェクトをしてください。

1. build.gradleにをする

```
compile 'com.squareup.okhttp3:logging-interceptor:3.8.1'
```

2. Retrofitのにロギングインターセプタをする

```
  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();
```

ターミナルAndroidモニターにログをすることは、リリースではけておくべきことです。これは、Authトークンなどのながされるがあるためです。

にログがされないようにするには、のをしてください

```
if(BuildConfig.DEBUG){
     //your interfector code here
    }
```

えば

```
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();
```

マルチパートでファイルをアップロードする

Retrofit2アノテーションをしてインタフェースをします。

```
public interface BackendApiClient {
    @Multipart
    @POST("/uploadFile")
    Call<RestApiDefaultResponse> uploadPhoto(@Part("file\"; filename=\"photo.jpg\" ")
RequestBody photo);
}
```

`RestApiDefaultResponse`はレスポンスをむカスタムクラスです。

APIのをし、びしをエンキューします。

```
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>);
```

OkHttpインターセプタをした

このは、リクエストインターセプタをOkHttpとともにするをしています。これには、のようなくのユースケースがあります。

- に`header`をする。えばをする
- ネットワークアプリケーションのデバッグ
- の`response`する
- ネットワークのロギング

- カスタムユーザーエージェントをする

```
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);
```

については、 OkHttpトピックをしてください。

ヘッダとの

@Headerアノテーションと@Bodyアノテーションはメソッドシグネチャにでき、 @Headerモデルについてアノテーションをにします。

```
public interface MyService {
     @POST("authentication/user")
     Call<AuthenticationResponse> authenticateUser(@Body AuthenticationRequest request,
@Header("Authorization") String basicToken);
}
```

AuthenticaionRequestは、サーバーがとするをむたちのモデル、POJOです。このでは、サーバーはクライアントのとをしています。

```
public class AuthenticationRequest {
     String clientKey;
     String clientSecret;
}
```

@Header("Authorization") 、Authorizationヘッダーをすることをしています。 Retrofitは、しているオブジェクトのについてレトロフィットができるため、にのヘッダーがされます。

どこかでRetrofitサービスをします。 HTTPSをしていることをします。

```
Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https:// some example site")
            .client(client)
```

```
            .build();
MyService myService = retrofit.create(MyService.class)
```

その、たちのサービスをすることができます。

```
AuthenticationRequest request = new AuthenticationRequest();
request.setClientKey(getClientKey());
request.setClientSecret(getClientSecret());
String basicToken = "Basic " + token;
myService.authenticateUser(request, basicToken);
```

Retrofitをマルチパートとしてしてのファイルをアップロードする

プロジェクトにRetrofitをセットアップしたら、Retrofitをしてのファイルをアップロードするをするのをできます。

```
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;
}
```

はインター―フェイスです

```
public interface WebAPIService {
    @Multipart
    @POST("main.php")
    Call<String> postFile(@PartMap Map<String,RequestBody> Files, @Part("json") String
description);
}
```

Retrofit2をしてサ―バ―からファイルをダウンロ―ドする

ファイルをダウンロ―ドするためのインタフェ―ス

```
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);
}
```

オプション1は、URLをつサ―バ―からファイルをダウンロ―ドするためにされます。オプション2をしてURLをなURLとしてびしにします。これは、ユ―ザ―やなどのパラメ―タにするファイルをダウンロ―ドするときにちます。

apiびしのためのセットアップの

```
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);
    }

}
```

、サーバーからファイルをダウンロードするためのAPIをする

```
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) {

            }
        });
    }
```

そして、コールバックでをた、ファイルをディスクにするためのIOをコーディングします。ここにコードです

```
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;
        }
    }
```

ResponseBodyをりのとしてしたことにしてください。そうしないと、Retrofitはしてしようとしますが、これはファイルのダウンロードにはがありません。

あなたがRetrofitのものをもっととするなら、になので、このリンクをにれました。 [1] https://futurestud.io/blog/retrofit-getting-started-and-android-client

Stethoによるデバッグ

アプリケーションにのをします。

```
compile 'com.facebook.stetho:stetho:1.5.0'
compile 'com.facebook.stetho:stetho-okhttp3:1.5.0'
```

Applicationクラスの`onCreate`メソッドで、をびします。

```
Stetho.initializeWithDefaults(this);
```

あなたのするときに`Retrofit`インスタンスを、カスタムOkHttpインスタンスをします。

```
OkHttpClient.Builder clientBuilder = new OkHttpClient.Builder();
clientBuilder.addNetworkInterceptor(new StethoInterceptor());
```

に、このカスタムOkHttpインスタンスをRetrofitインスタンスにします。

```
Retrofit retrofit = new Retrofit.Builder()
    // ...
    .client(clientBuilder.build())
    .build();
```

すぐおいのをコンピュータにし、アプリを`chrome://inspect`て、Chromeブラウザに「`chrome://inspect`とします。あなたがするためのネットワークのがすぐされるはずです。

2つのカスタムXmlコンバータを

build.gradleファイルへのの

```
dependencies {
    ....
    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile ('com.thoughtworks.xstream:xstream:1.4.7') {
        exclude group: 'xmlpull', module: 'xmlpull'
    }
    ....
}
```

その、Converter Factoryをします。

```
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);
    }
}
```

をするクラスをします。

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

ボディレスポンスをするクラスをします。

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

ですから、このでXMLをできます。エンティティにしてXStreamアノテーションをするだけでみます。

に、Retrofitインスタンスをします。

```
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();
```

GSONをったなPOSTリクエスト

サンプルJSON

```
{
    "id": "12345",
    "type": "android"
}
```

あなたのをする

```
public class GetDeviceRequest {

    @SerializedName("deviceId")
    private String mDeviceId;

    public GetDeviceRequest(String deviceId) {
        this.mDeviceId = deviceId;
    }

    public String getDeviceId() {
        return mDeviceId;
    }

}
```

サービスをするヒットするエンドポイント

```
public interface Service {

    @POST("device")
    Call<Device> getDevice(@Body GetDeviceRequest getDeviceRequest);

}
```

ネットワークライアントのシングルトンインスタンスをします。

```
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;
    }

}
```

デバイスのなモデルオブジェクトをする

```
public class Device {

    @SerializedName("id")
    private String mId;

    @SerializedName("type")
    private String mType;

    public String getId() {
        return mId;
    }

    public String getType() {
        return mType;
    }

}
```

デバイスのをするコントローラをする

```
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
            }

        });
```

Retrofit 2でXMLフォームのURLをむ

レトロフィット2とSimpleXmlConverterをして、urlからxmlデータをし、Javaクラスにします。

をGradleスクリプトにする

```
compile 'com.squareup.retrofit2:retrofit:2.1.0'
compile 'com.squareup.retrofit2:converter-simplexml:2.1.0'
```

インターフェイスの

また、たちのケースのRssクラスにxmlクラスラッパーをする

```
    public interface ApiDataInterface{

        // path to xml link on web site

        @GET (data/read.xml)

        Call<Rss> getData();

}
```

XMLみみ

```
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());
        }


    }
```

これはSimpleXMLアノテーションをつJavaクラスのです

の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;
```

オンラインでレトロフィット2をむ https://riptutorial.com/ja/android/topic/1132/レトロフィット2

234: レルム

き

Realm Mobile DatabaseはSQLiteのです。 Realm Mobile Databaseは、ORMよりもはるかにで、raw SQLiteよりもです。

オフライン、クエリ、なスレッド、クロスプラットフォームアプリケーション、、リアクティブアーキテクチャ。

レルムをするときは、スレッドでRealmObjects、RealmResults、およびRealmインスタンスをさないでください。のスレッドでクエリがなは、そのスレッドでRealmインスタンスをきます。スレッドのに、レルムをじるがあります。

> の ソフトウエアにはのとなるがまれていることをしており、 キュバ、イラン、をむのまたはにしていないことをします スダン、シリア、クリミアで 、された、の、またはきとしているのリストにっていないことをしてください。

Examples

レルムをプロジェクトにする

プロジェクトレベルの`build.gradle`ファイルにのをします。

```
dependencies {
    classpath "io.realm:realm-gradle-plugin:3.1.2"
}
```

アプリレベルの`build.gradle`ファイルのにのをします。

```
apply plugin: 'realm-android'
```

gradle syncをすると、レルムがプロジェクトにするようになりました

Realmは2.0.0のびしがです。これは、 `Application`クラスまたはのアクティビティの`onCreate`メソッドでうことができます。

```
Realm.init(this); // added in Realm 2.0.0
Realm.setDefaultConfiguration(new RealmConfiguration.Builder().build());
```

レルムモデル

レルムモデルは`RealmObject`ベースクラスをするがあり、となるデータベースのスキーマをします。

サポートされるフィールドタイプは、 `boolean` 、 `byte` 、 `short` 、 `int` 、 `long` 、 `float` 、 `double` 、 `String` 、 `Date` 、 `byte[]` 、のRealmObjectへのリンク、 `RealmList<T extends RealmModel>`です。

```
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
}
```

RealmObjectにしいフィールドをまたはしたり、しいRealmObjectクラスをしたり、のフィールドをすると、 がになります。あなたはどちらかにすることができます`deleteIfMigrationNeeded()`あなたに`RealmConfiguration.Builder` 、またはにじてをします。 `@Required` 、 `@Index` 、または`@PrimaryKey`アノテーションをまたはするときにもがです。

リレーションシップはでするがあり、キーについてではありません。

0.88.0、RealmObjectクラスのプライベートフィールド/ゲッター/セッターのわりにパブリックフィールドをすることもできます。

クラスに`@RealmClass`もがけられている、 `RealmObject`をするわりに`RealmModel`をすることもできます。

```
@RealmClass
public class Person implements RealmModel {
    // ...
}
```

この、 `person.deleteFromRealm()`や`person.addChangeListener()`などのメソッドは、`RealmObject.deleteFromRealm(person)`および`RealmObject.addChangeListener(person)`きえられます。

`RealmObject`では、 `RealmObject`のみを`RealmObject`することができ、 `final`フィールド、 `volatile` `transient`フィールドおよび`transient`フィールドはサポートされていないというがあります。

の RealmObjectクラスは、トランザクションでしかできないことがです。 マネージ RealmObject はスレッドですことはできません。

プリミティブのリスト**RealmList**

レルムは、プリミティブのリストのをサポートしていません。それはそのToDoリスト GitHub issue575 にありますが、そのにがあります。

プリミティブのしいクラスをします。これはIntegerをしますが、するものはします。

```
public class RealmInteger extends RealmObject {
    private int val;

    public RealmInteger() {
    }

    public RealmInteger(int val) {
        this.val = val;
    }

    // Getters and setters
}
```

これを`RealmObject`できるように`RealmObject`。

```
public class MainObject extends RealmObject {

    private String name;
    private RealmList<RealmInteger> ints;

    // Getters and setters
}
```

しているは`GSON`ごする`RealmObject`、カスタムタイプのアダプタをするがあります。

```
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();
```

リソースをしてみる

```
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
}
```

リソースは、KITKATminSDK 19からのみできます。

べえられたクエリ

クエリをソートするには、 `findAll()`をするわりに、 `findAllSorted()`をするがあります。

```
RealmResults<SomeObject> results = realm.where(SomeObject.class)
                                        .findAllSorted("sortField", Sort.ASCENDING);
```

`sort()`は、 **`sort()`**たまったくしいRealmResultsをしますが、このRealmResultsのによってリセットされます。あなたがしているは`sort()`あなたはにあなたにそれをソートしすがあります`RealmChangeListener`、 `RealmChangeListener`から`RealmResults`とされたしいに`RealmResults`。 まだロードされていないクエリによってされた`RealmResults` `sort()`をするとします。

`findAllSorted()`はされても、フィールドでソートされたをにします。 `findAllSorted()`をすることをおめします。

クエリ

すべてのクエリメソッド `findAll()`や`findAllSorted()` には、の `findAllAsync()` / `findAllSortedAsync()` があります。

クエリは、 `RealmResults`のをのスレッドに`RealmResults`します。のスレッドでこれらのをけるには、のスレッドはルーパースレッドでなければなりませんreadクエリはUIスレッドでのみします。

```
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);
```

RxJavaでレルムをする

クエリーの、Realmは`realmResults.asObservable()`メソッドをします。は、ルーパースレッドはUIスレッドでのみです。

これをするには、にのものがまれているがあります

```
realmConfiguration = new RealmConfiguration.Builder(context)       //
                         .rxFactory(new RealmObservableFactory()) //
                            //...
                         .build();
```

その、をなものとしてすることができます。

```
Observable<RealmResults<SomeObject>> observable = results.asObservable();
```

クエリのは、クエリがされたときにのみイベントをけるように、 `isLoaded()`でをフィルタリングするがあります。この`filter()`はクエリにはありません `isLoaded()`にクエリで`true`をし`true`。

```
    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));
```

きみのは、 `executeTransactionAsync()`メソッドをするか、バックグラウンドスレッドでRealmインスタンスをき、トランザクションをしてしてから、Realmインスタンスをじます。

```
public Subscription loadObjectsFromNetwork() {
    return objectApi.getObjects()
        .subscribeOn(Schedulers.io())
        .subscribe(response -> {
            try(Realm realmInstance = Realm.getDefaultInstance()) {
                realmInstance.executeTransaction(realm ->
realm.insertOrUpdate(response.objects));
            }
        });
}
```

な

インスタンスをする

レルムをするには、まずそのインスタンスをするがあります。Realmインスタンスは、ディスクのファイルにマップされます。インスタンスをするもなはのとおりです。

```
// 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();
```

`Realm.getInstance()`メソッドは、されていないはデータベースファイルをし、それのはファイルをきます。 `RealmConfiguration`オブジェクトは、 `inMemory()`データベース、Realmファイルの、マイグレーションがなにレルムをクリアするがあるかどうか、データなど、レルムののすべてのをします。

`Realm.getInstance()`へのびしはカウントされびしはカウンタをインクリメントします、`realm.close()`がびされるとカウンタはデクリメントされることにしてください。

インスタンスのクローズ

バックグラウンドスレッドでは、Realmインスタンスがされなくなるとたとえば、トランザクションがしてスレッドのがするなど、Realmインスタンスをじることがにです。バックグラウンドスレッドですべてのRealmインスタンスをじることができないと、バージョンがされ、ファイルサイズがにするがあります。

```
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
```

APIレベル19では、このコードをのようにきえることができます。

```
try(Realm realm = Realm.getDefaultInstance()) {
    // ...
}
```

モデル

のステップはあなたのモデルをすることです。ここで、「モデルはですか」というがされることがあります。モデルは、データベースにされているオブジェクトのプロパティをするです。えば、ではをモデルする。

```
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;
    }
}
```

モデルがRealmObjectクラスをするがあることにしてください。キーは`@PrimaryKey`アノテーションによってもされます。キーはnullでもかまいませんが、キーとして`null`をつは1つだけです。また、ディスクにされるべきでないフィールドに`@Ignore`アノテーションをすることもできます

```
@Ignore
private String isbn;
```

データのまたは

ブックオブジェクトをRealmデータベースインスタンスにするには、まずモデルのインスタンスをしてから`copyToRealm`メソッドをしてデータベースにします。またはするには、`copyToRealmOrUpdate`をできます。 よりいはしくされた`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);
    }
});
```

データにするすべてのは、トランザクションでわれなければならないことにしてください。オブジェクトをするのは、のパターンをすることです。

```
Book book = realm.createObject(Book.class, primaryKey);
...
```

データベースのクエリ

- すべての

```
RealmResults<Book> results = realm.where(Book.class).findAll();
```

- 10のIDをつすべての

```
RealmResults<Book> results = realm.where(Book.class)
                                  .greaterThan("id", 10)
                                  .findAll();
```

- `'Taylor Swift'`や`'%Peter%'`

```
RealmResults<Book> results = realm.where(Book.class)
                                  .beginGroup()
                                      .equalTo("author", "Taylor Swift")
                                      .or()
                                      .contains("author", "Peter")
                                  .endGroup().findAll();
```

オブジェクトの

たとえば、Taylor Swiftのすべてのをしたいとします。

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

オンラインでレルムをむ https://riptutorial.com/ja/android/topic/3187/レルム

235: ローダ

き

oncreateメソッドがびされたときにバックグラウンドでデータをロードしたいは、Loaderはメモリリークをぐのにしています。たとえば、oncreateメソッドでAsynctaskをし、のAsyncTaskをするアクティビティがされるようにをさせると、おそらくAsynTaskが2つしてされます。

パラメーター

クラス	
LoaderManager	1つのLoaderインスタンスをするためのActivityまたはFragmentにけられたクラス。
LoaderManager.LoaderCallbacks	クライアントがLoaderManagerとやりとりするためのコールバックインターフェイス。
ローダ	データのみみをするクラスです。
AsyncTaskLoader	をうAsyncTaskをするローダー。
CursorLoader	ContentResolverをしてCursorをすAsyncTaskLoaderのサブクラスです。

Android 3.0でされたローダーをすると、アクティビティやフラグメントのデータをでみむことがになります。ローダーにはのがあります。

- それらはすべてのアクティビティーとフラグメントでできます。
- それらは、データのロードをします。
- らはデータのをし、コンテンツがされたときにしいをします。
- にされると、のローダーのカーソルににします。したがって、データをクエリするはありません。

ローダーをしない

バックグラウンドタスクをするがあるは、ローダーをしないでください。 Androidは、Loaderがしているactivities / Fragmentとにします。するまでしなければならないタスクをしたいは、ローダーをしないでください。わりに、こののサービスにサービスをするがあります。

Examples

AsyncTaskLoader

`AsyncTaskLoader`は、をう`AsyncTask`をする`Loader`です。

ここではいくつかのな

```
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();
    }
}
```

、 `Loader`はアクティビティの`onCreate()`メソッド、またはフラグメントの`onActivityCreated()`でされます。また、、アクティビティまたはフラグメントは`LoaderManager.LoaderCallbacks`インターフェイスをします。

```
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) {
    }
}
```

このでは、ロ―ダ―がすると、トースト―とがされます。

キャッシュき**AsyncTaskLoader**

ロ―ドされたをキャッシュして、じデ―タをロ―ドすることをけることはいです。
キャッシュをにするには`onContentChanged()`を`onContentChanged()`があります。 loaderがすでにされているは、 `forceLoad()`がびされます。そうでなければロ―ダ―がのloaderは`takeContentChanged()` checkを`takeContentChanged()`コンテンツのをできます。

*`onContentChanged()`*は、プロセスのメインスレッドからびすがあります。

*takeContentChanged*にする*Javadocs*のコメント

> ロ―ダ―のコンテンツがにされたかどうかをすのフラグをします。それがあったは trueがされ、フラグがクリアされます。

```
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
    }
}
```

リロード

いデータをにしてのロダをするには、 `restartLoader()` メソッドをします。

```
private void reload() {
    getLoaderManager().reastartLoader(LOADER_ID, Bundle.EMPTY, this);
}
```

バンドルをしてパラメタをす

Bundleでパラメータをすことができます

```
Bundle myBundle = new Bundle();
myBundle.putString(MY_KEY, myValue);
```

onCreateLoaderのをします。

```
@Override
public Loader<String> onCreateLoader(int id, final Bundle args) {
    final String myParam = args.getString(MY_KEY);
    ...
}
```

オンラインでローダをむ https://riptutorial.com/ja/android/topic/4390/ローダ

236: ログとログキャストの

- `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)`

パラメーター

オプション	
-bバッファ	イベントやラジオなど、にログバッファをみみます。メインバッファはデフォルトでされます。ログバッファのをしてください。
-c	ログをクリアフラッシュしてします。
-d	ログをにダンプしてします。
-fファイル	ログメッセージのをファイルにきみます。デフォルトはstdoutです。
-g	されたログバッファのサイズをし、します。
-nカウント	したログのをcountにします。デフォルトは4です。-rオプションがです。
-rkbytes	ログファイルをごとにさせますkbytes。デフォルトは16です。-fオプションがです。
-s	デフォルトのフィルタをにします。
-vフォーマット	ログメッセージのをします。デフォルトはなです。

Logcatは、デバイスがエラーをスローしたときのスタックトレースや、アプリケーションからLogクラスできんだメッセージなど、システムメッセージのログをダンプするコマンドラインツールです。

いつするか

JavaのSystem.outメソッドをAndroidのLogメソッドの1つではなくコンソールにすることをしているは、にじでることをするがあります。しかし、AndroidのLogメソッドがするなとフォーマットがよりであるため、Javaのメソッドをわないほうがよいでしょう。また、System.outのメソッドは`Log.i()`メソッドにリダイレクトされます。

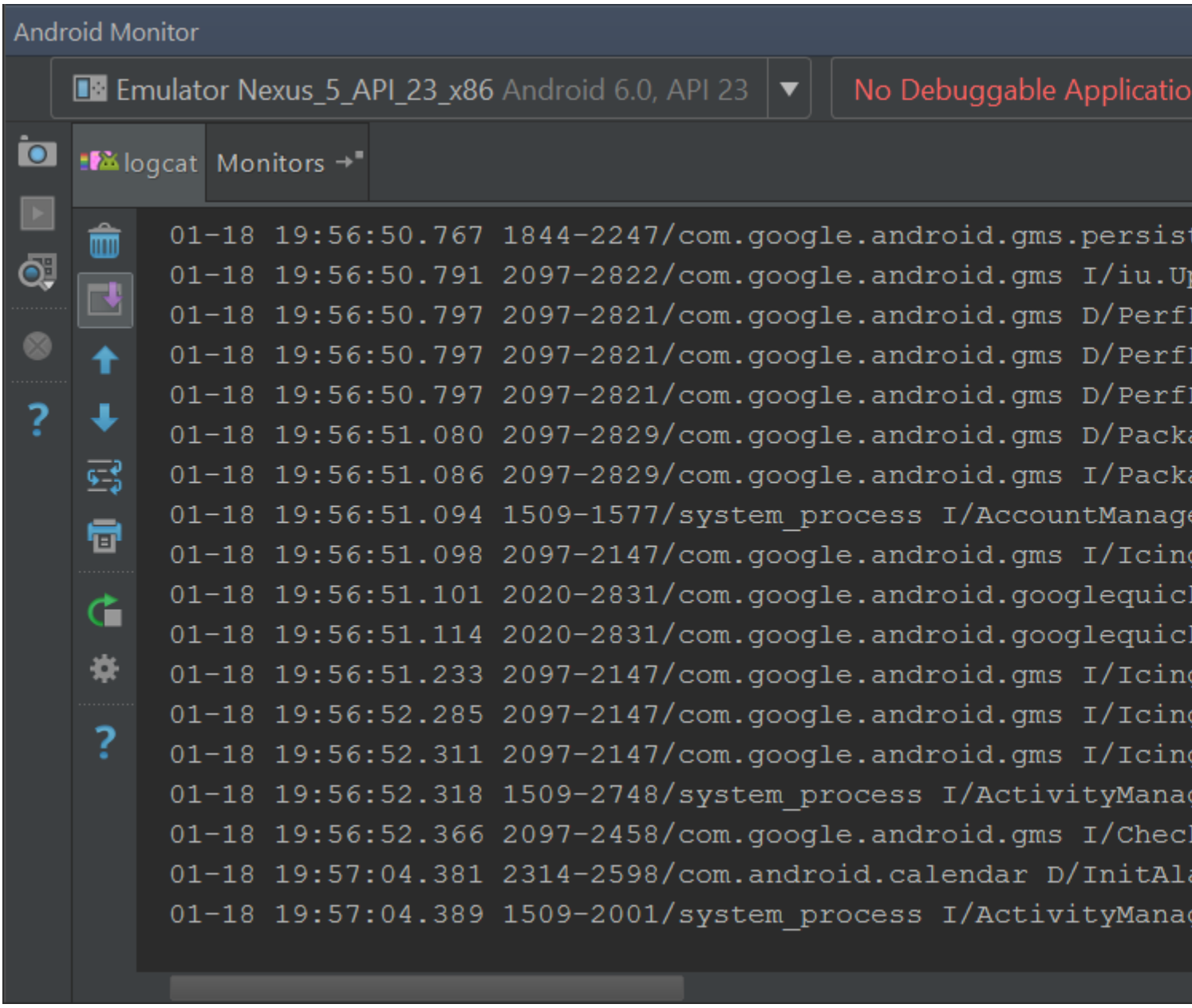
につリンク

- ログとログ キャストの Androidのドキュメント。
- Stackoveflow AndroidのLog.v、Log.d、Log.i、Log.w、Log.e - それぞれをするタイミングは

Examples

logcatのフィルタリング

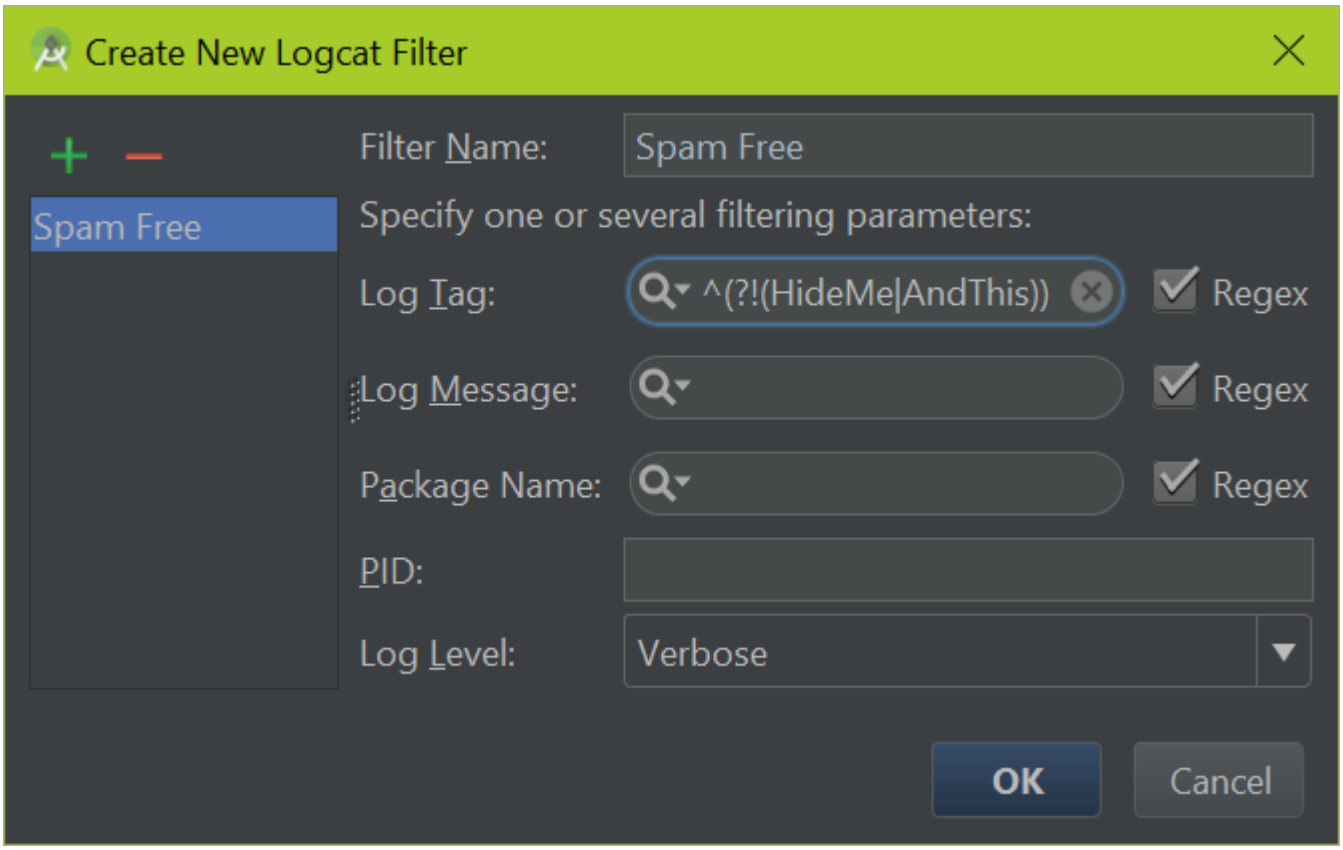
のないメッセージがたくさんあるので、logcatのをフィルタリングするとです。をフィルタリングするには、「Android Monitor」をき、のドロップダウンをクリックし、「フィルタの」をします



