Kostenloses eBook

LERNEN Android

Free unaffiliated eBook created from
Stack Overflow contributors.

#android

Inhaltsverzeichnis

Über

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

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

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

Kapitel 1: Erste Schritte mit Android

Bemerkungen

Wenn Sie mehr über die Android Gradle Plugin-Einstellung erfahren möchten, `android-gradle` Sie die Android-Gradle `android-gradle` Dokumentation .

Wenn Sie sich für alternative Emulatoren interessieren, können Sie sich Genymotion anschauen . Es bietet einen kostenlosen Plan und benötigt weniger RAM.

Versionen

Ausführung	API-Ebene	Versionscode	Veröffentlichungsdatum
1,0	1	`BASE`	2008-09-23
1.1	2	`BASE_1_1`	2009-02-09
1,5	3	`CUPCAKE`	2009-04-27
1.6	4	`DONUT`	2009-09-15
2,0	5	`ECLAIR`	2009-10-26
2.0.1	6	`ECLAIR_0_1`	2009-12-03
2.1.x	7	`ECLAIR_MR1`	2010-01-12
2.2.x	8	`FROYO`	2010-05-20
2.3	9	`GINGERBREAD`	2010-12-06
2.3.3	10	`GINGERBREAD_MR1`	2011-02-09
3.0.x	11	`HONEYCOMB`	2011-02-22
3.1.x	12	`HONEYCOMB_MR1`	2011-05-10
3.2.x	13	`HONEYCOMB_MR2`	2011-07-15
4,0	14	`ICE_CREAM_SANDWICH`	2011-10-18
4.0.3	fünfzehn	`ICE_CREAM_SANDWICH_MR1`	2011-12-16
4.1	16	`JELLY_BEAN`	2012-07-09
4.2	17	`JELLY_BEAN_MR1`	2012-11-13

Ausführung	API-Ebene	Versionscode	Veröffentlichungsdatum
4.3	18	`JELLY_BEAN_MR2`	2013-07-24
4.4	19	`KITKAT`	2013-10-31
4,4W	20	`KITKAT_WATCH`	2014-06-25
5,0	21	`LOLLIPOP`	2014-11-12
5.1	22	`LOLLIPOP_MR1`	2015-03-09
6,0	23	`M` (Eibisch)	2015-10-05
7,0	24	`N` (Nougat)	2016-08-22
7.1	25	`N_MR1` (Nougat MR1)	2016-10-04
8,0	26	`O` (Entwicklervorschau 4)	24.07.2017

Examples

Android Studio einrichten

Android Studio ist die Android-Entwicklungs-IDE, die von Google offiziell unterstützt und empfohlen wird. Android Studio wird mit dem Android SDK Manager geliefert , einem Tool zum Herunterladen der `Android SDK` Komponenten, die zum Starten der Entwicklung von Apps erforderlich sind.

Installieren von Android Studio- und `Android SDK` Tools:

1. Laden Sie Android Studio herunter und installieren Sie es.
2. Laden Sie die neuesten SDK Tools und SDK Platform-Tools herunter, indem Sie das Android Studio öffnen und dann den Anweisungen zum Android SDK Tool Updates folgen. Sie sollten die neuesten verfügbaren stabilen Pakete installieren.

   Wenn Sie an alten Projekten arbeiten müssen, die mit älteren SDK-Versionen erstellt wurden, müssen Sie möglicherweise auch diese Versionen herunterladen

Seit Android Studio 2.2 ist eine Kopie des neuesten OpenJDK im Lieferumfang der Installation enthalten. Es ist das empfohlene JDK (Java Development Kit) für alle Android Studio-Projekte. Dadurch entfällt die Notwendigkeit, dass das JDK-Paket von Oracle installiert ist. Um das mitgelieferte SDK zu verwenden, gehen Sie wie folgt vor:

1. Öffnen Sie Ihr Projekt in Android Studio und wählen Sie in der Menüleiste **Datei> Projektstruktur aus** .
2. **Aktivieren Sie auf der** Seite **SDK Location** und unter **JDK Location das** Kontrollkästchen **Embedded JDK verwenden** .
3. Klicken Sie auf **OK** .

Konfigurieren Sie Android Studio

Android Studio bietet über das **Hilfe-** Menü Zugriff auf zwei Konfigurationsdateien:

- studio.vmoptions : Passen Sie die Optionen für die Java Virtual Machine (JVM) von Studio an, z. B. Heapgröße und Cache-Größe. Beachten Sie, dass diese Datei auf Linux-Computern je nach Ihrer Version von Android Studio möglicherweise als *studio64.vmoptions bezeichnet* wird.
- idea.properties : Anpassen der Android Studio-Eigenschaften, z. B. des Ordnerpfad des Plugins oder der maximal unterstützten Dateigröße.

Thema ändern / hinzufügen

Sie können es nach Belieben ändern. `File->Settings->Editor->Colors & Fonts->` und wählen Sie ein Thema aus. Sie können auch neue Themen von http://color-themes.com/ herunterladen. Wenn Sie die `.jar.zip` Datei heruntergeladen haben, gehen Sie zu `File -> Import Settings...` und wählen Sie die heruntergeladene Datei.