これで、のあるメッセージをするカスタムフィルタをするだけでなく、にできるのログをフィルタすることができます。のをするには、 をすることができます。するタグをするをにします。

```
^(?!(HideMe|AndThis))
```

これはのでできます



はをするです。 ブラックリストにのタグをするは、パイプのろにします|キャラクター。たとえば、「GC」をブラックリストにするは、のようなフィルタをします。

```
^(?!(HideMe|AndThis|GC))
```

については、 LoggingとLogcatのをしてください。

ロギング

どんなのAndroidアプリケーションでも、アプリケーションログをじてがわれているかをします。これらのログにより、はアプリケーションのをにすることができます。なAndroidドキュメントがここにありますが、はのとおりです

ログ

Logクラスは、 tagとmessageすることによって、ログをきむなソースmessage。 このタグは、ログメッセージをフィルタリングしてのアクティビティからたをするためにできるものです。にする

```
Log.v(String tag, String msg);
```

そして、Androidシステムはlogcatにメッセージをいてくれます

```
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
```

プロセスIDとスレッドIDをします。それらがじであれば - ログはメイン/ UIスレッドからています

すべてのタグをできますが、クラスをタグとしてするのがです。

```
public static final String tag = MyAnimator.class.getSimpleName();
```

ログレベル

Androidロガーには6つのレベルがあり、それぞれはのをたします。

- `ERROR` `Log.e()`
  - なをすためにされます。これは、 `Exception`スローするときにされるレベルです。
- `WARN` `Log.w()`
  - になについてをするためにされます。
- `INFO` `Log.i()`
  - アプリケーションのにするよりいレベルのをすためにされます。
- `DEBUG` `Log.d()`
  - アプリケーションのデバッグにっておくとなをログにするのにされますが、アプリケーションの
- `VERBOSE` `Log.v()`
  - アプリケーションのにするさなをするのログにされます。
- `ASSERT` `Log.wtf()`
  - してはならないにするをするためにされます。
  - *wtf*は「がひどいか」をします。

ロギングの

ロギングのは、アプリケーションからののイベントをて、エラーやなどのをにつけることです。たとえば、テキストファイルからをみむアプリケーションをしてください。ただし、ファイルがではないとってしています。ログをしないアプリケーションのログトレースは、のようになります。

```
E/MyApplication: Process: com.example.myapplication, PID: 25788
                         com.example.SomeRandomException: Expected string, got 'null' instead
```

にはデバッガをしてステップスルーすると、にはにつながる、なにつながるのスタックトレースがきます

ただし、ロギングをにしたアプリケーションのログトレースは、のようになります。

```
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
```

ログをすばやくれば、ファイルがであることはらかです。

ロギングにすべき

ロギングは、Androidがアプリケーションのをよりくできるなツールですが、ロギングにはいくつかのがあります。

ログの

Androidアプリケーションでは、のログをしてするのがです。したがって、ログはみやすく、するなのみがまれていることがにです。

パフォーマンス

ロギングにはのシステムリソースがです。に、これはをすものではありませんが、にすると、ログがアプリケーションのパフォーマンスにをえるがあります。

セキュリティ

、Google Playマーケットプレイスにいくつかのアプリケーションがされ、ユーザーはのすべてのアプリケーションのログをできます。このしないデータにより、ユーザーはをすることができます。まかには、アプリケーションをにするに、されていないデータにまれるログをにしてください。

ロギングは、にえるのために、Androidアプリケーションのなです。なログトレースをするは、ソフトウェアのもなの1つですが、AndroidのLogクラスは、よりにするのにちます。

については、 LoggingとLogcatのをしてください。

Logcatからソースへのリンクきログ

これは、コーードへのリンクをするうまいトリックなので、ログをしたコーードににジャンプできます。

のコーードをして、このびしをいます。

```
MyLogger.logWithLink("MyTag","param="+param);
```

```
07-26...012/com.myapp D/MyTag: MyFrag:onStart(param=3)  (MyFrag.java:2366) // << logcat
converts this to a link to source!
```

これはコーードですMyLoggerというクラス。

```
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());
}
```

これはなですが、にしてびしへのリンクをすることができますヒントスタックは[3]ではなく[4]になります。

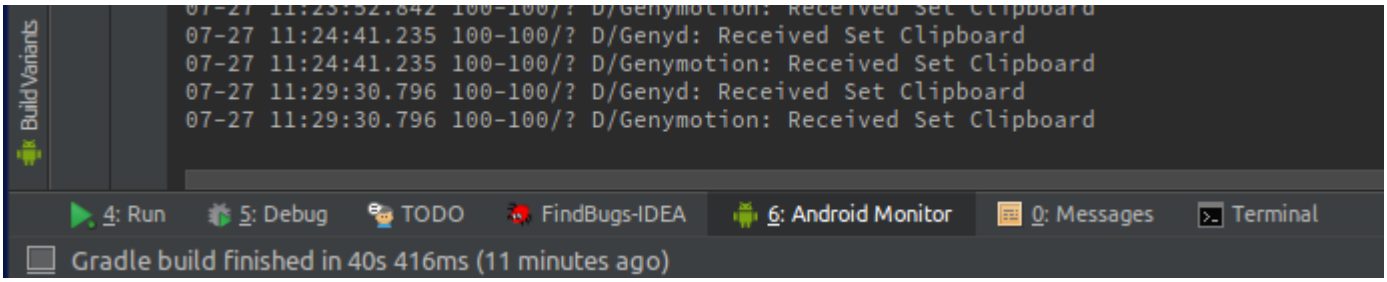
ログキャットの

Logcatは、デバイスがエラーをスローしたときのスタックトレースや、アプリケーションからLogクラスできんだメッセージなど、システムメッセージのログをダンプするコマンドラインツールです。

Logcatのは、Android StudioのAndroidモニターまたはadbコマンドラインでできます。

Android Studioの

「Android Monitor」アイコンをクリックして



またはMacでWindows / Linuxまたは`CMD` + `6`に`Alt`キー + `6`をして。

コマンドラインで

ない

```
$ adb logcat
```

タイムスタンプき

```
$ adb logcat -v time
```

のテキストのフィルタリング

```
$ adb logcat -v time | grep 'searchtext'
```

コマンドライン*logcat*には 、くのオプションとフィルタがあり、 ここでしています 。
シンプルでなは、すべてのタグに「error」をつすべてのログメッセージをするのフィルタです。

```
$ adb logcat *:E
```

ロギン グコードの

`Android Studio`の`Live templates`は、なロギングのためのいくつかのショートカットがされています。
ライブテンプレートをするには、テンプレートのをし、 `TAB`キーを`TAB`かしてステートメントをするだけです。

- `logi` →→ `android.util.Log.i(TAG, "$METHOD_NAME$: $content$");`
  - `$METHOD_NAME$`はあなたのメソッドににきえられ、カーソルはがたされるのをちます。
- `loge` →same、エラー
- りのロギングレベルでは、

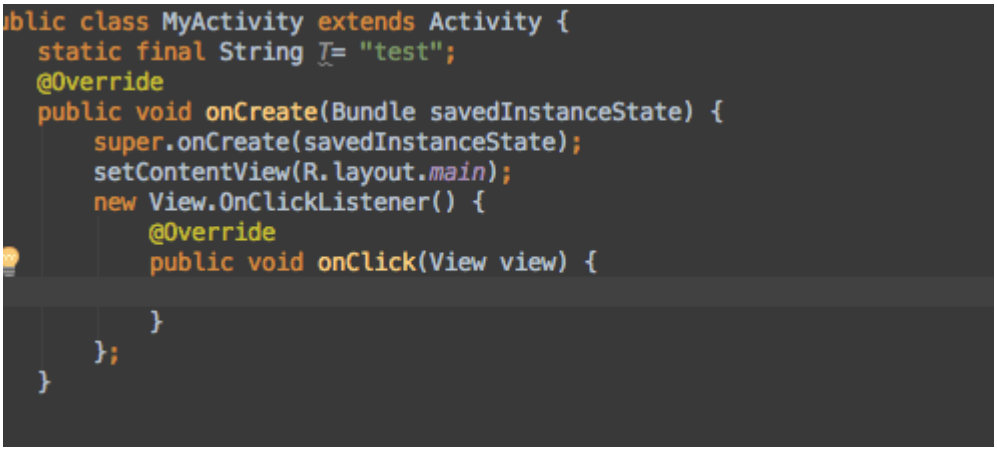
テンプレートのなリストは、 `Android Studio`の `ALT` + `s`とタイプ "live"にあります。カスタムテンプレートをすることもです。

`Android Studio`の`Live templates`がでないは、 Android Postfix Plugin

これは、ロギングをでくのをけるのにつになライブラリです。

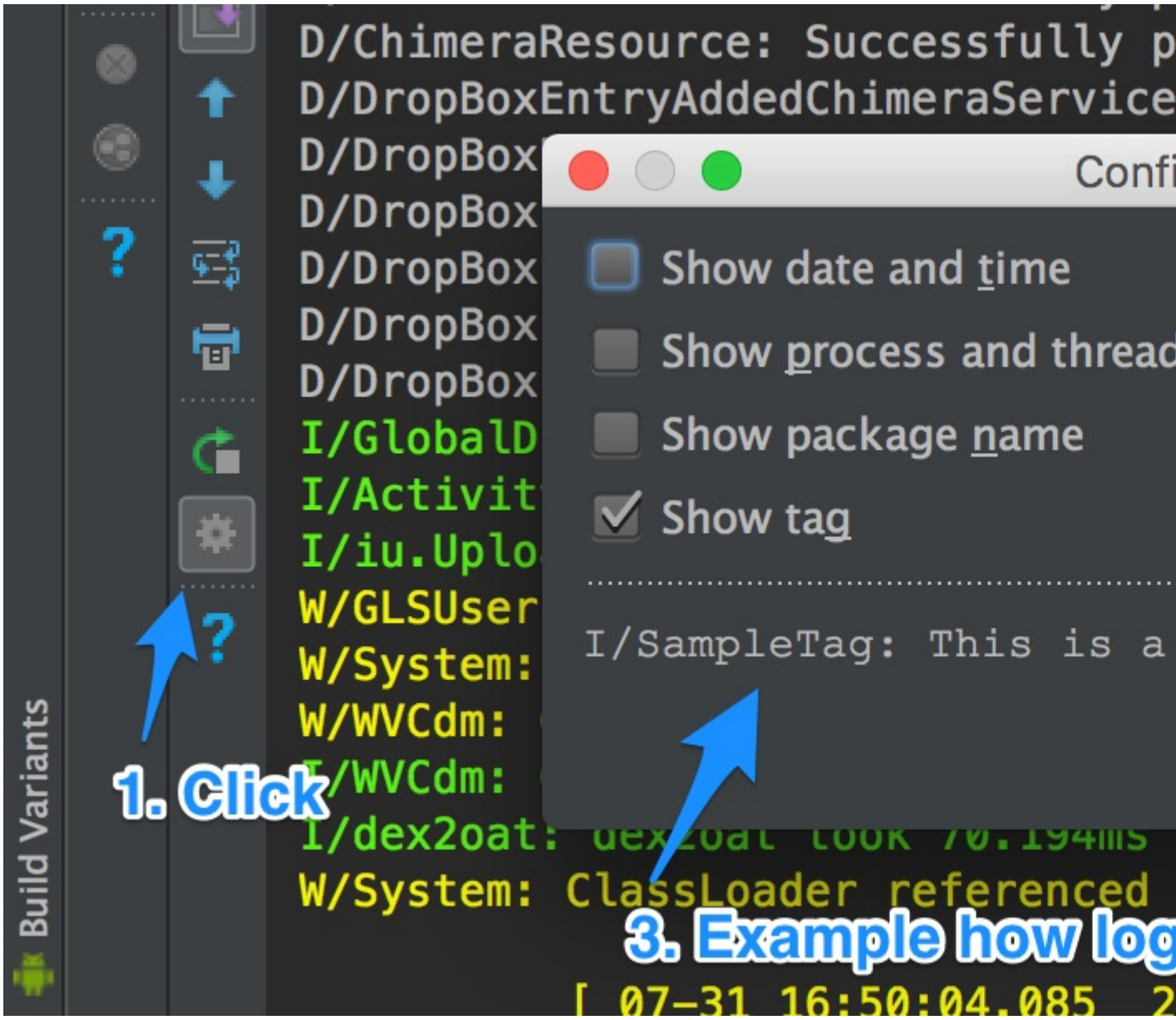
はにです

.log - ロギング。 "TAG"がある、 "TAG"をします。それはクラスをします。

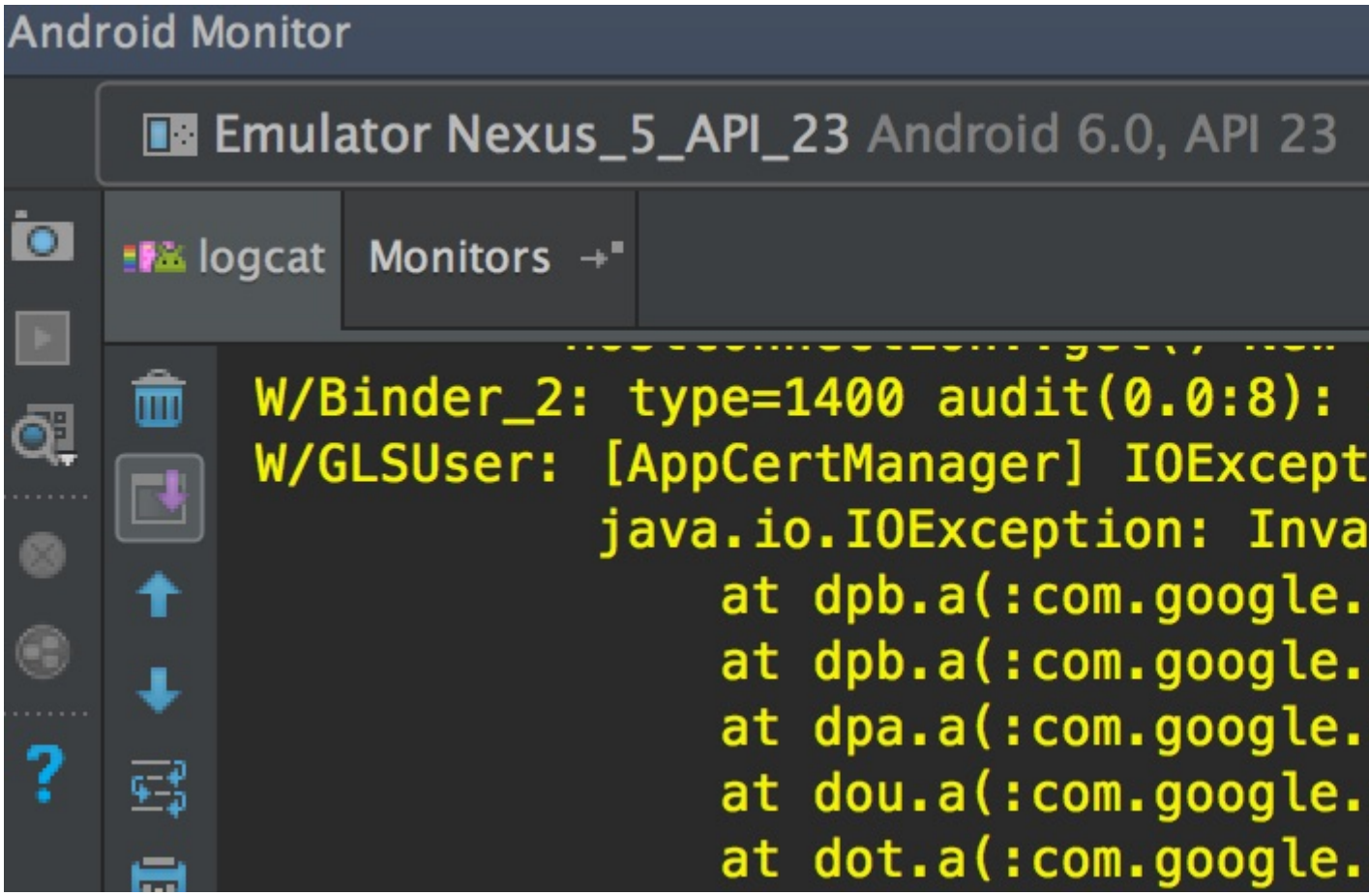


Android Studioの

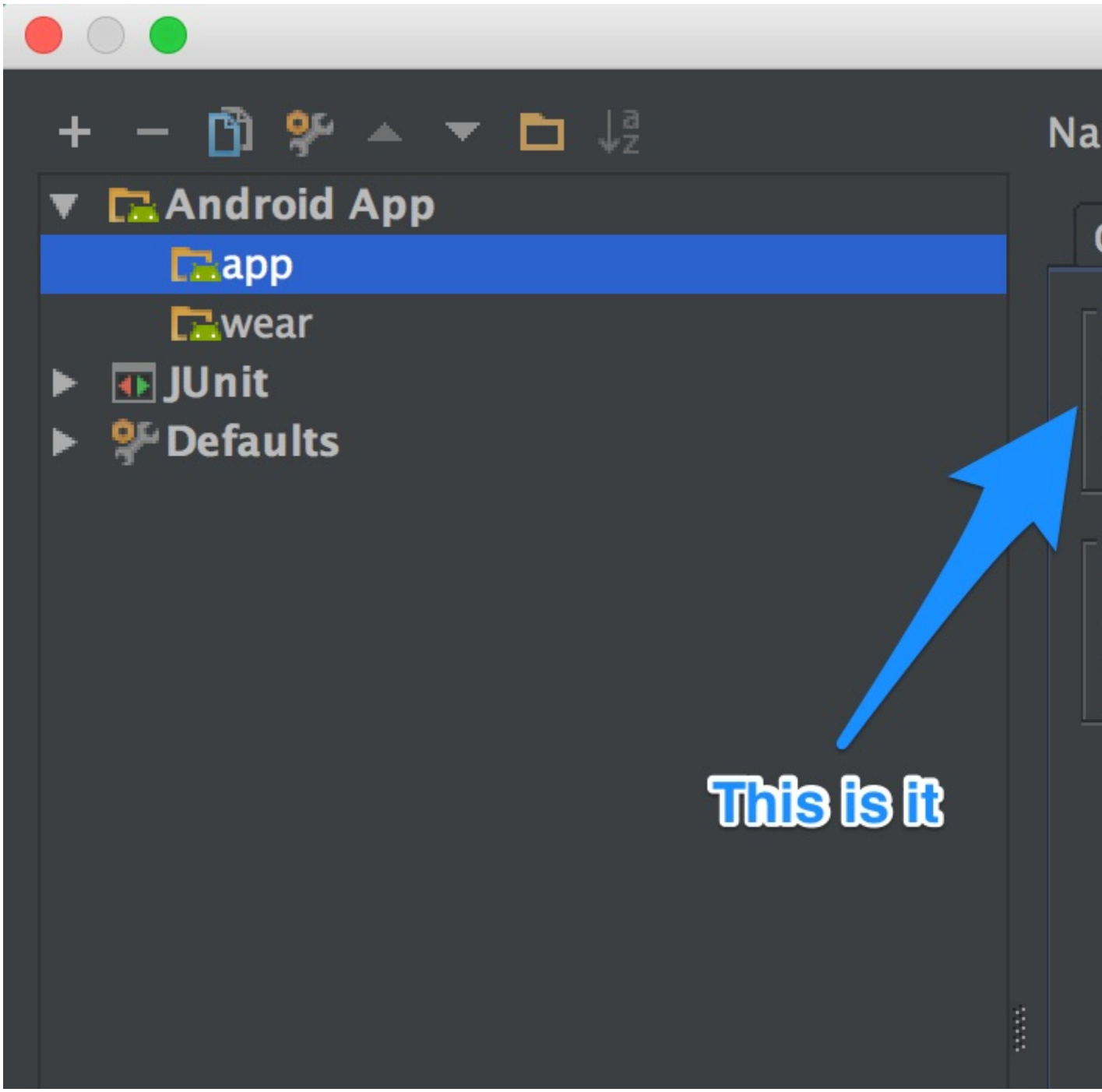
1. されたをす/する


D/ChimeraResource: Successfully p
D/DropBoxEntryAddedChimeraService
D/DropBox
D/DropBox
D/DropBox
D/DropBox
D/DropBox
I/GlobalD
I/Activit
I/iu.Uplo
W/GLSUser
W/System:
W/WVCdm:
I/WVCdm:
I/dex2oat:
W/System: ClassLoader referenced
Confi
Show date and time
Show process and thread
Show package name
Show tag
I/SampleTag: This is a
1. Click
3. Example how log
Build Variants

2. ロギングのの



3. /デバッグアプリケーションのにログウィンドウをく/にする



ログをクリアする

ログをフラッシュするには

```
adb logcat -c
```

オンラインでログとログキャストのをむ https://riptutorial.com/ja/android/topic/1552/ログとログキャストの

237: ロケーション

き

Androidのロケーションapiは、ユーザーのをしたり、ユーザーがいつエリアをれたかをしたりジオフェンシング、ユーザーの、、などをするなど、々なでさまざまなアプリケーションでされます。

しかし、Android Location APIはユーザーのをするのではありません。は、Androidの`LocationManager`とのなライブラリのいのです。

Androidでロケーションアプリをするには、の2つのがあります。

- Androidのネイティブオープンソース`LocationManager`
- Google Playサービスのである Googleの`FusedLocationProviderApi`

LocationManager

- よりな
- すべてのデバイスで
- Androidフレームワークの

- にされないと、バッテリのがになります
- デバイスがをつけることができないえば、のなGPSには、ロケーションプロバイダをりえるロジックがです。

- NMEAリスナー
- GPSステータスリスナー
- プロバイダのステータスのをくGPSがユーザーによってオフになっている
- ロケーションソースをするプロバイダのリスト

プロバイダー

GPS

- な
  - `ACCESS_FINE_LOCATION`
- 10m100m
- HIGH
- をはっきりとえる
-
  - は、、1に1のでわれますが、GPSをしていないでA-GPSができないは、をするまでにかかります。
  - がはっきりとえなくなった、GPSポイントはによくまらずロケーションポイントが「

ジャンプ」、「 アーバンキャニオン 」のためにのエリアでをくがあります。

ネットワーク

- な
  - ACCESS_COARSE_LOCATIONまたはACCESS_FINE_LOCATION
- 100m1000m +
- LOW - MEDIUM
- セルタワーまたはLANの
- ノート
  - のはGPSよりもにします
  - のは、にうまくいっていませんロケーションポイントは「ジャンプ」します。はさまざまなWi-Fi、、セルタワーのタイプなど

- な
  - ACCESS_FINE_LOCATION
- 10m1000m +
- なし
- のアプリがGPSまたはネットワークのいずれかからをしたのみ
- ノート
  - あなたになをえるためにこれにらないでください。これは、ロケーションをうのアプリににをけ、それらのロケーションをします。
  - FusedLocationProviderApiでされたポイントをしません。されたポイントをするためにされたなポイントのみをします。

FusedLocationProviderApi

- バッテリーのを「すぐに」
- いGPSをうまくう
- よりにをする

- GPSによるよりかい
- すべてのまたはのでできないがあります
- サードパーティのライブラリが

- なバッター・セービングのためのロケーション・プロバイダのな
- には、ネットワークロケーションプロバイダ
- よりなライブラリのにより、よりくのが
- するプロバイダのタイプをするはありません

LocationRequestのレベル

PRIORITY_HIGH_ACCURACY

-

な
    - ACCESS_FINE_LOCATIONよりな、またはACCESS_COARSE_LOCATIONよりでない
- 10m100m
- HIGH
- Google Playサ―ビスができるところはどこにでもあります。
- ノ―ト
    - ACCESS_FINE_LOCATIONがされていないは、ロケ―ションをするためにGPSはされませんが、しいではかなりなポイントがつかるでしょう。
    - ACCESS_FINE_LOCATIONがされている、がえられたときにデバイスをどのようににできるかにじて、GPSをしてをすることができます。
    - これは、のよりもなをするかもしれませんが、それはまだ " アーバンキャニオン "になりがちです。

PRIORITY_BALANCED_POWER_ACCURACY

- な
    - ACCESS_FINE_LOCATIONよりな、またはACCESS_COARSE_LOCATIONよりでない
- 100m1000m +
- MEDIUM
- Google Playサ―ビスができるところはどこにでもあります。
- ノ―ト
    - PRIORITY_HIGH_ACCURACYじノ―ト
    - はいですが、このではGPSをしてをすることがあります。

PRIORITY_LOW_POWER

- な
    - ACCESS_FINE_LOCATIONまたはACCESS_COARSE_LOCATION
- 100m1000m +
-
- Google Playサ―ビスができるところはどこにでもあります。
- ノ―ト
    - おそらく、GPSはしていませんが、これまでのところテストされていません。
    - アップデ―トはにあまりではありません
    - にのなをするためにされます

PRIORITY_NO_POWER

- な
    - ACCESS_FINE_LOCATIONまたはACCESS_COARSE_LOCATION
- 10m1000m +
- なし
- Google Playサ―ビスができるところはどこにでもあります。
- ノ―ト
    - LocationManager PASSIVE_PROVIDERとほぼじ

- にGoogle Playサービスのをします`PASSIVE_PROVIDER`は、されたなのをします

トラブルシューティング

OnLocationChangedはびされません

これはAndroidのをするのなであるとわれるため、よくあるのなチェックリストをします。

---

1. マニフェストをしてください

   もなの1つは、ながえられていないことです。 GPSをしているは、 `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>`をし、そうでないは`<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>`。 GoogleのFusedLocationApiにはACCESS_FINE_LOCATIONが`ACCESS_FINE_LOCATION`。

---

2. Android 6+ ランタイムのアクセスをしてください

   し、をリクエストしてくださいがえられていないは、クラッシュするか、すべてのをキャッチしているはします。もされません。ユーザーがアプリのにあなたにをえてもはありません。にすべてのにアクセスがあるかどうかをしてください。ユーザーはににき、りすことができます。

---

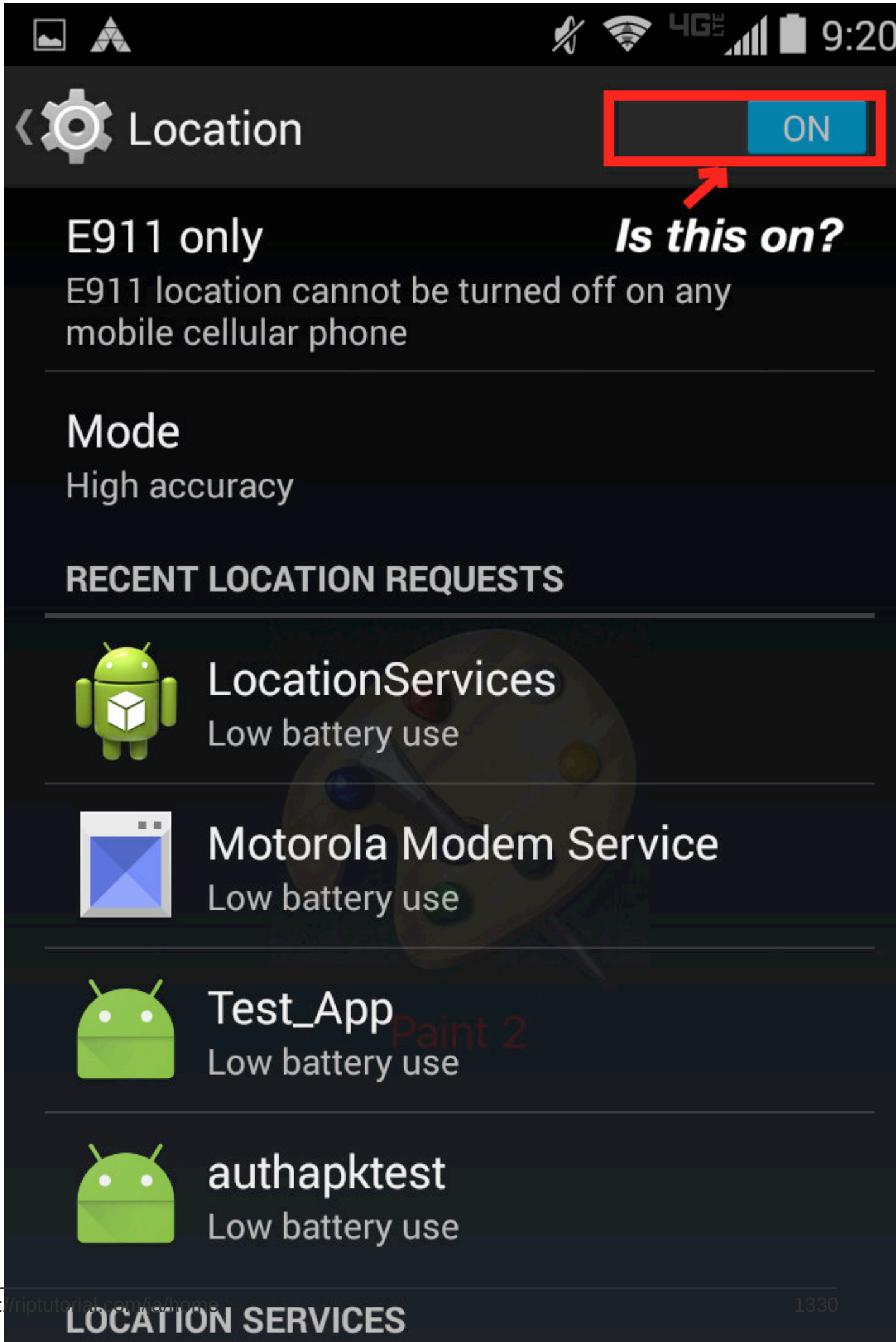
3. コードをしてください

   あなたはしいリスナーをしているとしていますか `BroadcastReceiver`または`IntentService`をマニフェストにしましたかあなたはしている`PendingIntent.getService()`に`BroadcastReceiver`クラス、または`getBroadcast()`に`IntentService`クラスあなたはリクエストしたにあなたのリスナーをあなたのコードのどこかにしていないとしていますか

---

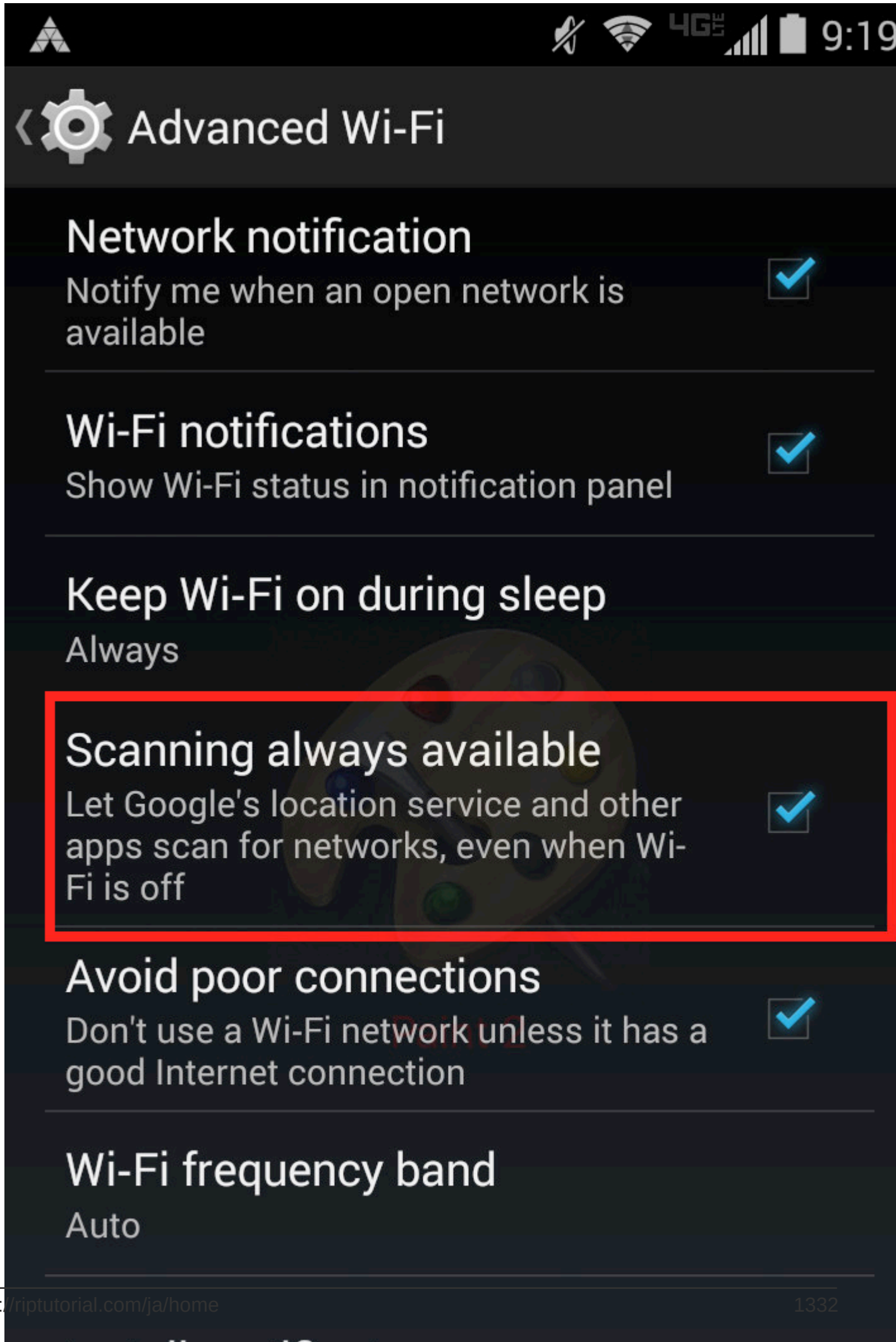
4. デバイスのをしてください

   らかに、サービスがオンになっていることをしてください。


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

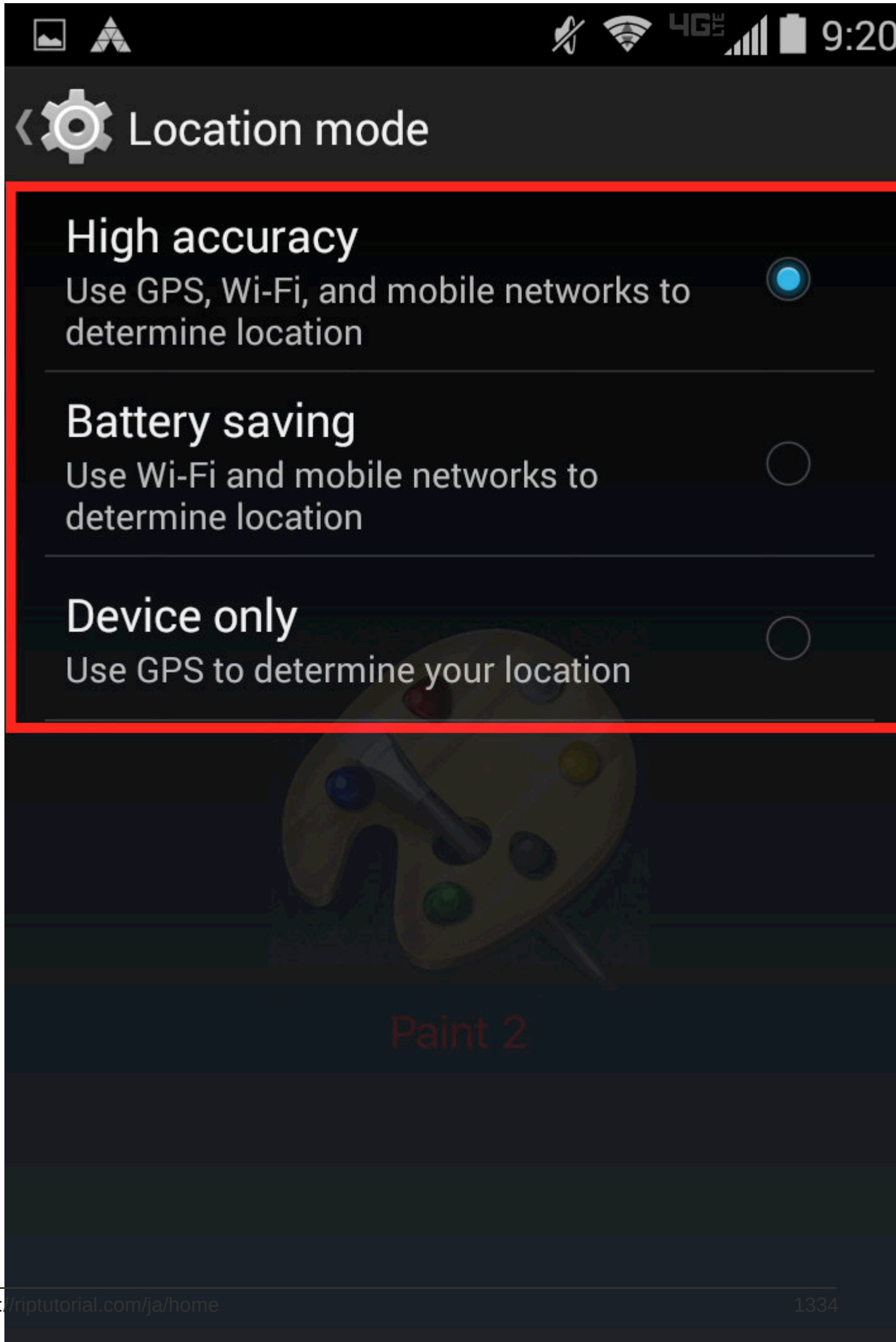
ネットワ―クサ―ビスをしているは、「にスキャン」をにしましたかあなたのモ―ドを「」「」または「バッテリ―」「ネットワ―クのみ」にしていますか

「」または「バッテリ―」「ネットワ―クのみ」にしていますか


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

GPSをしている、ロケーションモードで「ベスト」「」または「デバイスのみ」をオンにしましたか

にしましたか


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

はい、これはここに2あります。あなたがしてみましたLocationListenerのわりに、PendingIntent 、またはそのに、あなたがにするためLocationManagerしくアクティビティまたはサ―ビスのライフサイクルので、のリクエストがされていないことをしていますか

PendingIntent 、またはそのに、あなたがにするためLocationManagerしくアクティビティまたはサ―ビスのライフサイクルので、のリクエストがされていないことをしていますか

---

6. あなたのをチェックしてください

   サンフランシスコのんのの1にあるGPSをテストしていますかどこにいてもネットワ―クのをテストしていますかあなたは、あなたのデバイスがなぜをしていないのだろうかとにう、すべてののののバンカ―でいていますかののトラブルシュ―ティングをうときには、にをしてください。

がうまくいかないはそれほどではないかもしれませんが、これらのなをするに、このなチェックリストをしてください。

Examples

API

アクティビティw / LocationRequestをした

```
/*
 * 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
    }
}
```

サービスw / PendingIntentとBroadcastReceiverをした

アクティビティの

おめ 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
            }
        }
    }
}
```

サービス

マニフェストにこのサービスをすることをれないでください

```
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

このレシ―バ―をマニフェストにすることをれないでください

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

LocationManagerをしたロケ―ションのリクエスト

いつものように、ながあることをするがあります。

```
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) {

    }
}
```

LocationManagerをしてのスレッドでロケーションアップデートをリクエストする

いつものように、ながあることをするがあります。

```
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) {

    }
}
```

ジオフェンスをする

は`GeoFenceObserversationService`シングルトンクラスをしました。

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);
        }
    }
```

はどこでサービスをめたのですかアプリケーションクラスから

- `startService(new Intent(getApplicationContext(),GeoFenceObserversationService.class));`

ジオフェンスの

- `GeoFenceObserversationService.getInstant().addGeofences();`

Geocoderをしてロケーションからアドレスをする

`FusedAPI`から`Location`オブジェクトをしたら、そのオブジェクトから`Address`をにできます。

```
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;
}
```

BroadcastReceiverでのの

に、ロケーションのをするBroadcastReceiverクラスをします。

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

onConnectedコールバックでGoogleApiClientにすると、のようになります。

```
@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);
}
```

なライフサイクル·コールバックでロケーションインテントをすることをれないでください。

```
@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();
}
```

オンラインでロケーションをむ https://riptutorial.com/ja/android/topic/1837/ロケーション

238: ロボエレクトリック

き

テストは、のやシステムのえば、データベースをすることなく、コードをりしてしてテストします。

Robolectricは、Android SDKのjarファイルをして、Androidアプリのをテストできるユニットテストフレームワークです。テストはワークステーションのJVMのでされます。

それらをみわせることで、まだAndroid APIをしているJVNでテストをできます。

Examples

ロボフィールド

```
@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!");
  }
}
```

robolectricをするには、 `@Config`アノテーションをして、クラスまたはメソッドをテストします。

カスタムアプリケーションクラスでする

```
@RunWith(RobolectricTestRunner.class)
@Config(application = MyApplication.class)
public final class MyTest {
}
```

ターゲットSDKをする

```
@RunWith(RobolectricTestRunner.class)
@Config(sdk = Build.VERSION_CODES.LOLLIPOP)
public final class MyTest {
}
```

カスタムマニフェストでする

すると、robolectricはのディレクトリをにしてされます。デフォルトは`AndroidManifest.xml`

リソースとは、マニフェストにしてロードされます。

```
@RunWith(RobolectricTestRunner.class)
@Config(manifest = "path/AndroidManifest.xml")
public final class MyTest {
}
```

をする

のあるはアンドロイドのにあります。

```
@RunWith(RobolectricTestRunner.class)
public final class MyTest {

    @Config(qualifiers = "sw600dp")
    public void testForTablet() {
    }
}
```

オンラインでロボエレクトリックをむ https://riptutorial.com/ja/android/topic/8743/ロボエレクトリック

239: のあるアプリケーションの

Examples

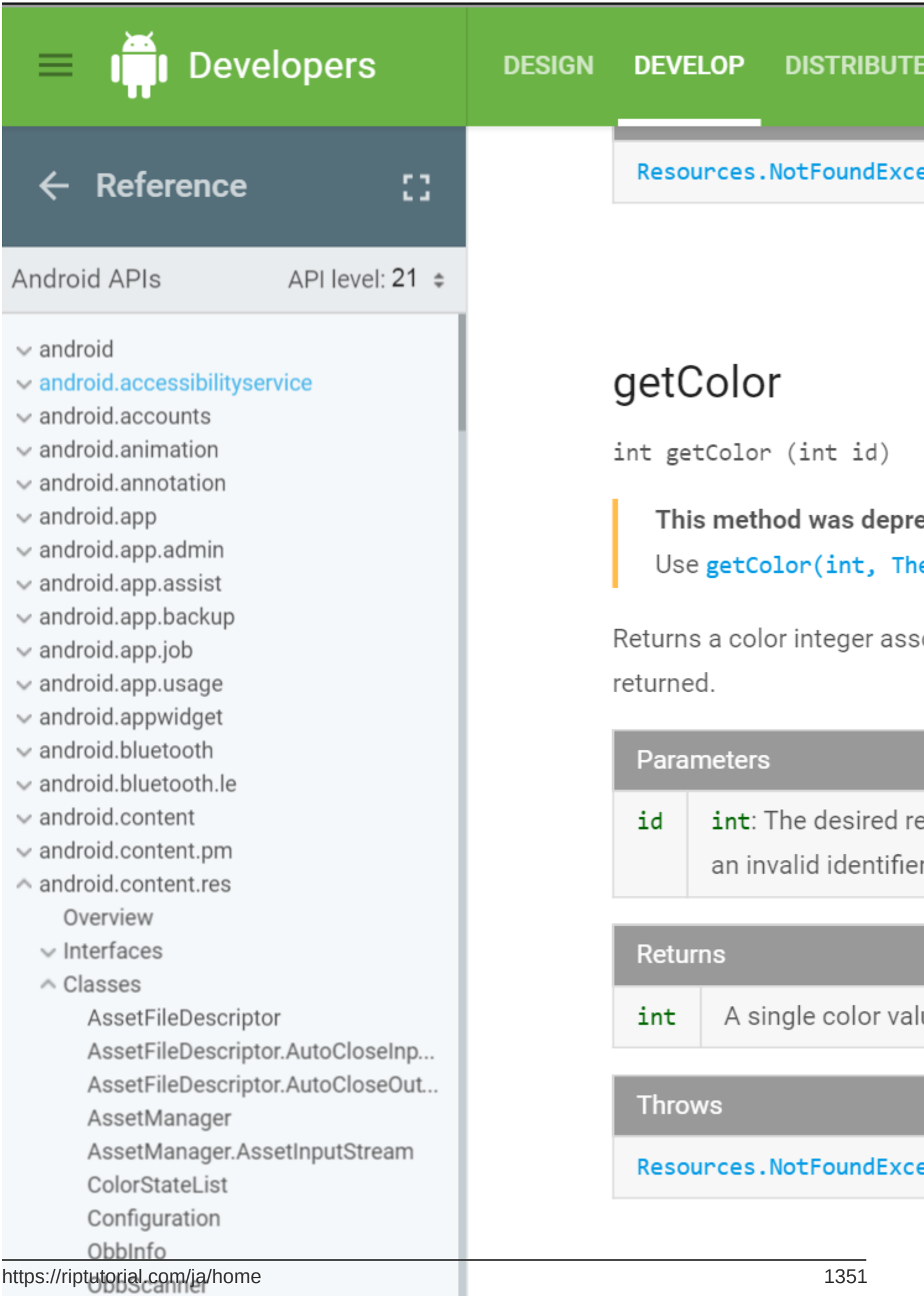
のAPIをする

がプロセスにのAPIにするはいです。されたプログラムは、プログラマがであるか、よりいがするために、にプログラマがすることをおめします。コンパイラーとアナライザー LINTのようなは、のプログラムがされるか、されないコードできされるとします。

されたAPIは、、Android Studioで、ちりをってされます。のでは、メソッド`.getColor(int id)`はされていません。

```
getResources().~~getColor~~(R.color.colorAccent));
```

であれば、はAPIとをすることをおめします。ライブラリのをするには、ライブラリのAndroidドキュメントにアクセスし、「APIレベルxの」のセクションをチェックします。


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

をするがあります。えば

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

このをすると、アプリはしいAndroidバージョンとのバージョンとのをします。

なサポートライブラリをする

サポートライブラリがされる、しばしば、よりないクライアントコードでじタスクをするためのヘルパーメソッドがあります。のif / elseブロックのわりに、をしてください

```
final int color = android.support.v4.content.ContextCompat
    .getColor(context, R.color.colorPrimary);
```

シグネチャがなるしいメソッドといバージョンではできなかったがあるくのをつメソッドは、このようなヘルパーメソッドをっています。のものをつけるには、 `ContextCompat` 、 `ViewCompat`などのクラスのサポートライブラリをして`ViewCompat`。

オンラインでのあるアプリケーションのをむ https://riptutorial.com/ja/android/topic/4291/のあるアプリケーションの

240:

Examples

NetworkOnMainThreadException

ドキュメントから

> アプリケーションがメインスレッドでネットワークをしようとしたときにスローされるです。
>
> これは、Honeycomb SDKをターゲットとするアプリケーションでのみスローされます。のバージョンのSDKをとしたアプリケーションでは、メインイベントループスレッドでネットワーキングをうことがされていますが、あまりされません。

そのをきこすがあるコードのをにします。

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

のコードは、アプリケーションがメインスレッドでネットワークをしようとしているため、Honeycomb SDKAndroid v3.0をターゲットとするアプリケーションにして

NetworkOnMainThreadExceptionをスローします。

このをするには、ネットワークをにAsyncTask 、 Thread 、 IntentServiceなどのバックグラウンドタスクでするがあります。

```
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

これはにExceptionなExceptionです。これにより、アプリケーションのまたはにアプリケーションがします。 LogCatメッセージがされます

```
android.content.ActivityNotFoundException : Unable to find explicit activity class;
have you declared this activity in your AndroidManifest.xml?
```

この、 AndroidManifest.xmlファイルでアクティビティをしているかどうかをしてください。

AndroidManifest.xmlでActivityをするもなはのとおりです。

```
<activity  android:name="com.yourdomain.YourStoppedActivity" />
```

OutOfMemoryError

これは、ヒープにのメモリをするとするランタイムエラーです。これは、ビットマップをImageViewにロードするときによくこります。

いくつかのがあります

1. なアプリケーションヒープをする

AndroidManifest.xmlのapplicationタグに "largeHeap"オプションをします。これにより、アプリでできるメモリがえますが、なはされないがあります。

```
<application largeHeap="true" ... >
```

2. ビットマップをリサイクルする

ビットマップをみんだ、リサイクルしてメモリをしてください。

```
    if (bitmap != null && !bitmap.isRecycled())
       bitmap.recycle();
```

3. サンプリングされたビットマップをメモリにロードする

BitmapOptionsとinSampleSizeをして、ビットマップをにメモリにロードしないでください。

たとえば、 Androidのドキュメントをしてください

DexException

```
com.android.dex.DexException: Multiple dex files define Lcom/example/lib/Class;
```

このエラーは、パッケージングに、じメソッドセットをする2つの.dexファイルをするためにします。

、これは、アプリケーションがってじライブラリに2つののをしたためにします。

たとえば、プロジェクトがあり、それぞれにのをつ2つのライブラリ$_A$と$_B$をしたいとします。ライブラリー$_B$すでにライブラリー$_A$にしている、ライブラリー$_A$がでプロジェクトにされると、このエラーがスローされます。ライブラリ$_B$コンパイルするとすでに$_A$からコードのセットがえられていたので、コンパイラがライブラリ$_A$をバンドル$_A$ 、パッケージ$_A$のメソッドがにパッケージされています。

するには、あなたののどれもがってこのように2されないようにしてください

されなかった

キャッチされていないをしたいは、それらをonCreateメソッドですべてキャッチしようとするアプリケーションクラス

```
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();
    }
}
```

しないにするハンドラ―の

これは、システムの「アプリケ―ションXYZがクラッシュしました」とに、キャッチされていないにどのようにできるかです。

```
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
    }
}
```

に、このApplicationクラスをAndroidManifest.xmlにします。

```
<application android:name="de.ioxp.arkmobile.MyApplication" >
```

オンラインでをむ https://riptutorial.com/ja/android/topic/112/

241: の

き

ここでは、をしてActivitiesまたはFragmentsののをつけます。こののとして、Google Playストアアプリがあります。このアプリは、リストのAppアイコンをAppのビュー—にします。

- transaction.addSharedElementsharedElementView、 "targetTransitionName";
- fragment.setSharedElementEnterTransitionしいCustomTransaction;

Examples

2つのフラグメントのの

このでは、2つのなるImageViews 1つをChooserFragmentからChooserFragmentにするがありDetailFragment。

ChooserFragmentレイアウトには、のtransitionNameがです。

```
<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" />
```

ChooserFragmentsクラスでは、クリックされたViewとIDをフラグメントのをするActivityにすがあります DetailFragmentにするイメージリソースをるためにIDがDetailFragment。 アクティビティにをにすについては、のドキュメントでしくしています。

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

`DetailFragment`では、の`ImageView`にもの`transitionName`がです。

```
<ImageView
    android:id="@+id/image_shared"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:transitionName="sharedImage" />
```

`DetailFragment`の`onCreateView()`メソッドでは、どのイメージ・リソースをするかをするがありますそうしないと、はにえます。

```
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;
}
```

`Activity`はコールバックをけり、フラグメントのをします。

```
@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();
}
```

`Transition`をれないでください。このでは、をおよび/します。

```
@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class DetailsTransition extends TransitionSet {

    public DetailsTransition() {
        setOrdering(ORDERING_TOGETHER);
        addTransition(new ChangeBounds()).
            addTransition(new ChangeTransform()).
            addTransition(new ChangeImageTransform());
    }

}
```