Apps kompilieren

Erstellen Sie ein neues Projekt oder öffnen Sie ein vorhandenes Projekt in Android Studio und drücken Sie die grüne Wiedergabetaste  auf der oberen Symbolleiste, um es auszuführen. Wenn es grau ist, müssen Sie eine Sekunde warten, damit Android Studio einige Dateien ordnungsgemäß indizieren kann, deren Fortschritt in der unteren Statusleiste angezeigt wird.

Wenn Sie ein Projekt von der Shell aus erstellen möchten, stellen Sie sicher, dass Sie über die Datei `local.properties` verfügen, die automatisch von Android Studio erstellt wird. Wenn Sie das Projekt ohne Android Studio erstellen müssen, benötigen Sie eine Zeile, die mit `sdk.dir=` gefolgt vom Pfad zu Ihrer SDK-Installation.

Öffnen Sie eine Shell und gehen Sie in das Projektverzeichnis. Geben Sie `./gradlew aR` und drücken Sie die Eingabetaste. `aR` ist eine Abkürzung für `assembleRelease` , die alle Abhängigkeiten für Sie herunterlädt und die App erstellt. Die endgültige APK-Datei befindet sich in `ProjectName/ModuleName/build/outputs/apk` und heißt `ModuleName-release.apk` .

Ein neues Projekt erstellen

Android Studio einrichten

Beginnen Sie mit dem Einrichten von Android Studio und öffnen Sie es. Jetzt können Sie Ihre erste Android-App erstellen!

> **Hinweis:** Dieses Handbuch basiert auf Android Studio 2.2, der Prozess in anderen Versionen ist jedoch im Wesentlichen derselbe.

Konfigurieren Sie Ihr Projekt

Grundlegende Einstellung

Sie können ein neues Projekt auf zwei Arten starten:

- Klicken `Start a New Android Studio Project` im Begrüßungsbildschirm auf `Start a New Android Studio Project` .
- Navigieren Sie zu `File` → `New Project` wenn Sie bereits ein Projekt geöffnet haben.

Als Nächstes müssen Sie Ihre Bewerbung beschreiben, indem Sie einige Felder ausfüllen:

1. **Anwendungsname** - Dieser Name wird dem Benutzer angezeigt.

   Beispiel: `Hello World` . Sie können es später `AndroidManifest.xml` Datei `AndroidManifest.xml` ändern.

2. **Unternehmensdomäne** - Dies ist das Qualifikationsmerkmal für den Paketnamen Ihres Projekts.

   Beispiel: `stackoverflow.com` .

3. **Paketname** (aka `applicationId`) - Dies ist die *vollständig qualifizierten* Projektpaketnamen.

   Es sollte *Reverse Domain Name Notation* (auch *Reverse DNS genannt*) folgen: *Top Level Domain* **.** ***Unternehmensdomäne.*** [*Unternehmenssegment* **.**] *Anwendungsname* .

   Beispiel: `com.stackoverflow.android.helloworld` oder `com.stackoverflow.helloworld` . Sie können Ihre *applicationId* jederzeit ändern, indem Sie sie in Ihrer Gradle-Datei überschreiben .

   Verwenden Sie nicht das Standardpräfix "com.example", es sei denn, Sie möchten Ihre Bewerbung nicht im Google Play Store einreichen. Der Paketname ist Ihre eindeutige **applicationId** in Google Play.

4. **Projektspeicherort** - Dies ist das Verzeichnis, in dem Ihr Projekt gespeichert wird.