オンラインでのをむ https://riptutorial.com/ja/android/topic/8933/の

242: およびストレージへのファイルの

- FileOutputStream openFileInput
- FileOutputStream openFileOutput、intモード
- ファイルファイルディレクトリ、
- ファイルパス
- ファイルgetExternalStoragePublicDirectoryString
- ファイルgetExternalFilesDirString

パラメータ

パラメータ	
	くファイルの。パスりはできません。
モード	モード。`MODE_PRIVATE`デフォルトののために、そして`MODE_APPEND`のファイルをします。そののモードには、`MODE_WORLD_READABLE`と`MODE_WORLD_WRITEABLE`がありますが、どちらもAPI 17ではです。
	しいファイルをするためのファイルのディレクトリ
パス	しいファイルのをするパス
タイプ	するファイルの。`null`、または`DIRECTORY_MUSIC`、`DIRECTORY_PODCASTS`、`DIRECTORY_RINGTONES`、`DIRECTORY_ALARMS`、`DIRECTORY_NOTIFICATIONS`、`DIRECTORY_PICTURES`、または`DIRECTORY_MOVIES`いずれかになります。

Examples

ストレージの

デフォルトでは、ストレージにするファイルはアプリケーションです。のでは、のアプリケーションやユーザーはアクセスできません。 これらのファイルは、ユーザーがアプリケーションをアンインストールするとされます 。

ファイルにテキストをきむには

```
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();
}
```

のファイルにテキストをするには

`openFileOutput` modeパラメータに`Context.MODE_APPEND`をする

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

ストレージの

「」ストレージは、ファイルをユーザーのデバイスにするためにできるのタイプのストレージです。「」ストレージとのないはのとおりです。

- になわけではありません。リムーバブルメディアSDカードの、ユーザーはにストレージをりすことができます。
- ではありません。ユーザーおよびのアプリケーションはこれらのファイルにアクセスできます。
- ユーザーがアプリケーションをアンインストールすると、 `getExternalFilesDir()` したディレクトリにしたファイルはされます。

ストレージをするには、まずなをするがあります。のものをするがあります

- みきの`android.permission.WRITE_EXTERNAL_STORAGE`
- のために`android.permission.READ_EXTERNAL_STORAGE`

これらのをするには、 `AndroidManifest.xml`そのをするがあります

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

APIレベル**23**をしているはなアクセス であるため、にアクセスをするがあります 。

ストレージへのきみまたはみりをみるに、ずストレージメディアがであることをするがあります。

```
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
}
```

ストレージにファイルをきむときは、ファイルをパブリックまたはプライベートとしてするかどうかをするがあります。これらののファイルは、デバイスのユーザーやのアプリケーションからききアクセスできますが、それらのにはきないがあります。

ユーザーがアプリをアンインストールすると、されたファイルはにります。パブリックとしてするがあるファイルのは、アプリケーションをじてされるです。

ユーザーがアプリをアンインストールすると、プライベートファイルはすべてされます。これらのタイプのファイルはアプリのものであり、ユーザーやのアプリケーションにはできません。 Ex 。アプリケーションによってダウンロード/されるファイル。

パブリックファイルとプライベートファイルのについて`Documents`ディレクトリにアクセスするはのとおりです。

パブリック

```
// 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();
```

プライベート

```
// 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およびストレージ - の

Androidには、およびのストレージについてしています。じことについてはStackoverflowにするくのがあります。これはに、Google /Androidのドキュメントにづくが、のAndroid OSユーザーのとはかなりなるというにしています。したがって、はこれをするとけになるといました。

たちがえること **-** ユーザーの**UT**

ストレージ**UT**	ストレージ**UT**
のメモリ	りしなセキュアデジタルSDカードまたはマイクロSD
Nexus 6Pの32 GB メモリ	samsung、sandisk、ストロンチウム、などのベンダーがするリムーバブルSDカードのストレージスペース

しかし、 **Android**のドキュメント/ガイド **- Google**の**GT**によると 、

ストレージ**GT**

デフォルトでは、ストレージにされているファイルはアプリケーションのプライベートなもので、のアプリケーションはアクセスできませんまたユーザーはアクセスできません。

ストレージ**GT**

これは、リムーバブルSDカードなどまたはリムーバブルにすることができます。

ストレージ**GT**は、2つのタイプにすることができます。

ストレージ	ストレージまたはリムーバブルストレージ**GT**
これは、のメモリまたはストレージUTとじです。	リムーバブルマイクロSDカードストレージまたはストレージUTとじです。
Nexus 6Pの32 GBメモリ	samsung、sandisk、ストロンチウム、などのベンダーがするリムーバブルSDカードのストレージスペース
このタイプのストレージは、USBケーブルをしてをPCにし、USBオプションのでカメラ*PTP*をすることで、Windows PCでアクセスできます。	このタイプのストレージは、USBケーブルをしてをPCにし、USBオプションのでファイルをすることによって、Windows PCでアクセスできます。

に、

ストレージ**GT=**ストレージ**UT**およびストレージ**UT**

リムーバブルストレージ**GT=**ストレージ**UT**

ストレージ**GT**には**UT**のはありません。

にはっきりとしましょう、

ストレージ**GT**デフォルトでは、ストレージにされたファイルはアプリケーションのプライベートなものであり、のアプリケーションはそのファイルにアクセスできません。あなたのアプリケーションユーザーはファイルマネージャをってそれらにアクセスすることもできません。ファイルマネージャで「しファイルを」オプションをにしたでも。ストレージGTのファイルにアクセスするには、Androidをルートするがあります。さらに、ユーザーがアプリケーションをアンインストールすると、これらのファイルは/されます。

したがって、ストレージGTはNexus 6Pの32/64 GBメモリとはえません

に、 ストレージ**GT**のはのようになります
`/data/data/your.application.package.appname/someDirectory/`

ストレージ**GT**

すべてのAndroidデバイスは、ファイルをするためにできるの「ストレージ」をサポートしています。ストレージにされたファイルはでみりで、USBストレージがコンピュータのファイルをできるようにすると、ユーザーがすることができます。

ストレージ**GT**のストレージUTまたはリムーバブルストレージGTのどこにでもくことができますmicro SDカード。これは、おいののOEMとAndroid OSのバージョンによってなります。

ストレージGTのファイルをみきするには、アプリが`READ_EXTERNAL_STORAGE`または`WRITE_EXTERNAL_STORAGE`システムをするがあります。

えば

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

ファイルのみりときみのをうがあるは、にみりアクセスもとするため、`WRITE_EXTERNAL_STORAGE`のみをするがあります。

ストレージ**GT**では、 アプリプライベートなファイルをすることもできます

しかし、

ユーザーがアプリケーションをアンインストールすると、このディレクトリとそのすべてのがされます。

ストレージ**GT**に アプリのファイルをいつするがありますか

のアプリケーションがするためのものではないファイルグラフィックテクスチャやアプリケーションのみでされるサウンドエフェクトなどをうは、ストレージのプライベートストレージディレクトリをするがあります

Android 4.4、アプリのプライベートディレクトリにあるファイルのみりやきみには、`READ_EXTERNAL_STORAGE`または`WRITE_EXTERNAL_STORAGE`はありません。したがって、`maxSdkVersion`をすることで、Androidのバージョンでのみアクセスをするがあることをできます。

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

ストレージ**GT**にする

これらのメソッドはどちらもContextクラスにします

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

File getFilesDir ()
```

プライマリストレージ、つまりストレージ**UT**にする

```
File getExternalStorageDirectory ()

File getExternalFilesDir (String type)

File getExternalStoragePublicDirectory (String type)
```

は、もが**Primary External Storage**の ルートをしますEnvironment.getExternalStorageDirectoryをしました。その、プライマリストレージはランダムなコンテンツでたされました。

その、これらの2つのメソッドがされました。

1. `Context`クラスでは、 getExternalFilesDirをし、プライマリストレージのアプリケーションのディレクトリをしていました。このディレクトリとそのは 、アプリケーションのアンインストールにされます。

2. Environment.getExternalStoragePublicDirectoryは、やムービーのようなよくられたファイルタイプをするをします。このディレクトリとそのは 、アプリケーションのアンインストールにされません 。

リムーバブルストレージ**GT**、すなわちマイクロ**SD**カードにする

APIレベル**19**は、SDカードにするなはありませんでした 。しかし、くのがのやAPIをってそれをうことができます。

には、APIレベル19Androidバージョン4.4 - Kitkatの`Context`クラスに1つのメソッドがされました 。

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

> これは、アプリケーションがするファイルをくことができるすべての/ストレージ·デバイスのアプリケーションディレクトリーへのパスをします。これらのファイルはアプリケーションのにあり、はメディアとしてユーザーにされません。

つまり、メモリとマイクロSDカードののタイプのストレージGTへのパスがされます。に、 **2**のは、マイクロSDカードのであるにではない。だから、このメソッドでコードをしてチェックアウトするがあります。

コードスニペットの

のアクティビティをつしいアンドロイドプロジェクトをし、にのコードをきました

`protected void onCreate(Bundle savedInstanceState)`の`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);
```

のコードをした、

```
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
```

はのをWindows PCにしました。けオプション、USBデバッグのをにしてから、このコードをしました。 あなたのをしていない。 **Android**のエミュレータでこれをすると、がなるがあります。のモデルはCoolpad Note 3 - Android 5.1でしています

のの

マイクロ**SD**ストレージの `/storage/sdcard1`

ストレージ**UT**の `/storage/sdcard0`

`/sdcard` `/storage/emulated/0`もストレージUTをしていることにしてください。しかし、これらは `/storage/sdcard0`へのシンボリックリンク`/storage/sdcard0` 。

Androidのさまざまなストレージパスをはっきりとするには、 このをごください

のすべてのストレージパスは、 ののパスです。ファイルは、じストレージパスにされないことがあります。ベンダー、メーカー、Android OSのバージョンによっては、/パスがのでなるがあります。

SDカードにデータベースをするSDのバックアップDB

```
    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;
    }
}
```

このメソッドをのようにびします。

ExportDB "myDB.db"、 "com.example.exam"、 "/ myFolder";

デバイスディレクトリを

にデバイスディレクトリをみみ/するためのストレージをします。

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

モデルクラスをする

```
//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;
    }
}
```

ディレクトリモデルをしてリストをし、ディレクトリデータをします。

```
//define list to show directory

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

のでディレクトリをします。

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

ディレクトリリストをログにします。

```
//print directory list in logs

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

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

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

のディレクトリのファイル/フォルダをフェッチするには、をするだけでじをし、されているパスをにし、じをします。

ファイルをするには

```
private String getExtension(String filename) {

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

    return extension;
}
```

オンラインでおよびストレージへのファイルのをむ https://riptutorial.com/ja/android/topic/150/およびストレージへのファイルの

243: モードポリシーコンパイルのバグをするツール。

き

Strict ModeはAndroid 2.3でデバッグにされたなクラスです。このツールは、ってされたことをし、するためにをします。これは、UIがされ、アニメーションがされるアプリケーションのメインスレッドで、なディスクまたはネットワークアクセスをするためにもにされます。 StrictModeは、にコンパイルモードでバグをするためのツールです。

StrictModeは、にコンパイルモードでバグをするためのツールです。これをすることで、アプリケーションでのメモリリークをできます。

Examples

のコードスニペットは、StrictMode for Thread Policiesをセットアップすることです。このコードは、アプリケーションのエントリポイントでされます。

```
StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
    .detectDiskWrites()
    .penaltyLog() //Logs a message to LogCat
    .build())
```

のコードは、SQLLiteのファイナライズにびされるかどうかをするようなメモリのリークをっています。

```
StrictMode.setVmPolicy(new StrictMode.VmPolicy.Builder()
    .detectActivityLeaks()
    .detectLeakedClosableObjects()
    .penaltyLog()
    .build());
```

オンラインでモードポリシーコンパイルのバグをするツール。をむ
https://riptutorial.com/ja/android/topic/8756/モードポリシー-コンパイルのバグをするツール-

244: アダプタによるデータ

Examples

スタブプロバイダをするダミーアダプタ

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");
    }

}
```

サービス

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

オーセンティケータ

```
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();
    }
}
```

オーセンティケータサービス

```
/**
 * A bound Service that instantiates the authenticator
 * when started.
 */
public class AuthenticatorService extends Service {
    // Instance field that stores the authenticator object
    private Authenticator mAuthenticator;
    @Override
    public void onCreate() {
        // Create a new authenticator object
        mAuthenticator = new Authenticator(this);
    }
    /*
     * When the system binds to this Service to make the RPC call
     * return the authenticator's IBinder.
     */
```

```
    @Override
    public IBinder onBind(Intent intent) {
        return mAuthenticator.getIBinder();
    }
}
```

AndroidManifest.xmlの

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.READ_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />

     <service
        android:name=".syncAdapter.SyncService"
        android:exported="true">
        <intent-filter>
            <action android:name="android.content.SyncAdapter" />
        </intent-filter>
        <meta-data
            android:name="android.content.SyncAdapter"
            android:resource="@xml/syncadapter" />
    </service>

    <service android:name=".authenticator.AuthenticatorService">
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator" />
        </intent-filter>
        <meta-data
            android:name="android.accounts.AccountAuthenticator"
            android:resource="@xml/authenticator" />
    </service>

    <provider
        android:name=".provider.StubProvider"
        android:authorities="com.yourpackage.provider"
        android:exported="false"
        android:syncable="true" />
```

res / xml / authenticator.xml

```
<?xml version="1.0" encoding="utf-8"?>
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.yourpackage"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:smallIcon="@mipmap/ic_launcher" />
```

res / xml / syncadapter.xml

```
<?xml version="1.0" encoding="utf-8"?>
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.yourpackage.android"
    android:allowParallelSyncs="false"
    android:contentAuthority="com.yourpackage.provider"
    android:isAlwaysSyncable="true"
    android:supportsUploading="false"
    android:userVisible="false" />
```

StubProvider

```
/*
 * Define an implementation of ContentProvider that stubs out
 * all methods
 */
public class StubProvider extends ContentProvider {
    /*
     * Always return true, indicating that the
     * provider loaded correctly.
     */
    @Override
    public boolean onCreate() {
        return true;
    }

    /*
     * Return no type for MIME type
     */
    @Override
    public String getType(Uri uri) {
        return null;
    }

    /*
     * query() always returns no results
     *
     */
    @Override
    public Cursor query(
            Uri uri,
            String[] projection,
            String selection,
            String[] selectionArgs,
            String sortOrder) {
        return null;
    }

    /*
     * insert() always returns null (no URI)
     */
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        return null;
    }

    /*
     * delete() always returns "no rows affected" (0)
     */
    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        return 0;
    }

    /*
     * update() always returns "no rows affected" (0)
     */
    public int update(
            Uri uri,
            ContentValues values,
            String selection,
            String[] selectionArgs) {
```

```
        return 0;
    }
}
```

ログインにこのをびして、ログインしているユーザーIDをつアカウントをします

```
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;
    }
```

をする

```
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);
```

オンラインでアダプタによるデータをむ https://riptutorial.com/ja/android/topic/1944/アダプタによるデータ

245: とローカリゼーションI18NとL10N

き

i18nとローカリゼーションL10nは、、のい、ターゲットオーディエンスのいにじてソフトウエアをさせるためにされます。

のローカリゼーションをするプロセス。つまり、のローカリゼーションのりみにしてできるににソフトウエアをすること。

ローカリゼーションソフトウエアをの//ロケールにさせるプロセス。

デバイスのローカライズをテストするには、のように`adb`をしてデバイスまたはエミュレータをのロケールでできます。

1. のコマンドをしてadbをします。 adb `adb shell`
2. `setprop persist.sys.locale [BCP-47 language tag];stop;sleep 5;start`ここで[BCP-47タグ]は、ここでするのコードです。BCP47コード

たとえば、アプリでのローカリゼーションをするには、のコマンドをします`setprop persist.sys.locale ja-JP;stop;sleep 5;start`

Examples

ローカリゼーションのマニフェストでのRTLサポートの

RTLからサポートは、i18nおよびL10nのにおいてなです。からにかれたとはなり、アラビア、、ヘブライなどのくのはからにかれています。よりグローバルなオーディエンスにアピールするには、プロジェクトのからこれらのをサポートするレイアウトをすることをおめします。ローカライゼーションのはでにえます。

RTLサポートは、 `AndroidManifest supportsRtl`タグをすることで、Androidアプリでにすることができます。

```
<application
    ...
    android:supportsRtl="true"
    ...>
...
</application>
```

ローカリゼーションのレイアウトでのRTLサポートの

SDK 17Android 4.2をすると、AndroidレイアウトでRTLサポートがされ、ローカリゼーションになです。は、レイアウトの`left/right`を`start/end`にきえるがあります。ただし、プロジェクトの

minSdkが17よりさいは、レイアウトでleft/rightおよびstart/endのをするがあります。

レイアウトの、 alignParentStartとalignParentEndをするがあります。

```
<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>
```

とレイアウトのをするには、のようにのをするがあります。

```
    <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"/>
```

パディングとマージンもにするがあります。

```
<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"/>
```

ローカリゼーションのRTLのレイアウトをテストする

されたレイアウトがRTLとがあるかどうかをテストするには、のをします。

[] - > [オプション] - > [] - > [RTLレイアウト]にします。

このオプションをにすると、デバイスはRTLロケールをするようになり、RTLサポートのためにアプリのすべてのをにできます。このまで、にしいロケール/サポートをするはありません。

ローカライズのためのコーディングデフォルトのとリソースの

ローカライゼーションをコーディングするためののステップは、デフォルトのリソースをすることです。このステップはのうちにくのがそれについてえることさえしません。ただし、サポートされていないロケールでデバイスをすると、すべてのリソースがデフォルトのフォルダからロードされるため、デフォルトのリソースをすることはです。リソースのいずれかがデフォルトのフォルダにない、アプリケーションはにクラッシュします。

デフォルトののセットは、されたののフォルダにくがあります。

```
res/values/strings.xml
```

このファイルには、アプリののユーザーがすことになっているのがまれているがあります。

また、アプリケーションのデフォルトリソースは、のフォルダとにするがあります。

```
res/drawable/
res/layout/
```

アプリに`anim`や`xml`などのフォルダがなは、デフォルトのリソースをのフォルダとにするがあります。

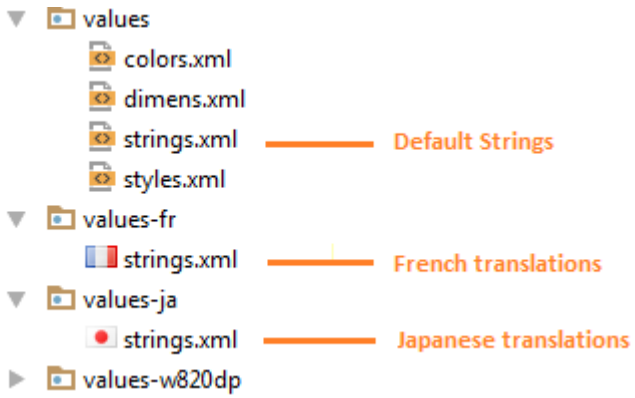
```
res/anim/
res/xml/
res/raw/
```

ローカライズのためのコーディングの

のロケールでをするには、のようなでのフォルダに`strings.xml`をするがあります。

```
res/values-<locale>/strings.xml
```

じもののをにします。



このでは、ファイル`res/values/strings.xml`にデフォルトののがあり、フランスのは`res/values-fr/strings.xml`というフォルダにあり、のは`res/values-ja/strings.xml`

のロケーールのもにアプリにできます。

ロケーールコードのなリストは、ここでつけることができます ISO 639コード

できない

プロジェクトには、されないのがまれているがあります。 SharedPreferencesのキーとしてされるまたはとしてされるは、このカテゴリにされます。これらのは、デフォルトの`strings.xml`のみするがあり、 `translatable="false"`でマークするがあります。えば

```
<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>
```

はバイリンガルなによってされることがいので、このはです。これにより、にするこれらのはされないをし、とをすることができます。

ローカライズのためのコーディングレイアウトの

のでしたように、しい`start/end`をしたは、のレイアウトをするはありません。ただし、デフォルトのレイアウトがのでしくしないがあります。 RTLの、からのレイアウトはされないことがあります。そのようなには、しいレイアウトをするがあります。

RTLレイアウトをにするために、 `ldrtl`リソース `ldrtl`はレイアウトをしますをして、に々のレイアウトファイルをできます。たとえば、デフォルトのレイアウトファイルを`res/layout/`し、RTLされたレイアウトを`res/layout-ldrtl/`することができます。

`ldrtl`はなリソースにしているため、みりにするにされたグラフィックスをできます。

`ldrtl`レイアウトのをするらしいがあります のレイアウト

オンラインでとローカリゼーションI18NとL10Nをむ https://riptutorial.com/ja/android/topic/8796/とローカリゼーション-i18nとl10n-

246:

Examples

のまり

のをえる

コードをするに、アンドロイドマニフェストにパーミッションをするがあります

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

ライブラリ

```
import android.os.Vibrator;
```

Contextから**Vibrator**のインスタンスをする

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

チェックデバイスにバイブレータがいている

```
void boolean isHaveVibrate(){
    if (vibrator.hasVibrator()) {
        return true;
    }
    return false;
}
```

に

*vibratelong []*パターン、*int repeat*をして、

```
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);
```

パターン

ロングのをすことによって、パターンをできます。それぞれのパターンは、ミリのをします。のはのです。アレイエントリは、、、、などのでする。

のは、このパターンをしています。

- 100ミリをさせ、1000ミリスリープする
- 200ミリをさせ、2000ミリスリープする

```
long[] pattern = {0, 100, 1000, 200, 2000};
```

パターンをりすには、りしをするパターンにインデックスをします。りしをにするには-1をします。

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(pattern, -1); // does not repeat
vibrator.vibrate(pattern,  0); // repeats forever
```

をめる

をめたいは、してください

```
vibrator.cancel();
```

させる

いミリをして、

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(500);
```

オンラインでをむ https://riptutorial.com/ja/android/topic/3359/

247:

き

フラグメントとそのメカニズムの

- void onActivityCreatedBundle savedInstanceState//フラグメントのアクティビティがされ、このフラグメントのビュ―がインスタンスされたときにびされます。
- void onActivityResultint requestCode、int resultCode、Intent data//のstartActivityForResult Intent、intびしのをけります。
- void onAttachActivity activity//このメソッドはAPIレベル23ではです。わりにonAttach Contextをしてください。
- void onAttachコンテキストコンテキスト//フラグメントがにそのコンテキストにアタッチされたときにびされます。
- void onAttachFragmentFragment childFragment//フラグメントがこのフラグメントのとしてアタッチされたときにびされます。
- void onConfigurationChangedConfiguration newConfig//コンポ―ネントのにデバイスのがされたときに、システムによってびされます。
- void onCreateBundle savedInstanceState//フラグメントのをうためにびされます。
- View onCreateViewLayoutInflaterインフレ―タ、ViewGroupコンテナ、Bundle savedInstanceState//フラグメントがユ―ザインタフェ―スビュ―をインスタンスするようにびされます。
- void onDestroy//フラグメントがされなくなったときにびされます。
- void onDestroyViewonCreateViewLayoutInflater、ViewGroup、Bundleによってにされたビュ―がフラグメントからりされたときにびされます。
- void onDetach//フラグメントがそのアクティビティにアタッチされなくなったときにびされます。
- void onInflateアクティビティアクティビティ、AttributeSet attrs、Bundle savedInstanceState//このメソッドはAPIレベル23でされました。わりにonInflateContext、AttributeSet、Bundleをしてください。
- void onInflateコンテキストコンテキスト、AttributeSet attrs、Bundle savedInstanceState//ビュ―レイアウトインフレ―ションのとしてフラグメントがされるときにびされます。は、アクティビティのコンテンツビュ―をします。

- void onPause//フラグメントがもうされなくなったときにびされます。
- void onResume//フラグメントがユーザーにされ、アクティブにされているときにびされます。
- void onSaveInstanceStateBundle outState//のをするようにフラグメントをするためにびされ、でそのプロセスのしいインスタンスでできます。
- void onStart//フラグメントがユーザーにされたときにびされます。
- void onStop//フラグメントがされなくなったときにびされます。
- void onViewStateRestoredBundle savedInstanceState//されたすべてのがフラグメントのビューにされたときにびされます。

フラグメントは、アクティビティーのビヘイビアまたはユーザー・インターフェースのをします。1つのアクティビティでのフラグメントをして、マルチペインのUIをし、のアクティビティでフラグメントをすることができます。フラグメントは、のライフサイクルをち、のイベントをけり、アクティビティがされているにまたはできる「サブアクティビティ」のようなものアクティビティのモジュラセクションとえることができますなるにおける。

コンストラクタ

すべてのフラグメントはのコンストラクタをつがあるため、アクティビティのをするときにインスタンスできます。フラグメントがインスタンスされたときにこれらのコンストラクタがびされることはないため、サブクラスにパラメータをつのコンストラクタがないことをくおめします。そのわりに、はsetArgumentsBundleでびしからされ、でgetArgumentsでFragmentによってできます。

Examples

newInstanceパターン

パラメータをつフラグメントコンストラクタをすることはですが、Androidはフラグメントをするときににのないコンストラクタをびしますたとえば、Androidのでそれらがされたなど。このため、パラメータをつコンストラクタをすることはおめできません。

されるフラグメントがにするようにするには、な`newInstance()`メソッドをしてフラグメントをし、しいインスタンスをするときにできるのパラメータをバンドルにれることができます。

```
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);
    }
  }
}
```

、ので

```
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();
```

このパターンは、なすべてのがにフラグメントににされるようにするためのベストプラクティスです。フラグメントをしてでると、にがされますが、`onSaveInstanceState(Bundle)`をするがあります。

バックスタックとスタティックファブリックパターンをしたフラグメントのナビゲーション

まずにの`Fragment`をするがあります。これは、アクティビティの`onCreate()`メソッドでいます。

```
if (null == savedInstanceState) {
    getSupportFragmentManager().beginTransaction()
      .addToBackStack("fragmentA")
      .replace(R.id.container, FragmentA.newInstance(), "fragmentA")
      .commit();
}
```

に、バックスタックをするがあります。もなは、すべてのFragmentTransactionsでされる、アクティビティでされたをすることです。

```
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();
}
```

に、 `onBackPressed()` をオ―バ―ライドして、バックスタックでなのフラグメントからるときにアプリケ―ションをするがあります。

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

アクティビティでの

```
replaceFragment(FragmentB.newInstance(), "fragmentB");
```

の `MainActivity` がたちのであるとします