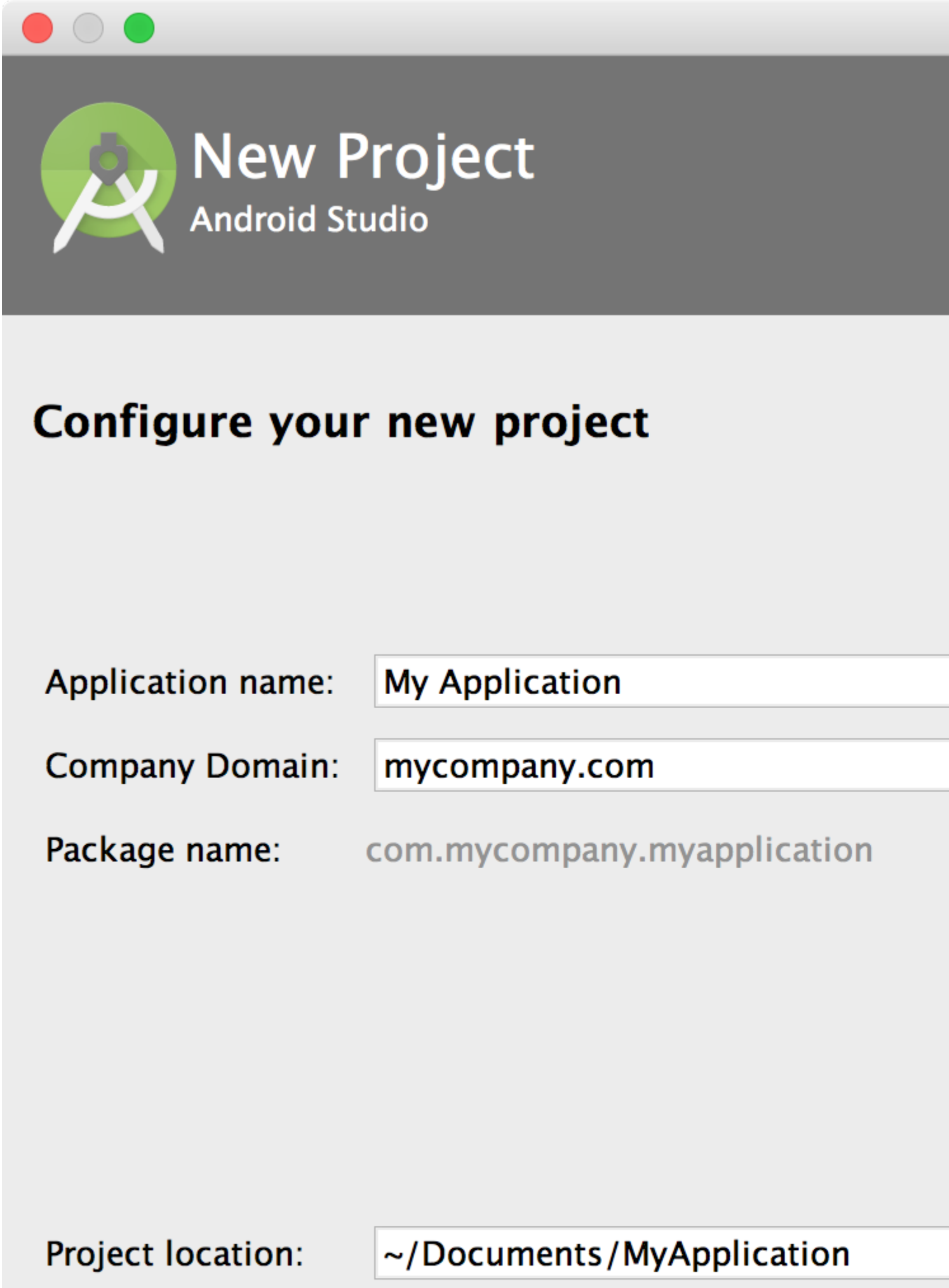
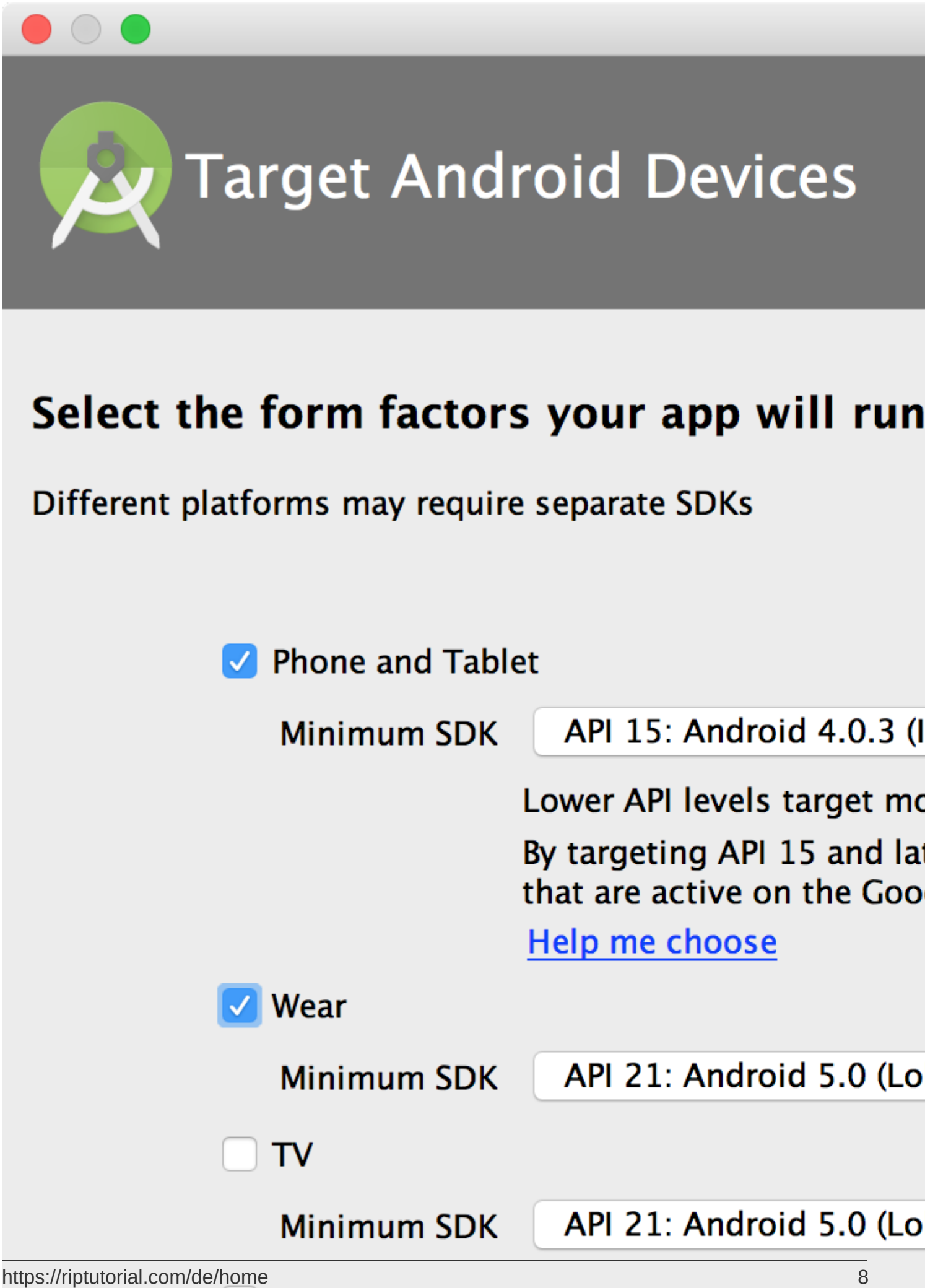