```
((MainActivity) getActivity()).replaceFragment(FragmentB.newInstance(), "fragmentB");
```

バンドルをして、アクティビティからフラグメントにデ―タをす

すべてのフラグメントにのコンストラクタつまり、をたないコンストラクタメソッドがです。したがって、するフラグメントにデ―タをすには、 `setArguments()`メソッドをするがあり`setArguments()`。 このメソッドは、デ―タをするバンドルをし、そのバンドルをにします。その、このBundleは、フラグメントの`onCreate()`および`onCreateView()`コールバックでりすことができます。

アクティビティ

```
 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();
```

```
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    String myValue = this.getArguments().getString("message");
    ...
 }
```

コ―ルバック･インタフェ―スをしてアクティビティにイベントをす

フラグメントをアクティビティ―からイベントにするがある、なの1つは、コ―ルバック･インタ―フェ―スをし、ホスト･アクティビティ―がそれをすることをすることです。

フラグメントのボタンがクリックされたときにアクティビティにコ―ルバックをする

まず、コ―ルバックインタフェ―スをします。

```
public interface SampleCallback {
    void onButtonClicked();
}
```

のステップは、このコ―ルバックをフラグメントにりてることです。

```
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;
    }
}
```

に、アクティビティでコールバックをします。

```
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
    }
}
```

フラグメントのをアニメーションする

フラグメントのをアニメーションするために、またはあなたがしてフラグメントをすかにれてのプロセスアニメーションする`FragmentManager`する`FragmentTransaction`。

の`FragmentTransaction`、アニメーションをするは2つあります。アニメーションをすることも、のカスタムアニメーションをすることもできます。

アニメーションは、 `FragmentTransaction.setTransition(int transit)`びし、 `FragmentTransaction`クラスでなされたのいずれかをしてします。では、これらのはのとおりです。

```
FragmentTransaction.TRANSIT_NONE
```

```
FragmentTransaction.TRANSIT_FRAGMENT_OPEN
FragmentTransaction.TRANSIT_FRAGMENT_CLOSE
FragmentTransaction.TRANSIT_FRAGMENT_FADE
```

なトランザクションはのようになります。

```
getSupportFragmentManager()
    .beginTransaction()
    .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

カスタムアニメーションは、 `FragmentTransaction.setCustomAnimations(int enter, int exit)`または`FragmentTransaction.setCustomAnimations(int enter, int exit, int popEnter, int popExit)`いずれかをびしてします。

`enter`と`exit`アニメーションは、バックスタックからフラグメントをポップすることをわない`FragmentTransaction`にしてされます。 `popEnter`と`popExit`バックスタックからのをポップするときのアニメーションがされます。

のコードでは、フラグメントを1つスライディングしてのスライスをスライディングしてきえるをします。

```
getSupportFragmentManager()
    .beginTransaction()
    .setCustomAnimations(R.anim.slide_in_left, R.anim.slide_out_right)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

XMLアニメーションは`objectAnimator`タグをします。 **slide_in_left.xml**のは、 のようになります。

```
<?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>
```

フラグメントの

フラグメントのすべてのはアクティビティでわなければなりません。フラグメントはアクティビティなしではいにできません 。

そののリソース

- OnFragmentInteractionListenerをする

- アンドロイドのとの

このサンプルでは、 `message`をしたりしたりするための`SenderFragment`と`ReceiverFragment` 2つのフラグメントこのはなStringをそれぞれホストする`MainActivity`があります。

`SenderFragment`のボタンは、メッセ―ジをするプロセスをします。 `ReceiverFragment`のTextViewは、メッセ―ジがされるとされます。

MainActivityのスニペットには、なコ―ドをするコメントがあります

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

`MainActivity`のレイアウトファイルは、 `MainActivity`の2つのフラグメントをホストします。

```
<?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 、インタフェースしSendMessageListenerちますMainActivityでボタンときっSenderFragmentクリックされたし。

は、なコードをするSenderFragmentコードスニペットです。

```
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;
}
}
```

`SenderFragment`のレイアウトファイル

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

<Button
    android:id="@+id/bSend"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:text="SEND"
    android:layout_gravity="center_horizontal" />
</LinearLayout>
```

`ReceiverFragment`はシンプルで、シンプルなパブリックメソッドをしてTextViewをします。`MainActivity`からメッセージをけると、 `ReceiverFragment`このパブリックメソッドが`SenderFragment`れます

`ReceiverFragment`コードスニペットは、なコードをするコメントきです

```
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);
}
}
```

`ReceiverFragment`のレイアウトファイル

```
<?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>
```

オンラインでをむ https://riptutorial.com/ja/android/topic/1396/

248: とのピッカー

Examples

マテリアルDatePicker

のをのセクションの`build.gradle`ファイルにしてください。 これはのためののライブラリです

```
  compile 'com.wdullaer:materialdatetimepicker:2.3.0'
```

Button Clickイベントで`DatePicker`をくがあります。

したがって、のようにxmlファイルにボタンを1つしてください。

```
     <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"/>
```

MainActivityではこのようにします。

```
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");

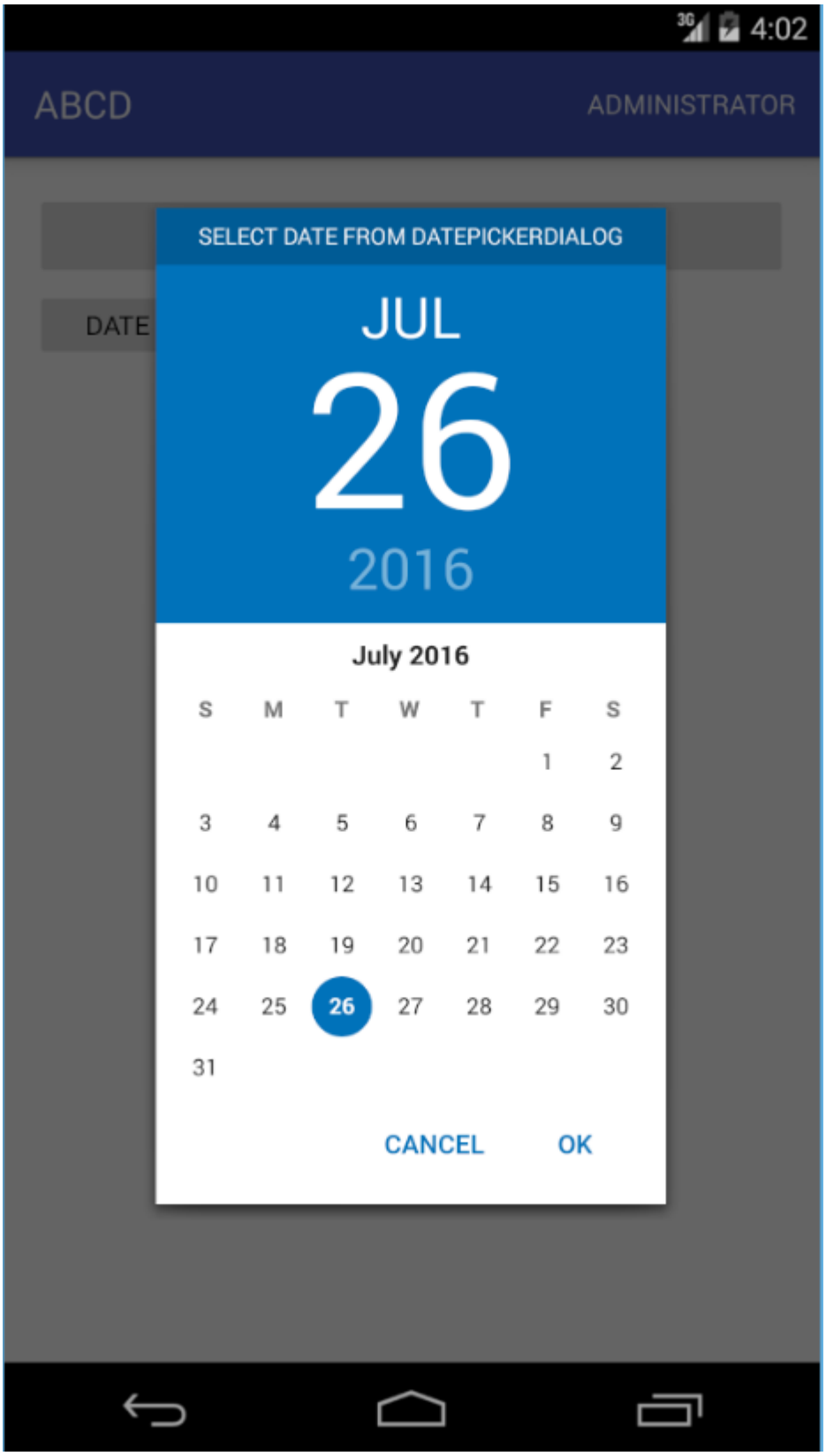
            }
        });
    }

    @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;
    }

}
```



ピッカーダイアログ

これは、 `DatePicker`をしてをするダイアログをします。このダイアログには、のダイアログをするために、コンテキスト、、、がです。ユーザーがをすると、 `DatePickerDialog.OnDateSetListener`でコールバックされます。

```
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();

    }
```

は1の0から12の11までのintであることにしてください

オンラインでとのピッカーをむ https://riptutorial.com/ja/android/topic/2836/とのピッカー

249: のインテント

- インテント
- インテントインテント
- インテントアクション
- インテントストリング·アクション、ウリ·ウリ
- インテントContext packageContext、Class <> cls
- IntentString action、Uri uri、コンテキストpackageContext、クラス<> cls

パラメーター

パラメーター	
o	インテント
アクション	`String: ACTION_VIEW.`などのインテントアクション`ACTION_VIEW.`
ウリ	`Uri:`インテントデータURI。
packageContext	`Context:`このクラスをするアプリケーションパッケージのコンテキスト。
CLS	`Class:`インテントにされるコンポーネントクラス。

- インテントの
- インテントのの

Examples

かつなインテント

なインテントは、じアプリケーションパッケージのアクティビティまたはサービスをするためにされます。この、したクラスのがにされています

```
Intent intent = new Intent(this, MyComponent.class);
startActivity(intent);
```

ただし、そのをできるユーザーのデバイスにインストールされているアプリケーションの、のインテントがシステムにされます。これは、なるアプリケーションでをするためにされます。

```
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
}
```

いのについては、Androidデベロッパーのこちらのドキュメントをごください インテント

のインテント

インテントは、のコンポーネントのをするのではなく、するなアクションをします。これにより、のアプリのコンポーネントがそのコンポーネントをできるようになります。

たとえば、ユーザーにのをするは、のインテントをして、のなアプリがマップのされたをするようにすることができます。

```
// 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);
}
```

オンラインでのインテントをむ https://riptutorial.com/ja/android/topic/5336/のインテント

250: されたVideoView

き

してビデオをするVideoViewSurfaceViewListViewユーザがリストをスクロールしようとするまでは、にしているようです。リストがスクロールしめると、ビデオはにわりますをすることもあります。それはバックグラウンドでをけますが、ビデオのりのをブラックボックスとしてレンダリングするので、もうることはできません。カスタムのOptimized VideoViewをすると、InstView、Facebook、TwitterのようにListViewスクロールでがされます。

Examples

ListViewでされたVideoView

カスタムVideoViewをパッケージにれておくがあります。

カスタムビデオビューのレイアウト

```
<your.packagename.VideoView
    android:id="@+id/video_view"
    android:layout_width="300dp"
    android:layout_height="300dp" />
```

カスタムされたVideoViewコード

```
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);
    }
}
```

このgitubリポジトリからのヘルプ。 3にかれたようにいくつかのがありますが、はのようにですることができました。

オンラインでされたVideoViewをむ https://riptutorial.com/ja/android/topic/10638/されたvideoview

251:

き

マテリアルデザインは、プラットフォームやデバイスでのビジュアル、モーション、インタラクションデザインのなガイドです。

デザインサポートライブラリをしているオリジナルのAndroidブログもごください

https://developer.android.com/design/material/index.html

マテリアルデザインのガイドライン

https://material.io/guidelines

そののリソースとライブラリ

https://design.google.com/resources/

Examples

AppCompatテーマをする

AppCompatサポートライブラリは、 Material Designでアプリケーションをするためのテーマをします。 `Theme.AppCompat`をつテーマは、Activityで`AppCompatActivity`を`AppCompatActivity`ためにもです。

のステップは、テーマのカラーパレットをカスタマイズして、アプリをにづけすることです。あなたのアプリの`res/styles.xml`をすることができます

```
<!-- 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>
```

がい`Theme.AppCompat`わりに、 `Theme.AppCompat.Light`または`Theme.AppCompat.Light.DarkActionBar`することもできます。

のでテーマをカスタマイズできます。 マテリアルデザインのカラーチャートとマテリアルパレッ

トのがあります。 "500"カラーはプライマリこのではブルー500にしています。いものにはじの"700"をんでください。アクセントのとはなるいのいがあります。なは、アプリのツールバーと、のアプリのエントリ、ステータスバーのをくするバリアント、およびのコントロールをするアクセントのにされます。

このテーマをしたら、それを`AndroidManifest.xml`アプリケーションにし、のアクティビティにテーマをします。これは、 `AppTheme.NoActionBar`テーマをするにです。このテーマをすると、デフォルトのツールバーをできます。

```
<application android:theme="@style/AppTheme"
    ...>
    <activity
        android:name=".MainActivity"
        android:theme="@style/AppTheme" />
</application>
```

また、 `android:theme`とテーマ`ThemeOverlay`テーマをして、々のビューにテーマをすることもできます。たとえば、 `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" />
```

または`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>
```

ツールバーの

`Toolbar`は、アプリケーションレイアウトでするための`ActionBar`されたものです。 `ActionBar`はに、フレームワークによってされる`Activity's`ウィンドウデコレーション`Activity's`ですが、ビューののネストレベルに`Toolbar`をすることができます。のをしてできます。

1. のであなたのモジュールえばappの **build.gradle**ファイルにのがされていることをしてください

   ```
   compile 'com.android.support:appcompat-v7:25.3.1'
   ```

2. あなたのアプリのテーマを`ActionBar`をたないテーマにしてください。これをうには、

`res/values`で**styles.xml**ファイルをし、 `Theme.AppCompat`テーマをします。
このでは、している`Theme.AppCompat.NoActionBar`あなたのとして`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>
```

`Theme.AppCompat.Light.NoActionBar`または`Theme.AppCompat.DayNight.NoActionBar` 、またはに`ActionBar`をたないのテーマをすることもでき`ActionBar`

3. アクティビティレイアウトに`Toolbar`をする

```
<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"/>
```

`Toolbar`に、りのレイアウトをすることができます。

4. `Activity` 、この`Activity ActionBar Toolbar`として`Toolbar`をし`Toolbar`。 *appcompat*ライブラリと`AppCompatActivity`をしているは、 `setSupportActionBar()`メソッドをします。

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

    final Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    //...
}
```

のをした、 `getSupportActionBar()`メソッドをして、 `ActionBar`としてされている`Toolbar`をできます。

たとえば、タイトルをのようにできます。

```
getSupportActionBar().setTitle("Activity Title");
```

たとえば、のようにタイトルとをすることもできます。

```
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)));
```

FloatingActionButtonFABをする

マテリアルデザインでは、 フローティングアクションボタンがアクティビティのアクションをします 。
それらは、UIのにかんでいるいアイコンによってされ、モーフィング、、およびアンカーポイントのをむモーションビヘイビアをっています。

のでアプリケーションのbuild.gradleファイルにのがされていることをしてください。

```
compile 'com.android.support:design:25.3.1'
```

に、レイアウトファイルに`FloatingActionButton`をします。

```
<android.support.design.widget.FloatingActionButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_margin="16dp"
      android:src="@drawable/some_icon"/>
```

`src`はアクションにするアイコンをします。



はのようになりますアクセントのがマテリアルピンクであるとします。

デフォルトでは、 `FloatingActionButton`はテーマのアクセントカラーにされます。また、`FloatingActionButton`は、にするには、にがです。ボトムのマージンは、のは`24dp` 、タブレットのは`16dp`です。

`FloatingActionButton`さらにカスタマイズするためにできるプロパティはの
`xmlns:app="http://schemas.android.com/apk/res-auto`
`xmlns:app="http://schemas.android.com/apk/res-auto`はレイアウトののとしてされているものとします。

- **`app:fabSize`** にでき`normal`または`mini`サイズのまたはさいバージョンをりえること。
- **`app:rippleColor`** `FloatingActionButton`リップルエフェクトのをします。カラーリソースまたは16にすることができます。
- **`app:elevation`** 、、ブール、カラー、、ディメンションをできます。
- **`app:useCompatPadding`** compatめみをにします。たぶんのようなブール、 `true`または`false`。いapiレベルでしたをするために、api-21でcompatめみをする`true`にします。

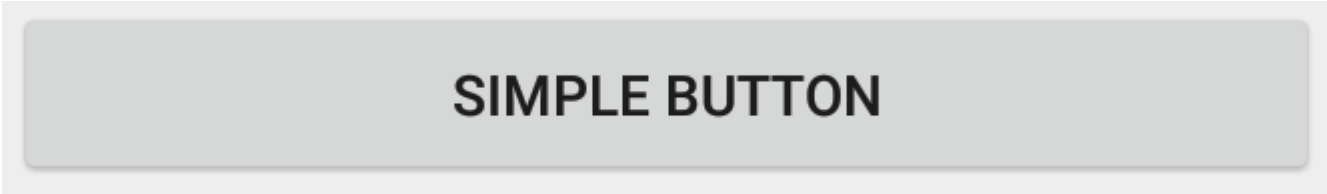
ここで FABのをつけることができます。

マテリアルデザインでスタイリングされたボタン

AppCompatサポートライブラリは、 ボタンのいくつかのなスタイルをします 。それぞれのスタイルは、 `AppCompat`テーマをしている、デフォルトですべてのボタンにされる`Widget.AppCompat.Button`スタイルをします。このスタイルは、 Material Designにって、すべてのボタンがデフォルトでじにえるようにします。

この、アクセントのはピンクです。

1. なボタン `@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. きボタン `@style/Widget.AppCompat.Button.Colored`
`Widget.AppCompat.Button.Colored`スタイルは、`Widget.AppCompat.Button`スタイルを、にアプリのテーマでしたアクセントカラーをします。




```
<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"/>
```

メインテーマのアクセントカラーをせずにをカスタマイズしたいは、 `Button` カスタムテーマ `ThemeOverlay`テーマををし、ボタンの`android:theme`にりてることができます

```
<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"/>
```

`res/values/themes.xml`テーマをし`res/values/themes.xml`。

```
<style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
     <item name="colorAccent">@color/my_color</item>
</style>
```

3. フチしボタン `@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. ボーダーレスカラーボタン `@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"/>
```

TextInputLayoutのい

のでアプリケーションの`build.gradle`ファイルにのがされていることをしてください。

```
compile 'com.android.support:design:25.3.1'
```

がされると、EditTextからヒントをラベルとしてします。

```
<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>
```

パスワードアイアイコンをTextInputLayoutでするために、のコードをすることができます

```
<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>
```

`app:passwordToggleEnabled="true" android:inputType="textPassword"`パラメータはです。

`app`は`xmlns:app="http://schemas.android.com/apk/res-auto"`をするがあります

のトピックでとをつけることができます。

TabLayoutの

TabLayoutはタブをするためのレイアウトをし、にViewPagerとされます。

のでアプリケーションの`build.gradle`ファイルにのがされていることをしてください。

```
compile 'com.android.support:design:25.3.1'
```

これで、 TabItemクラスをして、レイアウトのTabLayoutにをできます。

えば

```
<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>
```

`TabLayout`タブが//されたときにされる`OnTabSelectedListener`を`OnTabSelectedListener`します。

```
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) {

    }
});
```

`TabLayout`からタブをプログラムで/することもできます。

```
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`はとスクロールの2つのモードがあります。

```
tabLayout.setTabMode(TabLayout.MODE_FIXED);
tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);
```

これらはXMLでもできます

```
<android.support.design.widget.TabLayout
    android:id="@+id/tabLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:tabMode="fixed|scrollable" />
```

`TabLayout`モードはにです。つまり、にアクティブにできるのは1つだけです。

タブインジケータのは、マテリアルデザインのテーマにされているアクセントのです。`styles.xml`カスタムスタイルをし、そのスタイルを`styles.xml`にすることで、このをオーバーライドできます。

```
<style name="MyCustomTabLayoutStyle" parent="Widget.Design.TabLayout">
    <item name="tabIndicatorColor">@color/your_color</item>
</style>
```

に、をしてビュ―にスタイルをできます。

```
<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

Ripple TouchエフェクトはAndroid 5.0APIレベル21のデザインでされ、アニメ―ションはしい RippleDrawableクラスによってされました。

> にしてをすなもの。えられたにするリプルのアンカ―は、 `setHotspot(float x, float y)` をするでびすことによってすることができる。

5.0

に、 のボタンのリップルは、API 21ではデフォルトでし 、そののタッチなビュ―では、をすることでできます。

```
android:background="?android:attr/selectableItemBackground">
```

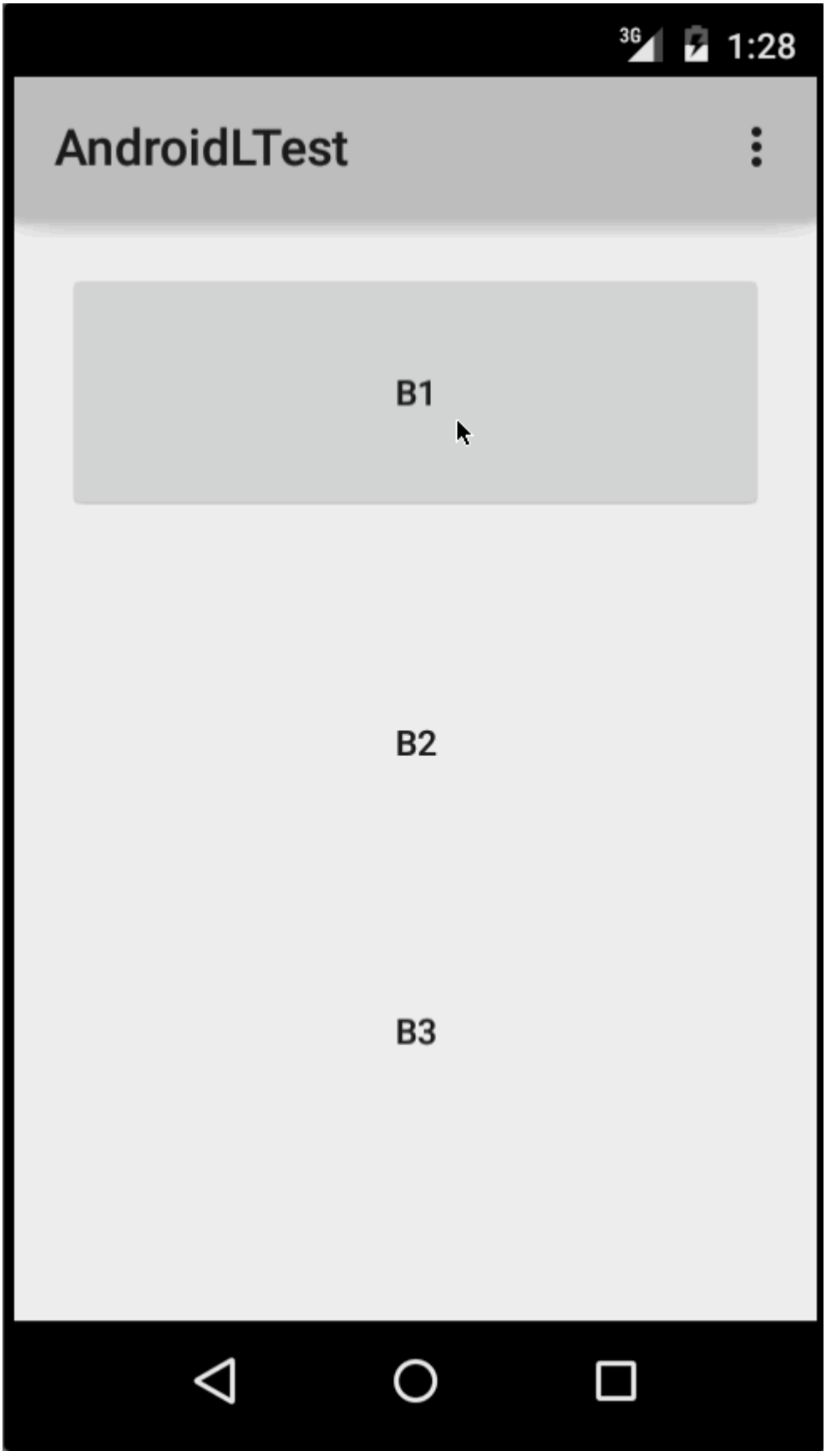
ビュ―にまれるや、

```
android:background="?android:attr/selectableItemBackgroundBorderless"
```

ビュ―のをえてがるのために。

たとえば、のでは、

- B1はバックグラウンドのないボタンで、
- B2は`android:background="android:attr/selectableItemBackground"`されます `android:background="android:attr/selectableItemBackground"`
- B3は`android:background="android:attr/selectableItemBackgroundBorderless"`されています `android:background="android:attr/selectableItemBackgroundBorderless"`



http : //blog.csdn.net/a396901990/article/details/40187203

のコードをしてじことができます。

```
int[] attrs = new int[]{R.attr.selectableItemBackground};
TypedArray typedArray = getActivity().obtainStyledAttributes(attrs);
```

```
int backgroundResource = typedArray.getResourceId(0, 0);
myView.setBackgroundResource(backgroundResource);
```

リップルは、とじで、 android:foregroundをしてビューにすることもできます。そのがすように、リップルがフォアグラウンドにされた、されたのビューのにリップルがされます ImageView 、のビューをむLinearLayoutなど。

リップルをビューにカスタマイズするは、drawableディレクトリにしいXMLファイルをするがあります。

いくつかのがあります

1 のリップル

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#ffff0000" />
```

2 マスクとのリップル

```
<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>
```

するview にでされたをつshape 、 cornersびのタグ、するビューにリップルをするためのmask layer 、ビューのとしてリップルをします。

```
 <?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>
```

3 なリソースのリップル

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#ff0000ff">
    <item android:drawable="@drawable/my_drawable" />
</ripple>
```

のビューにリップルxmlファイルをするには、のようにバックグラウンドとしてしますリップルフ

ァイルのが`my_ripple.xml`とします。

```
<View
    android:id="@+id/myViewId"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/my_ripple" />
```

セレクタ

ターゲットバージョンがv21またはそれであれば、リップルドロアブルをカラーリストセレクタのわりにすることもできますリップルセレクタを`drawable-v21`フォルダにすることもできます。