New Project
Android Studio
Configure your new project
Application name:
My Application
Company Domain:
mycompany.com
Package name:
com.mycompany.myapplication
Project location:
~/Documents/MyApplication

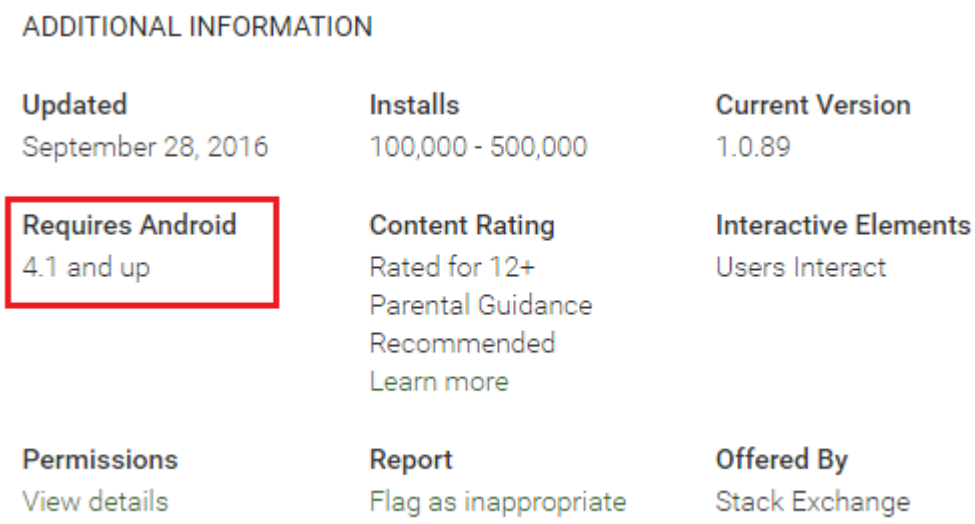
Diagramm der aktuellen Android-Versionen, angezeigt, wenn Sie auf "Hilfe bei der Auswahl" klicken.

Das Android Platform Distribution-Fenster zeigt die Verteilung mobiler Geräte, auf denen jede Android-Version ausgeführt wird (siehe Abbildung 2). Klicken Sie auf eine API-Ebene, um eine Liste der Funktionen anzuzeigen, die in der entsprechenden Android-Version eingeführt wurden. Auf diese Weise können Sie den Mindest-API-Level auswählen, der alle Funktionen enthält, die Ihre Apps benötigen, sodass Sie so viele Geräte wie möglich erreichen können. Klicken Sie dann auf **OK** .

Wählen Sie nun aus, welche Plattformen und welche Version von Android SDK die Anwendung unterstützt.


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

`Google Play Store` verwendet, um festzustellen, auf welchen Geräten eine App installiert werden kann. Die App von Stack Exchange unterstützt beispielsweise Android 4.1 und höher.



Android Studio teilt Ihnen (ungefähr) mit, wie viel Prozent der Geräte mit dem angegebenen Mindest-SDK unterstützt werden.

> Niedrigere API-Level zielen auf mehr Geräte ab, verfügen jedoch über weniger verfügbare Funktionen.

Bei der Entscheidung für das **Minimum-SDK** sollten Sie die Dashboards-Statistiken in Betracht ziehen, die Ihnen Versionsinformationen zu den Geräten geben, die den Google Play Store in der letzten Woche weltweit besucht haben.

Von: Dashboards auf der Android Developer-Website.

Aktivität hinzufügen

Jetzt werden wir eine Standardaktivität für unsere Anwendung auswählen. In Android ist eine `Activity` ein einzelner Bildschirm, der dem Benutzer angezeigt wird. Eine Anwendung kann mehrere Aktivitäten enthalten und zwischen diesen navigieren. Wählen Sie für dieses Beispiel `Empty Activity` und klicken Sie auf Weiter.