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

この、ビューのデフォルトのはで、されたはリップルをします。

`?android:colorControlHighlight`をすると、アプリのビルトインリップルとじがリップルにされます。

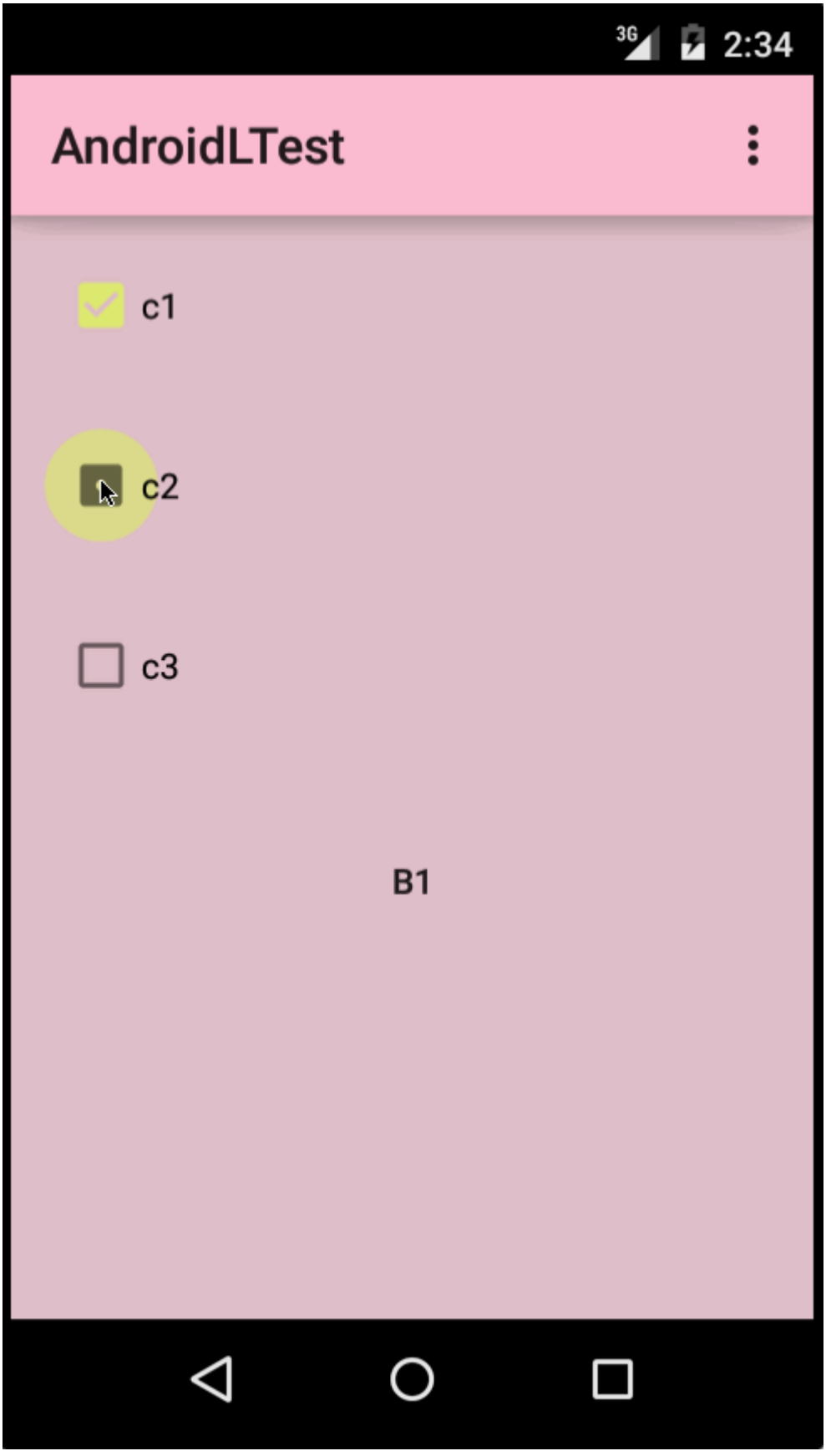
リップルのだけをするには、あなたのテーマでの`android:colorControlHighlight`カスタマイズすることができます

```
<?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>
```

あなたのなどでこのテーマをしてください。はののようになります



http : //blog.csdn.net/a396901990/article/details/40187203

ナビゲーションドロアをする

ナビゲーションドロワーは、アプリのトップレベルのにナビゲートするためにされます。

のでbuild.gradleファイルにデザインサポートライブラリをしたことをしてください

```
 dependencies {
    // ...
    compile 'com.android.support:design:25.3.1'
}
```

に、XMLレイアウトリソースファイルにDrawerLayoutとNavigationViewします。
DrawerLayoutは、のナビゲーションドロワーであるNavigationViewのからスライドさせることをにするらしいコンテナです。モバイルデバイスの、きしサイズは320dpです。

```
<!-- 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>
```

さあ、むなら、あなたのナビゲーションドロワのになるヘッダファイルをしてください。これは、はるかにエレガントなをきしにえるためにされます。

```
<!-- 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>
```

これは、 `app:headerLayout="@layout/drawer_header"`の`NavigationView`タグでされます。
この`app:headerLayout`は、されたレイアウトをヘッダーににします。わりに、にのでることもできます。

```
// 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);
```

ナビゲーション・ドロワーににしたナビゲーションをにするには、メニュー・ファイルをし、にじてをします。アイテムのアイコンはではありませんが、 マテリアルデザインではされています。
これは、 `app:menu="@menu/navigation_menu" attribute`の`NavigationView`タグでされ

`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>
```

アイテムをグループにけるには、 `android:title`をつの`<item>`ネストされた`<menu>`にれるか、`<group>`タグでラップします。

レイアウトがし`Activity`、 `Activity`コードにします。

```
  // 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();
```

これで`NavigationView`ヘッダービューでなをうことができます

```
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);
```

ヘッダービューはの`View`とじようにするので、 `findViewById()`をしてレイアウトファイルにの`View`をすると、そののすべてのプロパティをできます。

のトピックでとをつけることができます。

デザインサポートライブラリのボトムシート

のシートは、のからスライドして、よりくのコンテンツをします。
それらはv25.1.0バージョンのAndroidサポートライブラリにされ、すべてのバージョンよりもサポートされています。

のでアプリケーションのbuild.gradleファイルにのがされていることをしてください。

```
compile 'com.android.support:design:25.3.1'
```

なボトムシート

あなたは`CoordinatorLayout`ビューに`BottomSheetBehavior`をけるPersistent Bottom Sheetをることができます

```
<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>
```

に、コードでのようにをすることができます。

```
// The View with the BottomSheetBehavior
View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

setStateメソッドをして、 BottomSheetBehaviorのをできます。

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

のいずれかのをできます。

- `STATE_COLLAPSED` このりたたまれたはデフォルトであり、レイアウトのだけがにってされます。さは、 `app:behavior_peekHeight`デフォルトは0でできます。
- `STATE_EXPANDED` ボトムシートのにされた。ボトムシートがされますさが`CoordinatorLayout`をむ。または`CoordinatorLayout`がりつぶされている
- `STATE_HIDDEN` デフォルトではになっており `app:behavior_hideable`でになっています、これをにするとのシートをにスワイプしてのシートをにすことができます

さらに、したビューをクリックしてBottomSheetをいたりじたりするには、A Buttonはシートのをりえてビューをするをします。

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

しかし、 `BottomSheet`ビヘイビアには、ユーザーがスワイプUPまたはDRAGモーションをしてスワイプをできるもあります。このような、シートがされた、ビューをすることはできませんのボタンのように。そのため、のコールバックをしたいので、 `BottomSheetCallback`をしてユーザーのスワイプイベントをリッスンすることができます

```
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
   }
 });
```

また、ボトムシートをCOLLAPSEDとEXPANDEDモードでのみさせ、トグルしてしないようにしたいは

```
mBottomSheetBehavior2.setHideable(false);
```

ボトムシートDialogFragment

のシートのビューのわりにBottomSheetDialogFragmentをすることもできます。これをうには、まずBottomSheetDialogFragmentをするしいクラスをするがあります。

`setupDialog()`メソッドで、しいレイアウトファイルをし、アクティビティのコンテナビューのBottomSheetBehaviorをできます。ビヘイビアがあれば、 BottomSheetCallbackをしてけることで、シートがになったときにFragmentをじることができます。

```
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);
        }
    }
}
```

に、フラグメントのインスタンスにしてshowをびして、そのフラグメントをのシートにすることができます。

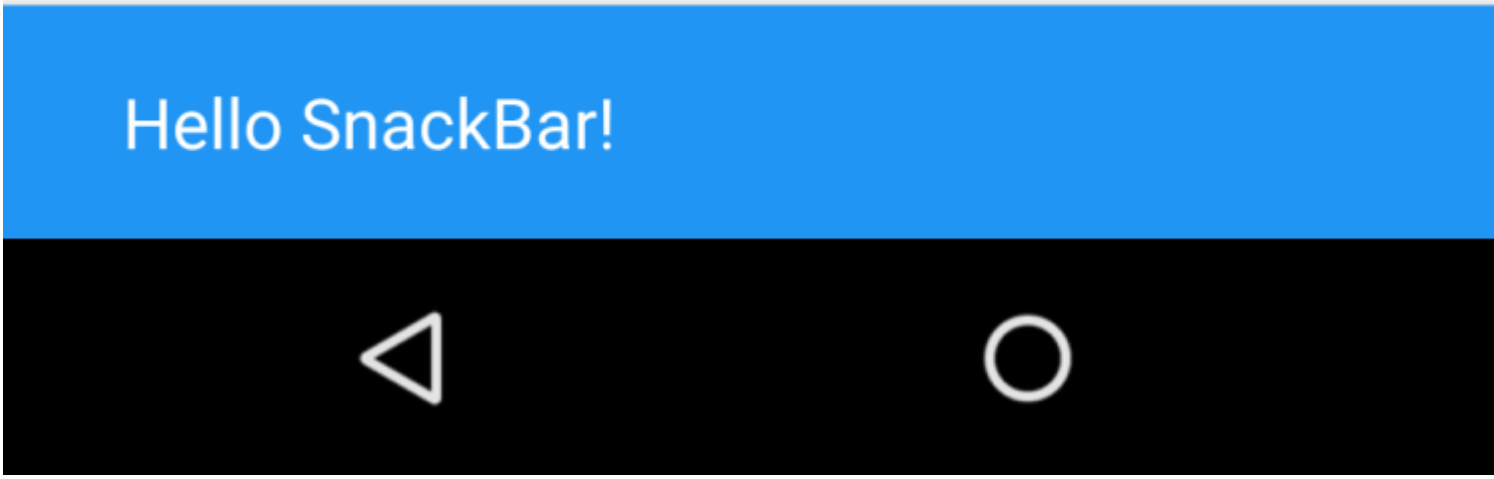
```
BottomSheetDialogFragment bottomSheetDialogFragment = new BottomSheetDialogFragmentExample();
bottomSheetDialogFragment.show(getSupportFragmentManager(),
bottomSheetDialogFragment.getTag());
```

のトピックでをつけることができます

スナックバーをする

マテリアルデザインのなの1つは、の`Toast`にきえる`Snackbar`のです。 Androidのドキュメントにって

> スナックバーには、されたにする1のテキストがまれています。テキストアクションはまれていますが、アイコンはありません。トーストは、にシステムメッセージングにされます。また、のにされますが、でスワイプされることはありません。



Androidにメッセ―ジをするためにトーストをすることはできますが、アプリでデザインをすることをめたは、にスナックバ―をすることをおすすめします。にオ―バ―レイとしてされるのではなく、から`Snackbar`がされます。

ここでそれがどのようにわれます

```
Snackbar snackbar = Snackbar
        .make(coordinatorLayout, "Here is your new Snackbar", Snackbar.LENGTH_LONG);
snackbar.show();
```

`Snackbar`をするのさについては、 `Toast`によってされるものとのオプションがあります。または、カスタムのをミリですることもできます。

- `LENGTH_SHORT`
- `LENGTH_LONG`
- `LENGTH_INDEFINITE`
- `setDuration()` バ―ジョン`22.2.1`

また、あなたになをすることができます`Snackbar`など`ActionCallback`またはカスタムカラ―。ただし、 `Snackbar`カスタマイズするにAndroidがするデザインガイドラインにはしてください 。

しかし、 `Snackbar`には1つのがあります。 `Snackbar`をするビュ―のレイアウトは、`CoordinatorLayout`であるがあります。これは、のポップアップをからできるようにするためです。

レイアウトXMLファイルで`CoordinatorLayout`をするはのとおりです。

```
<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>
```

`CoordinatorLayout`は、アクティビティの`onCreate`メソッドでしてから、 `Snackbar`をするときにするがあり`Snackbar`。

`Snackbar`についてのは、 またはトピックをしてください。

オンラインでをむ https://riptutorial.com/ja/android/topic/124/

252: プロセッサ

き

アノテーションプロセッサは、コンパイルにをスキャンしてするためのjavacのツールビルドです。

アノテーションは、クラス、メソッド、フィールド、さらにはそののアノテーションにけることができるメタデータのクラスです。アノテーションプロセッサをして、リフレクションおよびコンパイルにこれらのアノテーションににアクセスする2つのがあります。

Examples

@NonNullアノテーション

```
public class Foo {
    private String name;
    public Foo(@NonNull String name){...};
    ...
}
```

ここで@NonNullは、のがnullパラメータをとすることをするためにアンドロイドスタジオによってコンパイルされたにされるアノテーションです。

のタイプ

アノテーションには3つのタイプがあります。

1. マーカーアノテーション - メソッドをたないアノテーション

   ```
   @interface CustomAnnotation {}
   ```

2. アノテーション - 1つのメソッドをつアノテーション

   ```
   @interface CustomAnnotation {
       int value();
   }
   ```

3. の - のをつ

   ```
   @interface CustomAnnotation{
       int value1();
       String value2();
       String value3();
   }
   ```

カスタムのと

カスタムアノテーションをするには、

- ターゲット - これらのアノテーションは、フィールドレベル、メソッドレベル、タイプレベルなどでします。
- - どのレベルのがになるか。

このために、カスタムアノテーションをみみました。これらはにされているものをチェックしてください

@ターゲット

Element Types	Where the a
TYPE	class, interfac
FIELD	fields
METHOD	methods
CONSTRUCTOR	constructors
LOCAL_VARIABLE	local variable
ANNOTATION_TYPE	annotation ty
PARAMETER	parameter

@

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

カスタムの

```
@Retention(RetentionPolicy.SOURCE) // will not be available in compiled class
@Target(ElementType.METHOD) // can be applied to methods only
@interface CustomAnnotation{
      int value();
}
```

カスタムアノテーションの

```
class Foo{
  @CustomAnnotation(value = 1)  // will be used by an annotation processor
  public void foo(){..}
}
```

`@CustomAnnotation`でされるは、Annotationprocessorによってされ、コンパイルなどにコードをすることがあります。

オンラインでプロセッサをむ https://riptutorial.com/ja/android/topic/10726/プロセッサ

253: の

き

アクティビティとは、ドライブがされたときにポイントをる、がしているときにLANをオフにする、ユーザーがいるときににするなどの、デバイスでのアクションをするためのユーザーののですく。

Examples

Google Play ActivityRecognitionAPI

これは、GooglePlayサービスのActivityRecognitionApiのいのなです。これはらしいですが、Google Playサービスがインストールされていないではしません。

ActivityRecognition APIのドキュメント

マニフェスト

```
<!-- 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) {

    }
}
```

アクティビティレシーバ

```
public class ActivityReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

LocalBroadcastManager.getInstance(context).sendBroadcast(intent.setAction("activity"));
    }
}
```

PathSenseアクティビティの

PathSenseのアクティビティのは、Google Playサービスをたないデバイスでは、のアクティビティモデルをしているため、 http://developer.pathsense.comにしてAPIキーとクライアントIDをするがあります。

マニフェスト

```
<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);
    }
}
```

オンラインでのをむ https://riptutorial.com/ja/android/topic/9831/の

254: いリンクの

き

ディープリンクとは、ユーザーをアプリののコンテンツにするURLです。インテントフィルタをし、インテントからデータをして、ユーザーをアプリのしいにすることで、ディープリンクをできます。

パラメーター

<data>	
スキーム	URIのスキームをする。 http 、 https 、 ftp
ホスト	URIのホストをする。 google.com 、 example.org
	URIのポート。 80 、 443
パス	URIのパス。 /まるがあります。 / 、 /about
pathPrefix	URIのパスの。 /item 、 /article
パスパターン	URIのパスにするパターン。 /article/[0-9]* /item/.* /article/[0-9]*
mimeType	するMIMEタイプ。 image/jpeg 、 audio/*

<intent-filter>

この<action>と<category>のみわせは、ユーザーがのアプリケーションのリンクをクリックしたときにのアクティビティをするがあることをAndroidシステムにするものです。

```
<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>
```

の<data>タグ

<intent-filter>サポートするのディープリンクは、そのインテントフィルタでするすべての<data>のです。のドメイン、のパス、のスキームのがこれをしています。

リソース

- アプリコンテンツのディープリンクをにする developer.android.com
- `<intent-filter>` developer.android.com

Examples

シンプルなディープリンク

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>
```

これにより、`MainActivity`をするためのいリンクとして`http://www.example.com`からまるすべてのリンクがけれられます。

1つのドメインののパス

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>
```

ユーザーがのいずれかのリンクをクリックすると`MainActivity`がします

- http://www.example.com/
- http://www.example.com/about
- http://www.example.com/map

のドメインとのパス

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>
```

ユーザーがのいずれかのリンクをクリックするとMainActivityがします

- http://www.example.com/
- http://www.example2.com/
- http://www.example.com/map
- http://www.example2.com/map

じドメインの**http**と**https**の

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>
```

ユーザーがのいずれかのリンクをクリックするとMainActivityがします

- http://www.example.com/
- https://www.example.com/
- http://www.example.com/map
- https://www.example.com/map

クエリパラメータの

```
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");
        }
    }

}
```

ユーザーが`http://www.example.com/map?param1=FOO&param2=BAR`へのリンクをクリックすると、 `param1`のは`"FOO"`なり、 `param2`のは`"BAR"`ます。

pathPrefixの

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>
```

これにより、ユーザーが`http://www.example.com/item`でまるリンクをクリックするとMainActivityがします。

- `https://www.example.com/item`
- `http://www.example.com/item/1234`
- `https://www.example.com/item/xyz/details`

オンラインでいリンクのをむ https://riptutorial.com/ja/android/topic/3716/いリンクの

255:

き

****このマニュアルはされておらず、にです****

グライドのはもっといです

Glide v4については、 http: //bumptech.github.io/glide/をしてください。 Glide v3については、 https://github.com/bumptech/glide/wikiをしてください 。

Glideは、メディアのデコード、メモリとディスクのキャッシュ、およびリソースのプーリングをシンプルでいやすいインタフェースにまとめた、Androidのでなオープンソースのメディアおよびみみフレームワークです。

Glideは、ビデオ、、アニメーションGIFのりし、デコード、およびをサポートしています。
Glideには、がほぼすべてのネットワークスタックにプラグインできるなAPIがまれています。
デフォルトでは、Glideはカスタム HttpUrlConnectionベースのスタックをしますが、ユーティリティライブラリをGoogleのVolleyプロジェクトやSquareのOkHttpライブラリにプラグインします。

Glideのなは、あらゆるののスクロールをなりスムーズかつにうことですが、Glideは、リモートを、サイズ、およびするがあるほとんどすべてのにもです。

ソースコードとそののドキュメントはGitHubでできます https : //github.com/bumptech/glide

Examples

あなたのプロジェクトにグライドをする

から

Gradleの

```
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'
}
```

Mavenの

```
<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>
```

ProGuardDexGuardのとによっては、proguard.cfgにのをするがありますはGlideのwikiを。

```
-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
```

のみみ

ImageView

されたURL、Uri、リソースID、またはそののモデルから`ImageView`をロードするには

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
String yourUrl = "http://www.yoururl.com/image.png";

Glide.with(context)
    .load(yourUrl)
    .into(imageView);
```

Urisの、 `yourUrl`をあなたのUri `content://media/external/images/1` にきえます。 `yourUrl`、 `yourUrl` をリソースID `R.drawable.image` にきえます。

RecyclerViewとListView

ListViewまたはRecyclerViewでは、まったくじをできます。

```
@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);
}
```

`onBindViewHolder`でみみをしないは、 `ImageView`するに`ImageView` Glideがしているがあることをしてください`clear()`。

```
@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);
    }
}
```

グライドサークル**ImageView**でイメージをロード

グライドをしてをします。

```
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();
    }
}
```

```
Glide.with(context)
    .load(yourimageurl)
    .transform(new CircleTransform(context))
    .into(userImageView);
```

デフォルトの

グライドには、2つのデフォルト、フィットとがまれます。

フィットセンター

```
Glide.with(context)
    .load(yourUrl)
    .fitCenter()
    .into(yourView);
```

フィットセンターは、AndroidのScaleType.FIT_CENTERとじをします。

センター

```
Glide.with(context)
    .load(yourUrl)
    .centerCrop()
    .into(yourView);
```

Center cropはAndroidのScaleType.CENTER_CROPとじをします。

については、 Glideのwikiをしてください。

カスタムグライドターゲットでみをびたコーナーをらせる

まずユーティリティクラスをするか、このメソッドをなクラスでします

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

をみむ

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

asBitmapをするので、アニメーションはされます。このでのアニメーションをするには、animateメソッドをします。

デフォルトのグライドアニメーションにのフェードインをした

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .animate(R.anim.abc_fade_in)
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

このアニメーションは、サポートライブラリのプライベートリソースであることにしてください。またはされるがあるため、することはおめできません。

また、 RoundedBitmapDrawableFactoryをするにはサポートライブラリがです

のプリロード

リモートイメージをプリロードし、イメージがだけダウンロードされるようにするには

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE)
    .preload();
```

に

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE) // ALL works here too
    .into(imageView);
```

ローカルイメージをプリロードし、されたコピーがディスクキャッシュおよびおそらくメモリキャッシュにあることをするには

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // Or whatever transformation you want
    .preload(200, 200); // Or whatever width and height you want
```

に

```
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);
```

プレ―スホルダとエラ―

みみにDrawableをするは、プレ―スホルダをできます。

```
Glide.with(context)
    .load(yourUrl)
    .placeholder(R.drawable.placeholder)
    .into(imageView);
```

ロ―ドがらかのでしたにDrawableをするには

```
Glide.with(context)
    .load(yourUrl)
    .error(R.drawable.error)
    .into(imageView);
```

nullモデルURL、Uri、ファイルパスなどをするにDrawableをするには

```
Glide.with(context)
    .load(maybeNullUrl)
    .fallback(R.drawable.fallback)
    .into(imageView);
```

カスタムなしでImageViewにイメ―ジをロ―ドする

カスタムBitmapImageViewTargetをして、イメ―ジをロ―ドします。

```
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);
    }
}
```

```
Glide
    .with(context)
    .load(yourimageidentifier)
    .asBitmap()
    .into(new CircularBitmapImageViewTarget(context, imageView));
```

グライドのみにしました

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

ここで`SET_YOUR_DEFAULT_IMAGE`あなたは、のデフォルトすることができ`Drawable`。 のみみにした、このがされます。

オンラインでをむ https://riptutorial.com/ja/android/topic/1091/

256: れ

き

Leak Canaryは、アプリケーションのリークをするためにされるAndroidおよびJavaライブラリです

のリンクでそのをることができます

https://github.com/square/leakcanary

Examples

Androidアプリケーションでリークカナリアをする

*build.gradle*ではのをするがあります

```
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'
```

`Application`クラスでは、のコードを`onCreate()`にするがあります。

```
LeakCanary.install(this);
```

*LeakCanary*のためになのはこれだけです。あなたのビルドにリークがあると、にがされます。

オンラインでれをむ https://riptutorial.com/ja/android/topic/10041/れ

257:

Examples

サイズをせずにをする

シングルトンクラスからビットマップをする

```
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;
    }
}
```

ビットマップをしたのはじです。

どうやってチェックしたの

```
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());
```

```
Before Compress : Dimension: 1080-1452
After Compress : Dimension: 1080-1452
```

オンラインでをむ https://riptutorial.com/ja/android/topic/5588/

258: /の

Examples

Android Nougatでされたがされました。

マニフェストまたはエレメントにこのをして、マルチウィンドウをまたはにします。

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

このがtrueにされている、アクティビティはとのモードでできます。がfalseにされている、アクティビティはマルチウィンドウモードをサポートしません。このがfalseで、ユーザがマルチウィンドウモードでアクティビティをしようとすると、アクティビティはをきぎます。

アプリがAPIレベル24をターゲットにしていても、こののをしていない、のはデフォルトでtrueになります。

のコードは、アクティビティがフリーフォームモードでされているときに、アクティビティのデフォルトのサイズと、およびサイズをするをしています。

```
<--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>
```

マルチウィンドウモードでの

のは、のアクティビティやアプリケーションとデバイスをしているのあるアクティビティにはをなさないため、デバイスがマルチウィンドウモードになっていると、またはされます。そのようなには、

1. のシステムUIのカスタマイズオプションはになっています。たとえば、アプリケーションがフルスクリーンモードでされていない、ステータスバーをにすることはできません。
2. システムは**androidscreenOrientation**のをします。

アプリの**API**レベルが**23**の

アプリのAPIレベルが23で、ユーザーがマルチウィンドウモードでアプリをしようとすると、アプリがをしていないり、アプリはアプリのサイズをにします。

アプリがをしていないは、Android 7.0をしたデバイスでアプリをし、アプリをモードにするがあ

ります。アプリのサイズをにした、ユーザーエクスペリエンスがけれであることをします。

アプリがをしているは、アプリをマルチウィンドウモードにしてください。これをうと、アプリがフルスクリーンモードのままであることをします。

オンラインで/のをむ https://riptutorial.com/ja/android/topic/7130/-の

259:

Examples

スレッドのとその

アプリケーションをすると、まずメインスレッドがされます。このメインスレッドは、アプリケーションのすべてのUIコンセプトをします。 UIをとしないタスクをくしたいは、そのタスクをバックグラウンドでするためにthreadをします。

ブロックをするスレッドのをにします。

```
new Thread(new Runnable() {
    public void run() {
        for(int i = 1; i < 5;i++) {
            System.out.println(i);
        }
    }
}).start();
```

スレッドをするための`Thread.run()`メソッドをつThreadオブジェクトをしてスレッドをすることができます。ここで`run()`メソッドは`start()`メソッドによってびされます。

MultiThreadingとばれるのスレッドをしてすることもできます。このスレッドは、のスレッドがされただけスリープにをするスリープもえています。しかし、sleepはInterruptedExceptionをスローします。したがって、このようにtry / catchをしてするがあります。