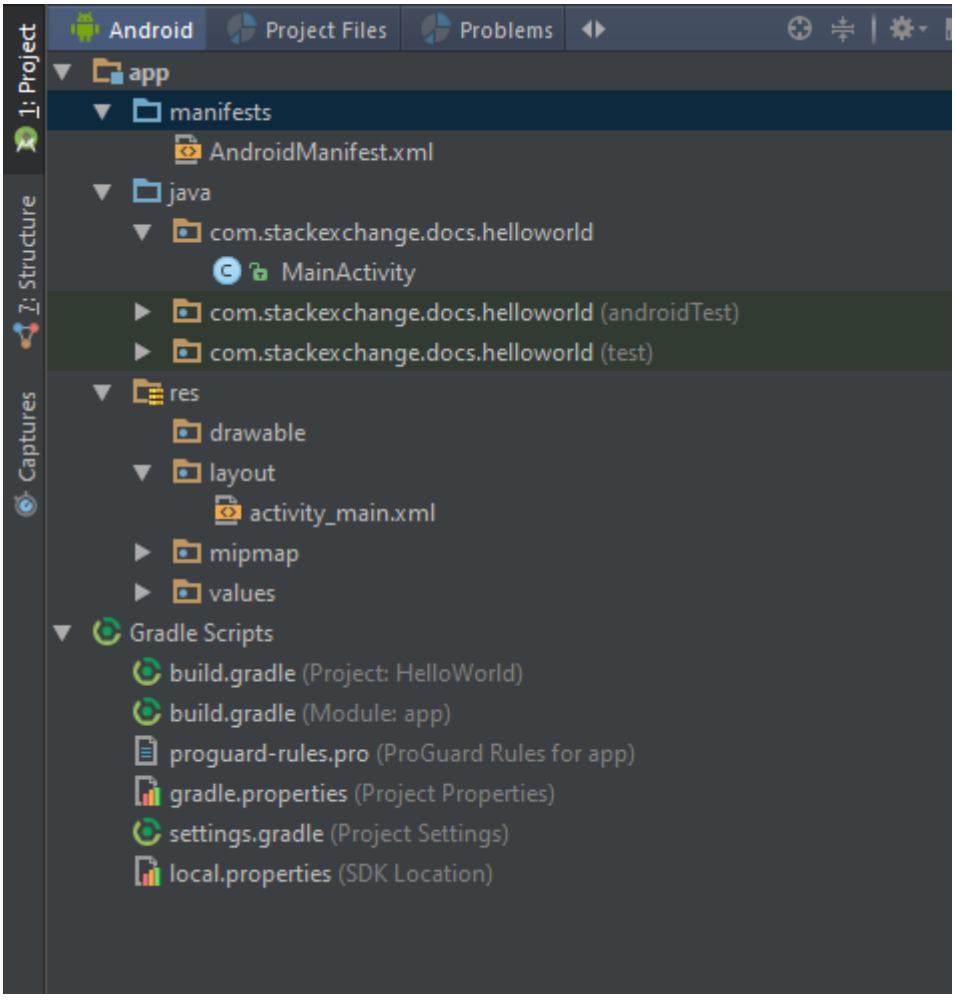
Wenn Sie möchten, können Sie hier den Namen der Aktivität und des Layouts ändern. Es empfiehlt sich, `Activity` als Suffix für den Aktivitätsnamen und `activity_` als Präfix für den Layoutnamen beizubehalten. Wenn wir diese Standardeinstellungen `MainActivity` , generiert Android Studio eine Aktivität mit dem Namen `MainActivity` und eine Layoutdatei mit dem Namen `activity_main` . Klicken Sie jetzt auf `Finish` .

Android Studio erstellt und konfiguriert unser Projekt, das je nach System einige Zeit dauern kann.

Inspektion des Projekts

Um zu verstehen, wie Android funktioniert, werfen wir einen Blick auf einige der Dateien, die für uns erstellt wurden.

Im linken Bereich von Android Studio sehen wir die Struktur unserer Android-Anwendung .



Zuerst öffnen `AndroidManifest.xml` mit einem Doppelklick. Die Android-Manifestdatei beschreibt einige grundlegende Informationen zu einer Android-Anwendung. Es enthält die Erklärung unserer Aktivitäten sowie einige fortgeschrittenere Komponenten.

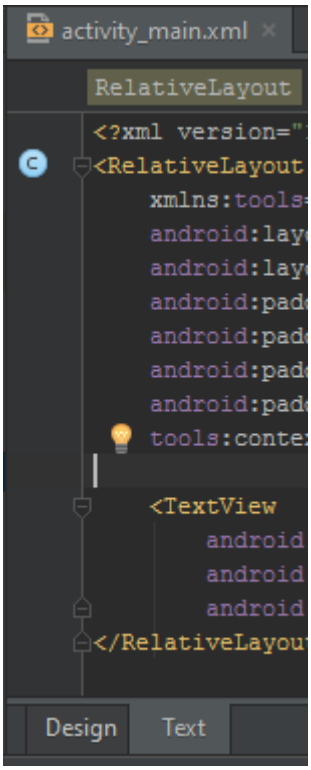
Wenn eine Anwendung Zugriff auf eine durch eine Berechtigung geschützte Funktion benötigt, muss sie erklären, dass diese Berechtigung mit einem Element `<uses-permission>` im Manifest erforderlich ist. Wenn die Anwendung auf dem Gerät installiert ist, bestimmt das

Installationsprogramm, ob die angeforderte Berechtigung erteilt werden soll, indem die Behörden überprüft werden, die die Zertifikate der Anwendung signiert haben, und in einigen Fällen den Benutzer gefragt. Eine Anwendung kann auch ihre eigenen Komponenten (Aktivitäten, Dienste, Rundfunkempfänger und Inhaltsanbieter) mit Berechtigungen schützen. Es kann alle von Android definierten Berechtigungen verwenden (aufgeführt in android.Manifest.permission) oder von anderen Anwendungen deklariert. Oder es kann sein eigenes definieren.

```
<?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>
```

Als nächstes öffnen wir die Datei `activity_main.xml` die sich in `app/src/main/res/layout/` . Diese Datei enthält Deklarationen für die visuellen Komponenten unserer MainActivity. Sie sehen Visual Designer. Auf diese Weise können Sie Elemente per Drag & Drop in das ausgewählte Layout ziehen.