```
try{Thread.sleep(500);}catch(InterruptedException e){System.out.println(e);}
```

バックグラウンドスレッドからのUIの

ネットワークやされるタスクをするためのバックグラウンドスレッドをし、にじてをUIでするのがです。

メインスレッドだけがUIをできるので、これはをきこします。

は、 `runOnUiThread()`メソッドをすることです。バックグラウンドスレッドからUIスレッドでコードをできるようにするためです。

このなでは、アクティビティのにスレッドがされ、マジックナンバー`42`がランダムにされるまでされ、 `runOnUiThread()`メソッドをしてこのがたされるとUIがされます。

```
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();
    }
}
```

オンラインでをむ https://riptutorial.com/ja/android/topic/7131/

260: くずの

Lintツールは、、セキュリティ、パフォーマンス、ユーザビリティ、アクセシビリティ、およびのためのなバグやのについて、Androidプロジェクトのソースファイルをチェックします。 Lintは、コマンドラインやAndroid Studioからできます。

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

Examples

ツールをうxmlファイルをする

`tools:ignore`は、lintのをするためにxmlファイルでできます。

しかし、このテクニックでくずのをすることは、ほとんどの、ったでむことです。

くずのはされ、されなければなりません...あなたがそれをにしていて、それをするながあるにり、することができます。

ここでは、リントをするなユースケースです

- あなたはシステム·アプリをしていますデバイスのでされています
- あなたのアプリは、デバイスのまたはそののされたアクションをするがあります

に、あなたのマニフェストでこれをうことができますつまり、されたアクセスをし、あなたのにがえられることをっているので、

```
<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"/>
```

「されない」エラーのないリソースのインポート

Android API 23をすると、ににそのようながえます

```
context.getResources().getColor(R.color.colorPrimaryDark);
init();
```

'getColor(int)' is deprecated more... (Ctrl+F1)

このようなは、Android APIのリソースのにするなによってします。の

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

すべきです。しかし、android.support.v4ライブラリにはのがあります。

build.gradleファイルにのをします。

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

サポートライブラリのすべてのメソッドがです

```
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));
```

さらにサポートライブラリからのよりくのメソッドをすることができます

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

graditeでLintOptionsをする

lintをするには、 `lintOptions`セクションを`build.gradle`ファイルにします。

```
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
    }
}
```

lintをのバリアントを、たとえば`./gradlew lintRelease`、またはすべてのバリアント `./gradlew lint` にしてすることができます。この、のがどののバリアントにされるかをすレポートがされます。

なすべてのオプションについては、 DSLリファレンスをしてください。

lint.xmlファイルをする

`lint.xml`ファイルでLintののをすることができます。このファイルをでするは、Androidプロジェクトのルートディレクトリにします。 Android StudioでLintをする、lint.xmlファイルがにされ、Androidプロジェクトににされます。

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

タグのをすることにより、のLintチェックをにしたり、のをすることができます。

のは、 `lint.xml`ファイルのをしています。

```
<?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>
```

JavaおよびXMLソースファイルのlintチェックの

JavaおよびXMLソースファイルからLintチェックをにすることができます。

Javaでのリントチェックの

AndroidプロジェクトのJavaクラスまたはメソッドのLintチェックをにするには、そのJavaコードに`@SuppressLint` アノテーションをします。

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

すべてのLintのチェックをにするには

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

XMLでのlintのチェックの

`tools:ignore`をすると、 **XML**ファイルののセクションについてLintチェックをに`tools:ignore`ことができます。

えば

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

XMLのすべてのLintののチェックをするには、のようにします。

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

をする

コードにいくつかのをマークするのがよいです。たとえば、いくつかのされなくなったメソッドは、テストやバージョンのサポートがです。しかし、リントチェックは、そのコードをでマークします。このをするには、アノテーション@SuppressWarningsをするがあります。

たとえば、のメソッドにをすることをします。アノテーションにのをれるもあります

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

このをすると、Lint、Androidなどのすべてのをできます。 Suppress Warningsをすると、コードをしくするのにちます

オンラインでくずのをむ https://riptutorial.com/ja/android/topic/129/くずの

261:

Examples

カラーマニピュレーション

をするために、のargbAlpha、Red、Green、Blueをします。

にあなたのからRGBをします。

```
int yourColor = Color.parse("#ae1f67");

int red = Color.red(yourColor);
int green = Color.green(yourColor);
int blue = Color.blue(yourColor);
```

これで、、、のをらしたりやしたりして、それらをみわせてびにすることができます

```
 int newColor = Color.rgb(red, green, blue);
```

または、アルファをいくつかしたいは、カラーのにすることができます

```
 int newColor = Color.argb(alpha, red, green, blue);
```

アルファとRGBは[0-225]のにあるがあります。

オンラインでをむ https://riptutorial.com/ja/android/topic/4986/

262: されたビュ—の

ViewインスタンスにけられたViewTreeObserverインスタンスは、そのViewがまだきているはになるがあることにしてください。 View.getViewTreeObserver javadocsから

```
// 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()}.
```

したがって、にViewTreeObserverインスタンスにリスナ—をしてするは、するViewインスタンスのgetViewTreeObserverをびして、しいViewTreeObserverインスタンスをけるのがもです。 のインスタンスでisAliveをチェックするのはしでもViewTreeObserverです;もしViewTreeObserverがもはやきていなければ、とにかくそのしいリファレンスをりすでしょう

Examples

アクティビティのビュ—の

```
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);
    }

}
```

オンラインでされたビューのをむ https://riptutorial.com/ja/android/topic/115/されたビューの

263: いし

fastlaneは、iOS、Mac、Androidのがスクリーンショットの、プロビジョニングプロファイルの、アプリケーションのなどのなをするためのツールです。

ドキュメント https : //docs.fastlane.tools/

ソースコード https : //github.com/fastlane/fastlane

Examples

CrashlyticsによってBetaにのをしてアップロードするFastfile

これは、マルチフレーバーアプリのサンプル**Fastfile**です。これは、すべてのフレーバーまたはのフレーバーをし、するオプションをします。デプロイ、 **Slack**にデプロイメントのステータスをし、BetaのテスターにCrashlyticsテスターグループによるをします。

すべてのをしてするには、をします。

```
fastlane android beta
```

のAPKをしてするには

```
fastlane android beta app:flavorName
```

1つのFastlaneファイルをして、iOS、Android、およびMacアプリをできます。このファイルを1つのアプリケーション`platform`にしているは、ではありません。

い

1. `android`は、fastlaneに`:android`プラットフォームをするようします。
2. `:android`プラットフォームでは、のレーンをつことができます。、は`:beta`レーンしかっていません。のコマンドの2は、したいレーンをします。
3. `options[:app]`
4. **Gradle**タスクは2つあります。まず、 `gradle clean`します。あなたがをしたは`app`キー、fastfileる`gradle assembleReleaseFlavor`。 それのは、 `gradle assembleRelease`をしてすべてのビルドフレーバをビルドします。
5. すべてのフレーバーにするは、されたAPKファイルのが`SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS`されます。これをして、されたファイルをループし、CrashlyticsによってBetaにします。 `notifications`および`groups`フィールドはオプションです。らは**Crashlytics**によってベータのアプリにされているテスターにするためにされます。
6. クラッシュスティックスにしているは、ポータルでアプリケーションをアクティブにするには、デバイスでアプリをしてからするがあります。そうしないと、Crashlyticsはアプリをア

クティブにしてエラーをスローします。このシナリオでは、はそれをキャプチャし、として**Slack**にするので、どのアプリがアクティブであるかをることができます。

7. がすると、 **fastlane**は**Slack**にメッセージをります。
8. `#{/([^\/]*)$/.match(apk)}`このは、APKパスからフレーバーをするためにされます。それがあなたのためにかないは、それをすることができます。
9. `get_version_name`と`get_version_code`は、アプリケーションのバージョンとコードをするための2つの**Fastlane**プラグインです。これらのは、いたいにインストールするか、することができます。プラグインのはこちらをごください。
10. のAPKをしてする、 `else`ステートメントがされます。たちは1つのアプリケーションしかないので、Crashlyticsに`apk_path`をするはありません。
11. にかのものがうまくいかないにをけるために、の`error do`ブロックがされます。

`SLACK_URL` 、 `API_TOKEN` 、 `GROUP_NAME` 、 `BUILD_SECRET`をのできえることをれないでください。

```
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
```

されたビルドタイプのすべてのフレーバーをビルドしてデバイスにインストールする**Fastfile**レーン

Fastlineにこのレーンをし、コマンドラインで`fastlane installAll type:{BUILD_TYPE}`をしてください。 `BUILD_TYPE`を、ビルドするビルドタイプにきえます。

にをします`fastlane installAll type:Debug`

このコマンドは、されたタイプのすべてのフレーバーをし、あなたのデバイスにインストールします。、のデバイスがされているはしません。あなたは1つだけをっていることをしてください。はターゲットデバイスをするオプションをするつもりです。

```
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
```

オンラインでいしをむ https://riptutorial.com/ja/android/topic/8215/いし

264:

Examples

なの

このでは、ユーザーがアプリケーションをクリックしたときにアプリケーションをするなをするをします。

のをする

```
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
```

クリックにするインテントをする

```
Intent intent = new Intent(this, MainActivity.class);
PendingIntent pi = PendingIntent.getActivity(this, 0, intent, Intent.FLAG_ACTIVITY_NEW_TASK);
mBuilder.setContentIntent(pi);
```

に、をしてします

```
NotificationManager mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(0, mBuilder.build());
```

いデバイスのティッカーによる

なデバイスのヘッズアップをし、いデバイスにティッカーをするはのとおりです。

```
// 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);

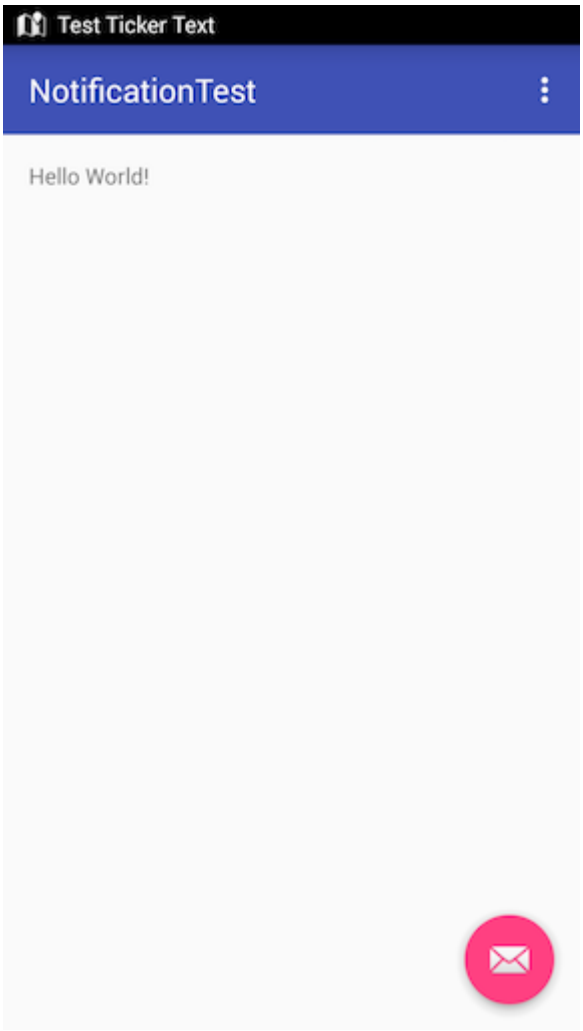
// It won't show "Heads Up" unless it plays a sound
if (Build.VERSION.SDK_INT >= 21) builder.setVibrate(new long[0]);

NotificationManager mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(999, builder.build());
```

ここで、Heads Up Notificationで**Android Marshmallow**でどのようにえるのですか

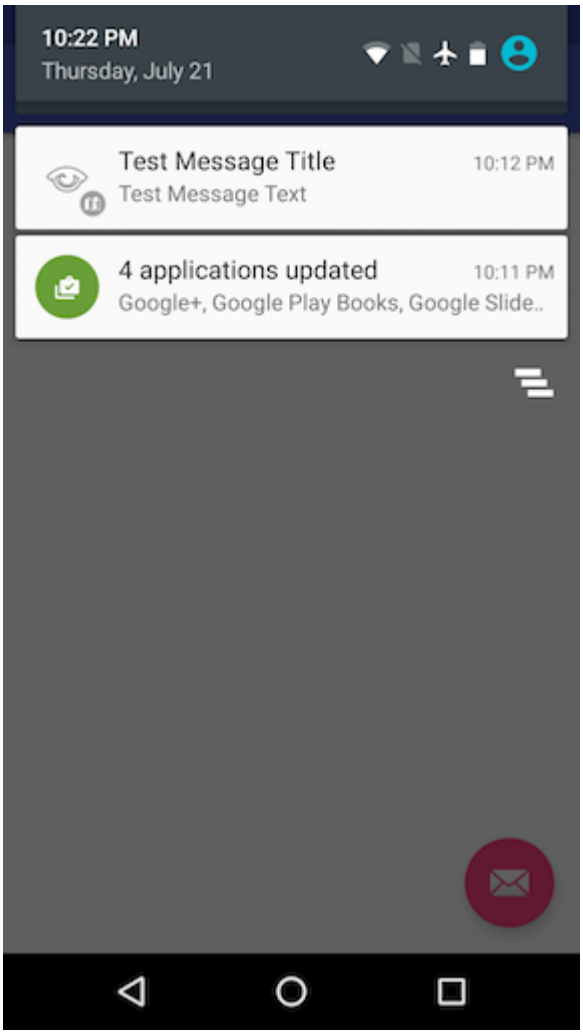


TicketerのAndroid KitKatでのはのとおりです。

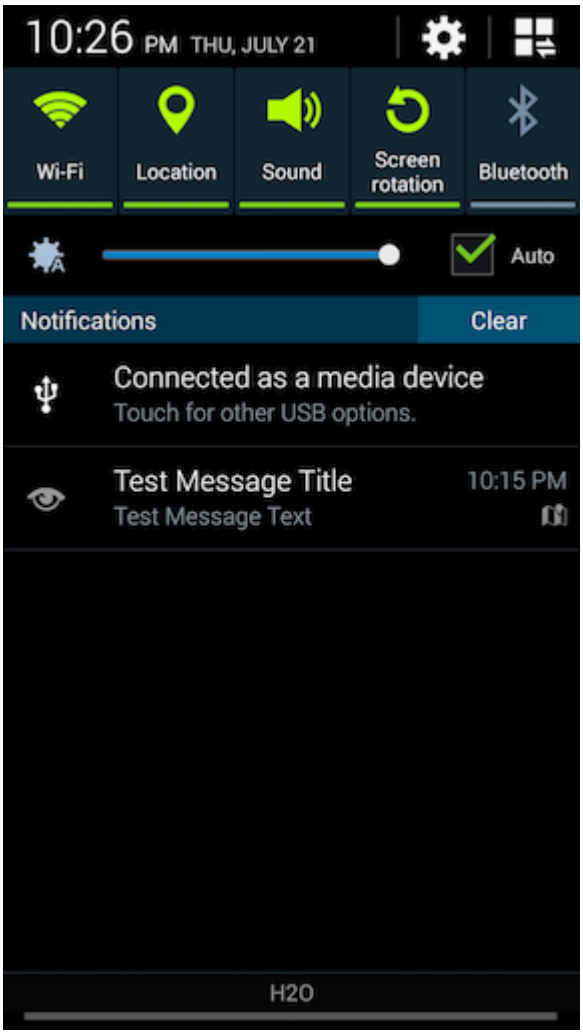


Androidのすべてのバージョンでは、 `Notification`がトレイにされます。

Android 6.0 Marshmallow



Android 4.4.x KitKat



のをする

```
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);
```

にがまれていて、のに「PRIORITY_MAX」をしてをするは、ごとにのをできます

なるレベル

PRIORITY_MAX - のタスクをするに、タイムクリティカルまたはするがあるをユーザーにするおよびのにします。

PRIORITY_HIGH - ユーザーにとってにいのメッセージやチャットイベントなど、なコミュニケーションににします。のいがヘッドアップをトリガします。

PRIORITY_DEFAULT - ここにされているのにしないすべてのにします。

PRIORITY_LOW - ユーザーにするにしますが、はそれほどくありません。のいはリストのにされるがあり、またはソーシャルアップデートのようなものにしています。ユーザーはをめられましたが、これらのはまたはするべきではありません。のコミュニケーション。

PRIORITY_MIN - やのなどのコンテキストやにします。のはステータスバーにされません。ユーザは、シェードのにそれらをする。

ガイドライン -

スケジュール

には、のにをするがあります。これは、Androidシステムでなことになことではないタスクです。メソッドの`setTime()`やとです。このでは、 `AlarmManager`をしてをスケジュールするためになのをします。

1. Android `AlarmManager`によってブロードキャストされた`Intent`をリッスンする**`BroadcastReceiver`**をします。

これは、 `Intent`するエクストラについてをするです。

```
   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. `AndroidManifest.xml`ファイルに**`BroadcastReceiver`**をしますそうしないと、は`AlarmManager`から`Intent`しません。

   ```
   <receiver
       android:name=".NotificationReceiver"
       android:enabled="true" />
   ```

3. あなたの`BroadcastReceiver` `PendingIntent`をな`Intent`エクストラをシステム`AlarmManager`して、をスケジュールします。したになると、 `BroadcastReceiver`は`Intent`をし、をします。のメソッドは、をスケジュールします。

   ```
   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);
}
```

の*42*は、スケジュールされたごとにであるがあります。そうしないと、 *PendingIntent*がいにきえられ、ましくないがします。

4. けられた`PendingIntent`をし、システム`AlarmManager`キャンセルすることで、 をりします。のメソッドは、をキャンセルします。

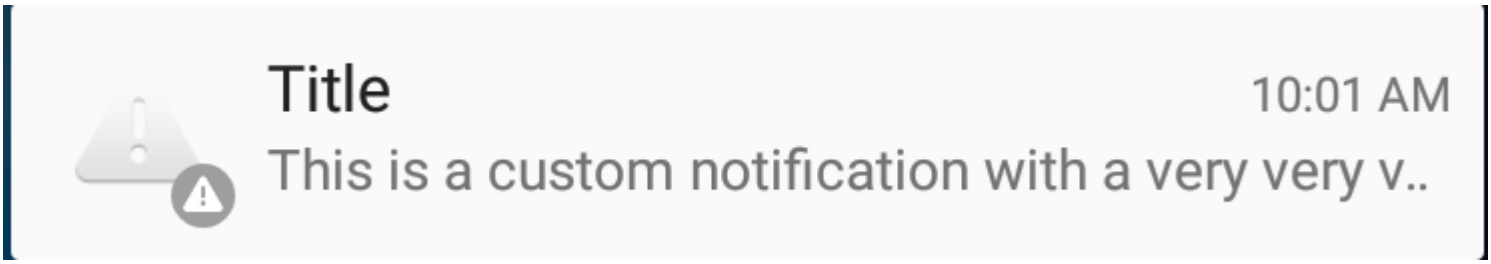
```
public static void cancelNotification(Context context, String title, String text) {
    Intent intent = new Intent(context, NotificationReceiver.class);
    intent.putExtra("title", title);
    intent.putExtra("text", text);
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
    // Cancel notification
    AlarmManager manager = (AlarmManager)
context.getSystemService(Context.ALARM_SERVICE);
    manager.cancel(pending);
}
```

の*42*は、ステップ3のとするがあることにしてください。

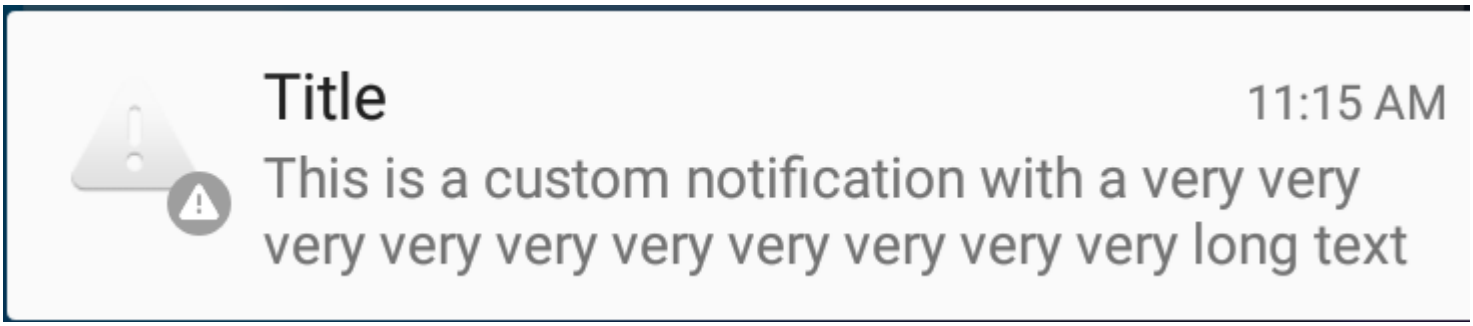
カスタムをする - なのテキストをする

コンテキストにいテキストをするには、カスタムコンテンツをするがあります。

たとえば、あなたはこれをっています



しかし、あなたのテキストがにされることをみます



あなたがするがあるのは、のようなコンテンツにスタイルをすることだけです。

```
  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);
    }
```

`Picasso`ライブラリをってカスタムアイコンをします。

```
    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);
```

レイアウトフォルダにレイアウトをします。

```
<?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>
```

きなアイコンのピクセルサイズをにする

をしたり、をデコードしたり、きなにわせてのサイズをするは、のようなしいピクセルサイズをできます。

```
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);
```

アクションボタンによるの

```
// 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);
```

アクションボタンのクリックイベントをするためにじアクションのブロードキャストレシーバをします。

オンラインでをむ https://riptutorial.com/ja/android/topic/1347/

265: チャンネルAndroid O

き

チャンネルをすることで、Googleのをグループチャンネルにグループすることができます。ユーザーはチャンネルのをにすることができます。Android Oでは、このがされました。

1. class NotificationUtils {} //チャネル
2. createChannel//をするなメソッド

パラメーター

IMPORTANCE_MAX	
IMPORTANCE_HIGH	どこにでもれ、ときをする
IMPORTANCE_DEFAULT	どこにでもせることができますが、はますが、にしません
IMPORTANCE_LOW	どこにでもされますが、ではありません
IMPORTANCE_MIN	シェード、りたたみのにのみされます
IMPORTANCE_NONE	のない。にらない

Examples

チャネル

チャネルとはですか

チャンネルをすることで、アプリはをグループにグループすることができます。ユーザーはチャネルのをにできます。たとえば、チャネルごとに、すべてのをにブロックしたり、をきしたり、バッジをしたりすることができます。このしいは、アプリのユーザーエクスペリエンスをにさせるのにちます。

チャネルをする

```
 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;
}}
```

のコードでは、NotificationChannelの2つのインスタンスをし、チャネルにuniqueidをし、コンストラクタにレベルをしました。チャネルごとに、のをしました。

1.
2. ライト
3.
4. ロックにする。

に、システムからNotificationManagerをし、createNotificationChannelメソッドをびしてチャネルをし、したチャネルをします。

createNotificationChannelsでにのチャネルをし、NotificationChannelインスタンスのJavaリストをすことができます。 getNotificationChannelsでアプリケーションのすべてのチャネルをし、getNotificationChannelでのチャネルをし、チャネルIDだけをとしてすことができます。

チャネルの

IMPORTANCE_MAX	
IMPORTANCE_HIGH	どこにでもれ、ときをする
IMPORTANCE_DEFAULT	どこにでもせることができますが、はますが、にしません
IMPORTANCE_LOW	どこにでもされますが、ではなく、は0です
IMPORTANCE_MIN	シェード、りたたみのにのみされます
IMPORTANCE_NONE	のない。にらない

のとチャンネルへの

NotificationBuilderをしてのNotificationUtilsをして2つのをしました。

```
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);
}
```

また、Notification.BuilderをしてNotificationChannelをすることもできます。そのために ***setChannelString channelId***をできます。

チャネルのをする

チャネルをすると、ユーザーはとをします。 createNotificationChannelをびして、のチャネルのをしたり、そのをしたりすることができます。のサンプルコードでは、アクティビティをするためのインテントをして、チャネルのにユーザーをリダイレクトするについてします。この、イン

テントにはチャネルのIDとアプリのパッケージなどのデータがです。

```
@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ファイル

```
<!--...-->
<Button
    android:id="@+id/btn_android_notif_settings"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Notification Settings"/>
<!--...-->
```

チャネルの

チャネルをするには、deleteNotificationChannelをびします。

```
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);
```

MainActivityと**xml**をする

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

MainActivityをしてEditTextコンポーネントからタイトルとをし、Androidチャンネルにることができます。 NotificationUtilsでしたAndroidチャンネルのNotification.Builderをし、NotificationManagerにします。

import android.app.Notification; import android.os.Bundle; import android.support.v7.app.AppCompatActivity; android.text.TextUtilsをインポートします。 import 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());
                }
            }
        });
    }
}
```

オンラインでチャンネルAndroid Oをむ https://riptutorial.com/ja/android/topic/10018/チャンネル-android-o

266: メールの

Examples

メールアドレスの

メールアドレスがかどうかをするには、のメソッドをします。

```
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();
}
```

のメソッドは、 `EditText`ウィジェットのテキストを`String`することでにできます

```
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();
}
```

パターンをしたメールアドレス

```
  if (Patterns.EMAIL_ADDRESS.matcher(email).matches()){
       Log.i("EmailCheck","It is valid");
  }
```

オンラインでメールのをむ https://riptutorial.com/ja/android/topic/5605/メールの

267: をパターンでフォーマットする。

き

このでは、のをするをします

グラデーションにはのライブラリがです。

コンパイル 'com.googlecode.libphonenumberlibphonenumber7.2.2'

Examples

パターン+17861234 5678

+ 178612345678のようなされたをすると、されたパターンでフォーマットがされます。

```
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;
}
```

オンラインでをパターンでフォーマットする。をむ https://riptutorial.com/ja/android/topic/9083/をパターンでフォーマットする-

268: からテキストへの

Examples

のGoogleプロンプトダイアログをしたテキストへの

スピーチをテキストにトリガー

```
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();
    }
}
```

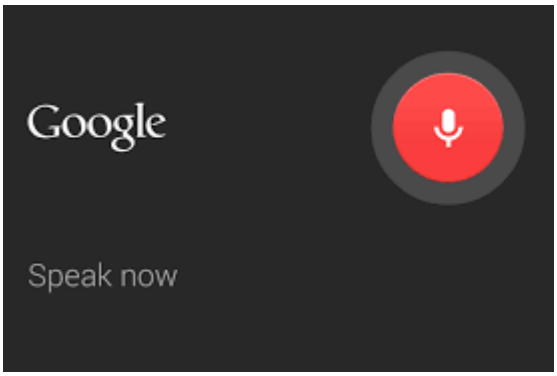
onActivityResultでをする

```
@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;
    }

    }
}
```