Sie können auch zum XML-Layout-Designer wechseln, indem Sie unten in Android Studio auf "Text" klicken.




```
<?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>
```

In diesem Layout wird ein Widget mit dem Namen " `TextView` Eigenschaft " `android:text` " ist auf "Hello World!" `TextView` . Dies ist ein Textblock, der dem Benutzer beim Ausführen der Anwendung angezeigt wird.

Sie können mehr über Layouts und Attribute lesen.

Lassen Sie uns als Nächstes einen Blick auf `MainActivity` . Dies ist der Java-Code, der für `MainActivity` generiert wurde.

```
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);
    }
}
```

Wie in unserem Android-Manifest definiert, wird `MainActivity` standardmäßig gestartet, wenn ein Benutzer die `HelloWorld` App startet.

Öffnen Sie zuletzt die Datei `build.gradle` in `app/` .
Android Studio verwendet das Build-System **Gradle** zum Kompilieren und Erstellen von Android-Anwendungen und -Bibliotheken.

```
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'
}
```

Diese Datei enthält Informationen zum Build und zur App-Version. Sie können damit auch Abhängigkeiten zu externen Bibliotheken hinzufügen. Lassen Sie uns zunächst keine Änderungen vornehmen.

Es ist ratsam, immer die aktuellste Version auszuwählen, die für die Abhängigkeiten verfügbar ist:

- buildToolsVersion : 26.0.0

- com.android.support:appcompat-v7 : 26.0.0 (Juli 2017)
- Feuerbasis : 11.0.4 (August 2017)

compileSdkVersion

`compileSdkVersion` gibt *Gradle an, mit* welcher Version des Android SDK Ihre App kompiliert werden kann. Die Verwendung des neuen Android SDK ist eine Voraussetzung für die Verwendung einer der neuen APIs, die auf dieser Ebene hinzugefügt wurden.

Es sollte betont werden, dass das Ändern der `compileSdkVersion` das Laufzeitverhalten nicht ändert. Während beim Ändern Ihrer `compileSdkVersion` neue Warnungen / Fehler des Compilers `compileSdkVersion` , ist Ihre `compileSdkVersion` nicht in Ihrem APK enthalten: Sie wird nur zur Kompilierzeit verwendet.

Daher wird dringend empfohlen, dass Sie immer mit dem neuesten SDK kompilieren. Sie erhalten alle Vorteile der neuen Kompilierungsprüfungen für vorhandenen Code, vermeiden neu veraltete APIs und sind bereit, neue APIs zu verwenden.

minSdkVersion

Wenn `compileSdkVersion` die neuesten verfügbaren APIs `minSdkVersion` ist `minSdkVersion` die *untere Grenze* für Ihre App. Die `minSdkVersion` ist eines der Signale, die der Google Play Store verwendet, um festzustellen, auf welchen Geräten eines Nutzers eine App installiert werden kann.

Es spielt auch eine wichtige Rolle während der Entwicklung: Lint wird standardmäßig für Ihr Projekt ausgeführt und warnt Sie, wenn Sie APIs über Ihrer `minSdkVersion` . So vermeiden Sie das Laufzeitproblem beim Versuch, eine nicht vorhandene API aufzurufen. Das Überprüfen der Systemversion zur Laufzeit ist eine übliche Technik, wenn APIs nur für neuere Plattformversionen verwendet werden.

targetSdkVersion

`targetSdkVersion` ist die Hauptmethode, mit der Android Vorwärtskompatibilität bereitstellt, indem Verhaltensänderungen nur `targetSdkVersion` wenn die `targetSdkVersion` aktualisiert wird. Auf diese Weise können Sie neue APIs verwenden, bevor Sie die Verhaltensänderungen durcharbeiten. Die Aktualisierung auf das neueste SDK sollte für jede App eine hohe Priorität haben. Das bedeutet nicht, dass Sie jedes neu eingeführte Feature verwenden müssen oder Ihre `targetSdkVersion` blind `targetSdkVersion` ohne zu testen.

targetSDKVersion ist die Android-Version, die die Obergrenze für die verfügbaren Tools darstellt. Wenn targetSDKVersion kleiner als 23 ist, muss die App zur Laufzeit keine Berechtigungen für eine Instanz anfordern, selbst wenn die App auf API 23+ ausgeführt wird. TargetSDKVersion verhindert **nicht,** dass Android-Versionen oberhalb der ausgewählten Android-Version die App ausführen.

Weitere Informationen zum Gradle-Plugin finden Sie hier:

- Ein grundlegendes Beispiel
- Einführung in das Gradle-Plugin für Android und den Wrapper

- Einführung in die Konfiguration des build.gradle und der DSL-Methoden

Anwendung ausführen

Lassen Sie uns jetzt unsere HelloWorld-Anwendung ausführen. Sie können entweder ein virtuelles Android-Gerät ausführen (das Sie mit AVD Manager in Android Studio einrichten können, wie im folgenden Beispiel beschrieben) oder Ihr eigenes Android-Gerät über ein USB-Kabel anschließen.

Ein Android-Gerät einrichten

Um eine Anwendung von Android Studio auf Ihrem Android-Gerät auszuführen, müssen Sie `USB Debugging` in den `Developer Options` in den Einstellungen Ihres Geräts aktivieren.

`Settings > Developer options > USB debugging`

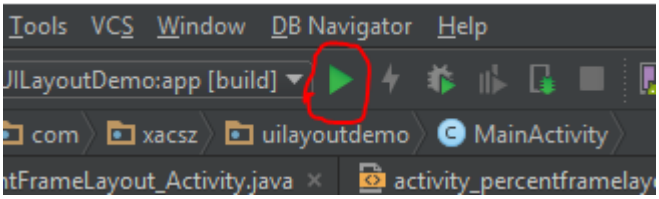
Wenn die `Developer Options` in den Einstellungen nicht sichtbar sind, navigieren Sie zu `About Phone` und tippen Sie sieben Mal auf die `Build Number` . Dadurch können `Developer Options` in Ihren Einstellungen angezeigt werden.

`Settings > About phone > Build number`

Möglicherweise müssen Sie auch die `build.gradle` Konfiguration ändern, um eine Version zu erstellen, die Ihr Gerät enthält.

Ausführen von Android Studio

Klicken Sie in der Symbolleiste oben in Android Studio auf die grüne Schaltfläche `Run` . Wählen Sie im daraufhin angezeigten Fenster das Gerät aus, auf dem Sie die App ausführen möchten (starten Sie ggf. ein virtuelles Android-Gerät oder lesen Sie Einrichten eines AVD (Android Virtual Device), falls Sie eines einrichten müssen), und klicken Sie auf `OK` .



Bei Geräten mit Android 4.4 (KitKat) und möglicherweise höher wird ein Popup-Fenster angezeigt, um das USB-Debugging zu autorisieren. Klicken Sie zur Bestätigung auf `OK` .

Die Anwendung wird nun auf Ihrem Android-Gerät oder Emulator installiert und ausgeführt.

APK-Dateispeicherort

Wenn Sie Ihre Anwendung für die Veröffentlichung vorbereiten, konfigurieren, erstellen und testen Sie eine Release-Version Ihrer Anwendung. Die Konfigurationsaufgaben sind unkompliziert und

umfassen grundlegende Aufgaben zur Code-Bereinigung und Code-Änderung, die zur Optimierung Ihrer Anwendung beitragen. Der Build-Prozess ähnelt dem Debug-Build-Prozess und kann mit JDK- und Android-SDK-Tools ausgeführt werden. Die Testaufgaben dienen als letzte Prüfung, um sicherzustellen, dass Ihre Anwendung unter realen Bedingungen wie erwartet funktioniert. Wenn Sie mit der Vorbereitung Ihrer Anwendung für die Veröffentlichung fertig sind, haben Sie eine signierte APK-Datei, die Sie direkt an die Benutzer verteilen oder über einen Anwendungsmarktplatz wie Google Play verteilen können.

Android Studio

Da in den obigen Beispielen Gradle verwendet wird, lautet der Speicherort der generierten APK-Datei: `<Your Project Location>/app/build/outputs/apk/app-debug.apk`

IntelliJ

Wenn Sie vor dem Wechsel zu Studio IntelliJ verwenden und Ihr IntelliJ-Projekt direkt importieren, hat sich nichts geändert. Die Position der Ausgabe ist unter:

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

Hinweis: Dies wird manchmal um 1,0 veraltet

Finsternis

Wenn Sie ein Android Eclipse-Projekt direkt importieren, tun Sie dies nicht! Sobald Sie Abhängigkeiten in Ihrem Projekt haben (Jars oder Library Projects), funktioniert dies nicht und Ihr Projekt wird nicht richtig eingerichtet. Wenn Sie keine Abhängigkeiten haben, befindet sich der apk an derselben Position wie in Eclipse:

```
bin/...
```

Android-Programmierung ohne IDE

Dies ist ein minimalistisches Hello World-Beispiel , das nur die grundlegendsten Android-Tools verwendet.

Anforderungen und Annahmen

- Oracle JDK 1.7 oder höher
- Android SDK-Tools (nur die Befehlszeilen-Tools)

Dieses Beispiel setzt Linux voraus. Möglicherweise müssen Sie die Syntax für Ihre eigene Plattform anpassen.

Einrichten des Android SDK

Nach dem Auspacken der SDK-Version:

1. Installieren Sie zusätzliche Pakete mit dem SDK-Manager. Verwenden Sie kein `android update sdk --no-ui` wie in der mitgelieferten Readme.txt-Anweisung beschrieben. Es werden etwa 30 GB nicht benötigte Dateien heruntergeladen. Verwenden Sie stattdessen den interaktiven SDK-Manager `android sdk` , um das empfohlene Minimum an Paketen zu erhalten.

2. Hängen Sie die folgenden JDK- und SDK-Verzeichnisse an Ihren Ausführungspfad PATH. Dies ist optional, aber die nachfolgenden Anweisungen gehen davon aus.

    - JDK / bin
    - SDK / Plattform-Tools
    - SDK / Tools
    - SDK / Build-Tools / LATEST *(wie in Schritt 1 installiert)*

3. Erstellen Sie ein virtuelles Android-Gerät. Verwenden Sie den interaktiven AVD Manager (`android avd` AVD). Möglicherweise müssen Sie ein wenig nachgeben und nach Rat suchen. Die Anweisungen vor Ort sind nicht immer hilfreich.