ダイアログなしのテキストへのスピーチ

のコードをして、ダイアログをせずにからテキストへのをすることができます。

```
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);
}
```

のコードでされているカスタムリスナークラス`CustomRecognitionListener`は、のようにされています。

```
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);
    }
}
```

オンラインでからテキストへのをむ https://riptutorial.com/ja/android/topic/6252/からテキストへの

クレジット

S. No		Contributors
1	Androidをいめる	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, Charuಞ, 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パッチ	Knossos, Nissim R, Tomik
3	ACRA	Zarul Izham, Zoe
4	ADBAndroid Debug Bridge	3VYZkz7t, adao7000, Ahmad Aghazadeh, Amit Thakkar, AndroidMechanic, Anirudh Sharma, Anup Kulkarni, auval, Barend, Blackbelt, Burak Day, Charuಞ, 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シエル	3VYZkz7t, Ahmad Aghazadeh, auval, Burak Day, Fabio, fyfyone Google, Hannoun Yassir, Natali, Pavel Durov, R. Zagórski, sukumar, Yury Fedorov

6	ADBをしたアプリのインストール	Ahmad Aghazadeh, fyfyone Google, Laurel, Xaver Kapeller
7	AdMob	Carlos Borau, honk, RamenChef, Sukrit Kumar, Zarul Izham, Zoe
8	AIDL	Krishnakanth
9	AlarmManager	Daniel Nugent, devnull69, Greg T, honk, TR4Android
10	Android Javaネイティブインタフェースjni	Doron Yakovlev-Golani, Muthukrishnan Rajendran, samgak
11	Android NDK	Alex, astuter, Doron Yakovlev-Golani, Flayn, Onik, samgak , still_learning, Täg, thiagolr
12	Android Paypalゲートウェイの	A-Droid Tech
13	Android Places API	busradeniz, honk, Karan Razdan, Murali
14	Android Studioでインスタント	AndroidMechanic, Daniel Nugent, ridsatrio, Zoe
15	Android StudioにOpenCVをする	MashukKhan, RamenChef, ssimm
16	Android Vk Sdk	alexey polusov
17	Android-x86 in VirtualBox	Daniel Nugent, Enrique de Miguel
18	Androidアーキテクチャのコンポーネント	DeKaNszn
19	Androidアプリケーションののライブラリをする	EpicPandaForce, honk, mnoronha
20	Androidオーセンティケータ	4444, kRiZ
21	Androidカーネルの	honk, Sneh Pandya
22	AndroidカスタムROMの	honk, Pradumn Kumar Mahanta
23	Androidサウンドとメディア	johnrao07, Muhammad Umair Shafique, Squidward
24	Androidスタジオ	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
25	AndroidでClingライブラリをしたポートマッピング	Shinil M S
26	Androidでのシェイクイベントの	N-JOY, tynn, Xiaozou
27	Androidでのリソースによるローカライゼーション	AndroidMechanic, electroid, Fabio, Gubbel, Harish Gyanani, honk, Jinesh Francis, mpkuth, RamenChef, USKMobility
28	Androidでファイルをする	Arth Tilva, Daniel Nugent, mnoronha
29	AndroidのJava	Eugen Martynov
30	Androidのゲーム	Zoe
31	Androidのバージョン	4444, AndroidMechanic, athor, BooleanCheese, Dalija Prasnikar, Daniel Nugent, Fildor, Gabriele Mariotti, H. Pauwelyn, Matt, RediOne1, tynn
32	Androidのもの	Fabio, honk
33	Androidのローカライズされた/	Geert, honk, mnoronha
34	AndroidプロジェクトへのFuseViewの	Tudor Luca
35	AndroidプロジェクトのJenkins CIセットアップ	honk, Ichthyocentaurs
36	AndroidのFileIO	h22, sun-solar-arrow
37	AndroidFacebook SDK	Akeshwar Jha, AndiGeeky, Community, Daniel Nugent, honk, Zarul Izham
38	AndroidGoogle Maps API v2	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
39	AndroidGradle	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
40	API-23 +におけるランタイムパーミッション	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
41	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
42	AudioManager	honk, Nicolai Weitkemper
43	AutoCompleteTextView	Harish Gyanani, Jon Adams, Ricardo Vieira, Vivek Mishra
44	BluetoothとBluetooth LE API	antonio, Jon Adams, Lukas, Myon, Pavel Durov, R. Zagórski, Reaz Murshed, V-PTR, WMios
45	Bluetoothエネルギー	Roberto Betancourt
46	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
47	CardView	Carlos, Dan, Er. Kaushik Kajavadara, Gabriele Mariotti,

		Kaushik, Nougat Lover, RamenChef, S.R, Sneh Pandya, Somesh Kumar, Stephen Leppik, sud007, WarrenFaith, Yury Fedorov
48	CleverTap	Jordan, judepereira
49	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
50	ConstraintSet	Pratik Butani
51	ContentProvider	Andrew Siplas, cdeange, Daniel Nugent, Dinesh Choudhary, Lewis McGeary, RamenChef
52	ContentValuesクラスをしたSQLiteデータベースへのアクセス	Adil Saiyad, emecas, honk
53	DayNightテーマ AppCompat v23.2 / API 14+	Ishita Sinha
54	Drawables	alanv, B001, Daniel Nugent, Greg T, Hiren Patel, Jinesh Francis, Nick Cardoso, TR4Android
55	EditText	Daniel Nugent, Gabriele Mariotti, Kaushik NP, Muthukrishnan Rajendran, Rubin Nellikunnathu, Yousha Aleayoub
56	ExoPlayer	Hamed Gh
57	FileProvider	Joost Verbraeken, pedros
58	Firebase App Indexing	shalini, tynn
59	Firebase Cloud Messaging	Gabriele Mariotti, shikhar bansal, Shubham Shukla, Zarul Izham
60	Firebaseクラッシュレポート	AndiGeeky, Gabriele Mariotti, honk, RamenChef, Stephen Leppik, Zarul Izham
61	Firebaseリアルタイムデータベース	Aawaz Gyawali, drulabs, Gabriele Mariotti, honk, Md. Ali Hossain, RamenChef, Sneh Pandya, Stephen Leppik, yennsarah, Zarul Izham
62	FloatingActionButton	Ahmad Aghazadeh, Charuක, Daniel Nugent, Gabriele Mariotti, mattfred, RamenChef, Shinil M S, Stephen Leppik

63	Google Awareness APIs	Dus, honk, Willie Chalmers III
64	Googleドライブ API	Christlin Joseph, honk
65	Googleログインの	AndiGeeky, RamenChef, Tot Zam
66	Googleの	Egek92, RamenChef, ReverseCold, Stephen Leppik, Zarul Izham
67	GradleでApache Archivaに.aarファイルをする	Marian Klühspies
68	GreenDAO	Allan Pereira, Carl Poole, Grundy, MiguelHincapieC, R. Zagórski, RamenChef, Stephen Leppik
69	GreenRobot EventBus	CaseyB, Daniel Nugent, Hamed Momeni, RamenChef
70	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
71	HttpURLConnection	Aleks G, Daniel Nugent, Duan Bressan, honk, KDeogharkar, marshmallow, Shantanu Paul, Simone Carletti
72	ImageView	Ahmad Aghazadeh, Ali Sherafat, Chip, Daniel Nugent, DanielDiSu, Dinesh, Gabriele Mariotti, Harish Gyanani, kit, lax1089, Pratik Butani, Squidward, Sup
73	IntentService	Anax, Daniel Nugent, honk, JonasCz, TRINADH KOYA, Yashaswi Maharshi
74	JCodec	Adhikari Bishwash
75	JSON with Android with 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
76	JUnitによるAndroidのユ	abhi, Andre Perkins, AndroidMechanic, Eugen Martynov,

	ニットテスト	honk, Lewis McGeary, N J, Namnodorel, Patrick Dattilio, Rolf ツ
77	KotlinによるAndroidプログラミング	Gian Patrick Quintana, Govinda Paliwal, Oknesif, Zarul Izham
78	LruCache	Daniel Nugent, honk, LordSidious, RamenChef, Stephen Leppik
79	MediaPlayer	Ahmad Aghazadeh, Carlos Vázquez Losada, hello_world, Makille, R. Zagórski, Redman
80	MediaSession	Disk Crasher, honk, KuroObi, RamenChef
81	MediaStore	Daniel Nugent, honk, RamenChef, Uttam Panchasara
82	MultidexとDexメソッドの	Adarsh Ashok, Ben, bigbaldy, cdeange, Daniel Nugent, Gabriele Mariotti, Mike, Pongpat, R. Zagórski, Shirane85
83	MVPアーキテクチャ	Atif Farrukh, Harish Gyanani, honk, Jon Adams, Magesh Pandian, N J, zmingchun
84	MVVMアーキテクチャ	Daniel W., RamenChef, Stephen Leppik
85	NavigationView	Adam Lear, akshay, Charuᔕ, Daniel Nugent, Gabriele Mariotti, Kedar Tendolkar, petrumo, RamenChef, rekire, SANAT, Sevle, Stephen Leppik, sud007
86	OkHttp	A-Droid Tech, Daniel Nugent, Gabriele Mariotti, Gubbel, noob, Rohit Arya, Vucko, Zarul Izham
87	Okio	Adhikari Bishwash
88	OpenGL ES 2.0をいめる	MarGenDo
89	PackageManager	FredMaggiowski, Hi I'm Frogatto, Muthukrishnan Rajendran, Piyush, Squidward
90	Ping ICMP	Carl Poole
91	Playストアにする	Carlos Borau, Fabio, mnoronha, Zoe
92	PorterDuffモード	Adarsh Ashok, AndroidMechanic, Knossos, PhilLab, S.D., Vasily Kabunov
93	ProGuard - コードのと	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

94	ProGuardとはですかAndroidでのはですか	Ayush Bansal, Daniel Nugent, Ghanshyam Sharma, Pratik Butani
95	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, Charuක, 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
96	RecyclerView onClickListener	abhishesh, Braj Bhushan Singh, Bryan, FromTheSeventhSky, fuwaneko, Gabriele Mariotti, honk, RamenChef, Smit.Satodia, Stephen Leppik
97	RecyclerViewおよびLayoutManager	4444, BalaramNayak, Felix Edelmann, Gabriele Mariotti, Kayvan N, MidasLefko, RamenChef, Stephen Leppik
98	RecyclerViewデコレーション	Barend, David Medenjak, Gabriele Mariotti, Muthukrishnan Rajendran, Peter Gordon, RamenChef, Stephen Leppik, Yasin Kaçmaz
99	RecyclerViewのページネーション	Muhammad Younas
100	RenderScript	Ankit Popli, Dalija Prasnikar, Froyo, honk, Rucha Bhatt, Xaver Kapeller
101	RoboGuice	AndroidRuntimeException, Lewis McGeary, Rajesh
102	RxJavaをったRetrofit2	Anand Singh, gaara87, GurpreetSK95, Lukas, mrtuovinen, R. Zagórski, Zarul Izham
103	SearchView	Daniel Nugent, Dmide, Hiren Patel, RamenChef, Stephen Leppik, sud007
104	SensorManager	honk, Simon, TDG
105	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
106	SpannableString	S.R
107	SparseArrayのい	honk, Robert Banyai
108	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
109	SurfaceViewをしたキャンバス	davidgiga1993
110	TabLayout	Daniel Nugent, Willie Chalmers III
111	TensorFlow	Pratik Butani
112	TextInputLayout	Adarsh Ashok, BrickTop, Gabriele Mariotti, Hi I'm Frogatto, RamenChef, Shashanth, Sneh Pandya, Stephen Leppik
113	TextView	Beena, Daniel Nugent, Eyad Mhanna, gaara87, Gabriele Mariotti, Hiren Patel, honk, keno, Michele, Sohail Zahid, Sujith Niraikulathan, sun-solar-arrow
114	TransitionDrawable	S.R, Yogesh Umesh Vaity
115	Twitter API	Mahmoud Ibrahim
116	Typedef@IntDef、@StringDef	Gabriele Mariotti, hardik m, mmBs, Pongpat
117	UIAutomatorによるアプリUIテスト	Timo Bähr
118	UIテストの - Android	Atif Farrukh, Daniel Nugent, Gabriele Mariotti, honk, Jon Adams, originx

119	UIライフサイクル	Daniel Nugent, Dinesh Choudhary, Floern, Lewis McGeary, orelzion, R. Zagórski, Sergey Glotov
120	VectorDrawableおよびAnimatedVectorDrawable	Ahmad Aghazadeh, Aleksandar Stefanović, gaara87, honk, Lewis McGeary, RamenChef, Stephen Leppik
121	VideoView	iravul, Sashabrava
122	ViewFlipper	Anita Kunjir, Daniel Nugent, honk
123	ViewPager	Adarsh Ashok, Adrián Pérez, Daniel Nugent, Gabriele Mariotti, Moustachauve, RamenChef, RediOne1, Rucha Bhatt, Sneh Pandya, Stephen Leppik, Usman, ZeroOne
124	WebView	Amod Gokhale, Daniel Nugent, g4s8, j2ko, jasonlam604, JonasCz, Mohammad Yahia, ppeterka, Prakash Bala, shtolik, Squidward, Sukrit Kumar, sukumar
125	Wi-Fi	4444, AndroidMechanic, Daniel Nugent, gus27
126	XMPPログインとチャットのな	4444, RamenChef, Saveen
127	Xposed	Medusalix
128	Youtube-API	abhishesh, Giannis, honk, MashukKhan, Zarul Izham, Zeeshan Shabbir
129	アイコンフォントをしてAndroidのパフォーマンスをさせる	Beto Caldas, honk, Neeraj
130	アカウントとAccountManager	gaara87, systemovich
131	アクティビティ	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
132	アニメーションのAlertDialogボックス	krunal patel, Thomas Easo
133	アニメーター	Aryan, Bartek Lipinski, Blundering Philosopher, Brenden Kromhout, Charuᘐ, Daniel Nugent, Eixx, Hiren Patel, Lewis McGeary, Piyush, TR4Android, Uriel Carrillo, Yury Fedorov

134	アプリ	Hussein El Feky, Pro Mode, Zoe
135	アラートダイアログの	Adil Saiyad, honk
136	アンドロイドのGoogleサインイン	jagapathi
137	アンドロイドのORMLite	Manos
138	アンドロイドのZipファイル	Adnan
139	アンドロイドのためのGenymotion	Atef Hares, Harish Gyanani
140	アンドロイドのAPI	Doron Yakovlev-Golani, RamenChef, user01232
141	アンドロイドプロジェクトでRetrolambdaをするためのい。	anatoli, Md. Ali Hossain
142	インターネットをする	AndiGeeky, Bill, Daniel Nugent, gbansal, Ichigo Kurosaki, Jon Adams, sukumar, TameHog, Yousha Aleayoub
143	インターフェイス	appersiano, Brenden Kromhout, Daniel Nugent, RediOne1
144	インテント	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, Charuක, 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, ɬolɐəz əɥɬ qoq, Tyler Sebastian, vasili111, Vasily Kabunov, Vinay , Vivek Mishra, Xaver Kapeller, younes zeboudj, Yury Fedorov, Zoe
145	ウィジェット	4444, Alex Ershov, Daniel Nugent, Don Chakkappan, Imdad, nenofite, sun-solar-arrow
146	エスプレッソによるUIのテスト	Daniel Nugent, Gabriele Mariotti, Jason Robinson, Michael Vescovo, Milad Faridnia, N J, RamenChef, Víctor Albertos
147	エミュレーター	Ahmad Aghazadeh, Dan Hulme, fyfyone Google, honk, rekire, Rubin Nellikunnathu, ThomasThiebaud
148	オーディオトラック	Ayush Bansal
149	オーバーレイにのWindowsの	honk, mnoronha, NitZRobotKoder, Rupali, Sujith Niraikulathan
150	オットーイベントバス	gus27, tynn
151	オリエンテーションの	EmmanuelMess, k3b, Ricardo Vieira, y.feizi
152	カウントダウンタイマー	privatestaticint
153	カスタムRangeSeekBarのステップインクリメントをする	Romu Dizzy
154	カスタムビューの	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
155	カスタムフォント	Daniel Nugent, Erik Ghonyan, Gabriele Mariotti, Hiren Patel, honk, kit, Nougat Lover, Simon Schubert, Stanojkovic, Sujith Niraikulathan
156	カメラ2 API	ChemicalFlash, devnull69, RamenChef, webo80
157	カメラとギャラリー	Ahmad Aghazadeh, carvaq, Daniel Nugent, Hiren Patel, johnrao07, RediOne1, Squidward, Yasin Kaçmaz
158	キーボード	Hiren Patel, Kayvan N

159	グーグルプレイストア	dakshbhatt21, Daniel Nugent, reVerse
160	クラッシュレポートツール	Ajit Singh, Charuज, Ekin, Gabriele Mariotti, Ishita Sinha, Jason Bourne, Madhukar Hebbar, pRaNaY
161	コーディネーターのレイアウトと	Adarsh Ashok, Gabriele Mariotti, honk, RamenChef, Stephen Leppik
162	コールバックURL	Atif Farrukh, honk, RamenChef, Stephen Leppik
163	コンテキスト	Will Evers
164	サービス	adao7000, AndroidMechanic, Apoorv Parmar, BadCash, B-GangsteR, Daniel Nugent, g4s8, Hiren Patel, JonasCz, Lazai, Lucas Paolillo, Michael Spitsin, Nougat Lover, rakeshdas, Vinícius Barros
165	さまざまな、サイズでをサポート	Eduardo, Guilherme Torres Castro, kalan, mpkuth, Onur, ppeterka
166	ジェスチャー	mpkuth
167	システムフォントのとフォントの	Adil Saiyad, honk
168	ジャクソン	KeLiuyue
169	ショートカットマネージャ	g4s8, Sukrit Kumar
170	ジョブスケジューリング	RamenChef, reflective_mind
171	スクリーンショットのキャプチャ	Ayush Bansal, Daniel Nugent, honk, Onik, sushant kumar, W0rmH0le
172	ストリングのフォーマット	Beena, Daniel Nugent, gaara87, Greg T, Michele, RamenChef, Suresh Kumar
173	スナックバー	AndroidRuntimeException, Charuज, 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
174	スピナー	AndroidMechanic, Anonsage, Daniel Nugent, Vishwesh Jainkuniya

175	スプラッシュの	honk, Kiran Benny Joseph, Zoe
176	スマートカード	shadygoneinsane
177	スワイプしてする	Chirag SolankI, Daniel Nugent, Gabriele Mariotti, Malek Hijazi, RamenChef, Stephen Leppik
178	セキュアな SharedPreferences	Christlin Joseph
179	セキュリティ	xDragonZ
180	ダイアログ	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
181	タイムユース	Burhanuddin Rashid, Mukesh Kumar Swami, Muthukrishnan Rajendran
182	ダガー2	Aurasphere, Cabezas, David Medenjak, EpicPandaForce, honk, mattfred, Tomik
183	タッチイベント	Yvette Colomb
184	タッチイベントとモーションイベントの	honk, Zoe
185	ツール	Dalija Prasnikar, Gabriele Mariotti, Harsh Sharma, Kayvan N, TR4Android
186	データのをにう SyncAdapter	Bhargavi Yamanuri
187	データの/	honk, HoseinIT, Robert
188	データバインディングライブラリ	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
189	データの	sukumar, Suresh Kumar
190	テーマ、スタイル、	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
191	テキストビューのサイズ	honk, Priyank Patel
192	テキストみげTTS	Ahmad Aghazadeh, honk, Jordan, Lukas, nibarius, Peter Taylor, RamenChef, Stephen Leppik
193	デザインパターン	Adhikari Bishwash, honk, Steve.P
194	デバイスメトリック	Daniel Nugent, Hiren Patel, Talha, W3hri
195	トースト	Adam Ratzman, adao7000, Aida Isay, Amit, Andrew Brooke, AndroidMechanic, Avijit Karmakar, Bartek Lipinski, cdeange, Charuᔓ, Daniel Nugent, Erik Ghonyan, Gabriele Mariotti, Lewis McGeary, LordSidious, Lukas, mpkuth, MrSalmon, RamenChef, Rohit Arya, Sammy T, saurav, SoroushA, sukumar, Vicky, Vucko
196	トーストメッセージのシングルトンクラスをする	Emad, Ishan Fernando
197	ドーズモード	Daniel Nugent, Fabio, honk, NitZRobotKoder, RamenChef, Rosário Pereira Fernandes, Rupali
198	バーコードとQRコードのみり	FlyingPumba
199	パーセルブル	Alex Sullivan, Andrei T, HoseinIT, Nick Cardoso
200	ハードウェアボタンイベント/インテントPTT、LWPなど	JensV
201	パスワードをにする	honk, Jaggs
202	バターナイフ	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
203	パフォーマンスの	honk, Jonas Köritz
204	ハンドラ	Daniel Nugent, Floern, Hasif Seyd, Lewis McGeary, Mike Scamell, Muhammed Refaat, Sweeper, Tomik, TR4Android
205	ピカソ	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
206	ビットマップキャッシュ	Lokesh Desai
207	ビットマップをにみむ	iDevRoids
208	ファイアベース	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
209	ファストジョン	KeLiuyue
210	フレスコ	Alexander Oprisnik, Daniel Nugent, honk, Nilesh Singh, Zarul Izham
211	プログレスバー	Gabriele Mariotti, Hiren Patel, mpkuth, Sanoop, shtolik
212	プロジェクトSDKのバージョン	Arnav M., Ranveer, Tanis.7x
213	ペイント	Nicolas Maltais
214	ベクトルファイル	Priyank Patel, ShahiM
215	ベトナムをのにする	1SStorm
216	ボタン	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
217	ボトムシート	Daniel Nugent, Gabriele Mariotti, Magesh Pandian, MiguelHincapieC, RamenChef, Stephen Leppik, sud007, Zarul Izham
218	ボトムナビゲーションビュー	Abdul Wasae, Daniel Nugent, Gabriele Mariotti, guik, Pankaj Kumar, Pratik Butani, Priyank Patel, RamenChef, rciovati, Stephen Leppik, sud007
219	ボレー	2943, Ankur Aggarwal, Endzeit, Harsh Dalwadi, herrmartell, honk, Jon Adams, Pablo Baxter, RamenChef, Rubin Nellikunnathu, Rucha Bhatt, sameera lakshitha, Stephen

		Leppik, VISHWANATH N P
220	マニフェストファイル	John Snow, Jon Adams, kit, mayojava, Menasheh
221	メニュー	Bhargavi Yamanuri, Chip, Daniel Nugent, Hi I'm Frogatto, honk, Iman Hamidi
222	メモリリーク	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
223	モシ	Blundell
224	ユニバーサルローダ	Greg T, honk, Jon Adams, priyankvex, Stephen Leppik
225	ライブラリダガー2アプリケーションの	Er. Kaushik Kajavadara, honk
226	ライブラリをMavenリポジトリにする	Farid
227	リストビュー	A.A., brainless, Daniel Nugent, Diti, Douglas Drumond, Fabian Tamp, Gabriele Mariotti, Hiren Patel, Mr.7, Ruben Pirotte, Saeed-rz, shaonAshraf, Squidward
228	リソース	biddulph.r, Brenden Kromhout, Charuজ, 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
229	リリースのためにAndroidアプリにする	Gabriele Mariotti, M M
230	ルーパー	tynn
231	レイアウト	a.ch., Adarsh Ashok, Adinia, AesSedai101, Ahmad Aghazadeh, Aleksandar Stefanović, ankit dassor, Aurasphere, Bartek Lipinski, Björn Kechel, bjrne, Brenden Kromhout, Charuজ, 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
232	レトロフィット2	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
233	レルム	bdash, Dan, EpicPandaForce, Hi I'm Frogatto, iurysza, null pointer, RamenChef, Stephen Leppik, sukumar
234	ローダ	chandsie, g4s8, jefry jacky, Marcus Becker, RamenChef, Stephen Leppik
235	ログとログキャストの	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
236	ロケーション	Alex Chengalan, Aryan, BadCash, Daniel Nugent, Hiren Patel, Mahmoud Ibrahim, MidasLefko, Pablo Baxter, RamenChef, Stephen Leppik
237	ロボエレクトリック	Blundell, g4s8
238	のあるアプリケーションの	Jon Adams, mnoronha, RamenChef, SoroushA
239		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
240	の	noongiya95
241	およびストレージへのファイルの	Amit Vaghela, Andrew Brooke, AnV, Daniel Nugent, Gabriele Mariotti, Nickan B, Uttam Panchasara
242	モードポリシーコンパイルのバグをするツール。	Shekhar
243	アダプタによるデータ	Arpit Gandhi, mnoronha
244	とローカリゼーション	Ankur Aggarwal

	I18NとL10N	
245		cdeange, Zerntrino
246		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
247	とのピッカー	adalPaRi, Brenden Kromhout, Daniel Nugent, Harish Gyanani, Ironman, Milad Nouri, RediOne1, Rohan Arora
248	のインテント	Blundering Philosopher, Daniel Nugent, mnoronha, Pratik Butani, SoroushA
249	されたVideoView	Chip
250		Akash Patel, Aleksandar Stefanović, Alex Chengalan, AndroidMechanic, Anirudh Sharma, ankit dassor, Bartek Lipinski, Bulwinkel, cascal, Charuᔕ, 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
251	プロセッサ	krishan
252	の	Pablo Baxter
253	いリンクの	Doron Yakovlev-Golani, Harsh Sharma, mnoronha, Tanis.7x
254		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
255	れ	Rakshit Nawani, tynn
256		Hiren Patel, mnoronha
257	/の	Vishal Puri
258		Daniel Nugent, PRIYA PARASHAR, RamenChef
259	くずの	ben75, Daniel Nugent, Gabriele Mariotti, GensaGames, R. Zagórski, rekire, SuperBiasedMan
260		Carlos Borau, Dalija Prasnikar, Daniel Nugent, Erfan Mowlaei, Jon Adams, N J, Sujith Niraikulathan
261	されたビューの	mnoronha, stkent
262	いし	Gokhan Arik
263		alexey polusov, bricklore, Da-Jin C, Daniel Nugent, Dus, gbansal, Jeeter, piotrek1543, RediOne1, Rupali, TR4Android, weston
264	チャンネルAndroid O	Lokesh Desai
265	メールの	Hiren Patel, honk, iravul, Nicolas Maltais
266	をパターンでフォーマットする。	Pedro Varela
267	からテキストへの	Hitesh Sahu, honk, RamenChef, Stephen Leppik