    (Sie können auch Ihr eigenes Gerät verwenden)

4. Führen Sie das Gerät aus:

    ```
    emulator -avd DEVICE
    ```

5. Wenn der Gerätebildschirm gesperrt zu sein scheint, streichen Sie, um ihn zu entsperren.

    Lassen Sie es laufen, während Sie die App codieren.

App kodieren

6. Wechseln Sie in ein leeres Arbeitsverzeichnis.

7. Machen Sie die Quelldatei:

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

    Inhalt:

    ```
    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. Fügen Sie ein Manifest hinzu:

```
touch AndroidManifest.xml
```

Inhalt:

```
<?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. Erstellen Sie ein Unterverzeichnis für die deklarierten Ressourcen:

```
mkdir res
```

Lass es für jetzt leer.

Code erstellen

10. Generieren Sie die Quelle für die Ressourcendeklarationen. Ersetzen Sie hier den korrekten Pfad zu Ihrem **SDK** und die zu installierende **API** (z. B. "android-23"):

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

Ressourcendeklarationen (weiter unten beschrieben) sind eigentlich optional. In der Zwischenzeit macht der obige Anruf nichts, wenn res / noch leer ist.

11. Kompilieren Sie den Quellcode in Java-Bytecode (.java → .class):

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

12. Übersetzen Sie den Bytecode von Java auf Android (.class → .dex):

    Zuerst mit Jill (.class → .jayce):

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

    Dann Jack (.jayce → .dex):

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

    Android-Bytecode wurde früher als "ausführbarer Dalvik-Code" und daher als "Dex" bezeichnet.

    Sie können die Schritte 11 und 12 durch einen einzigen Anruf bei Jack ersetzen, wenn Sie möchten. Es kann direkt aus Java-Quellen kompiliert werden (.java → .dex). Das Kompilieren mit `javac` bietet jedoch Vorteile. Es ist ein bekanntes, besser dokumentiertes und umfassender anwendbares Werkzeug.

13. Packen Sie die Ressourcendateien einschließlich des Manifests zusammen:

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

    Dies führt zu einer teilweisen APK-Datei (Android-Anwendungspaket).

14. `ApkBuilder` Sie den vollständigen APK mit dem `ApkBuilder` Tool:

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

    Es warnt: "DIESES WERKZEUG IST ABGESCHRIEBEN. Weitere Informationen finden Sie unter --help." Wenn `--help` mit einer `ArrayIndexOutOfBoundsException` , übergeben Sie stattdessen keine Argumente:

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

    Es erklärt, dass die CLI (`ApkBuilderMain`) zugunsten des direkten Aufrufs der Java-API (`ApkBuilder`) `ApkBuilder` . (Wenn Sie wissen, wie Sie dies über die Befehlszeile tun, aktualisieren Sie bitte dieses Beispiel.)

15. Optimieren Sie die Datenanpassung der APK (empfohlene Vorgehensweise):

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

Installieren und ausführen

16. Installieren Sie die App auf dem Android-Gerät:

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

17. Starten Sie die App:

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

    Es sollte laufen und Hallo sagen.

Das ist alles. Das ist es, was man braucht, um mit den grundlegenden Android-Tools Hallo zu sagen.

Ressource deklarieren

Dieser Abschnitt ist optional. Für eine einfache "Hallo Welt" -App sind keine Ressourcendeklarationen erforderlich. Wenn sie für Ihre App auch nicht erforderlich sind, können Sie den Build etwas rationalisieren, indem Sie Schritt 10 auslassen und den Verweis auf das Verzeichnis res / aus Schritt 13 entfernen.

Ansonsten finden Sie hier ein kurzes Beispiel, wie Sie eine Ressource deklarieren und darauf verweisen können.

18. Fügen Sie eine Ressourcendatei hinzu:

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

    Inhalt:

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

19. Verweisen Sie auf die Ressource aus dem XML-Manifest. Dies ist ein deklarativer Referenzstil:

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

20. Verweisen Sie auf dieselbe Ressource aus der Java-Quelle. Dies ist ein zwingender

Hinweis:

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

21. Testen Sie die obigen Änderungen, indem Sie die App neu erstellen, erneut installieren und erneut ausführen (Schritte 10-17).

    Es sollte neu starten und sagen: "Diese App heißt Hallo sagen".

Deinstallation der App

```
adb uninstall dom.domain
```

Siehe auch

- ursprüngliche Frage - Die ursprüngliche Frage, die zu diesem Beispiel geführt hat
- Arbeitsbeispiel - Ein funktionierendes Build-Skript, das die obigen Befehle verwendet

Anwendungsgrundlagen

Android Apps sind in Java geschrieben. Die Android SDK-Tools kompilieren die Code-, Daten- und Ressourcendateien in einem APK (Android-Paket). Im Allgemeinen enthält eine APK-Datei den gesamten Inhalt der App.

Jede App wird auf einer eigenen virtuellen Maschine (VM) ausgeführt, sodass die App getrennt von anderen Apps ausgeführt werden kann. Das Android-System arbeitet mit dem Prinzip des geringsten Privilegs. Jede App hat nur Zugriff auf die Komponenten, die sie für ihre Arbeit benötigt, und nicht mehr. Es gibt jedoch Möglichkeiten für eine App, Daten mit anderen Apps gemeinsam zu nutzen, z. B. durch die gemeinsame Nutzung der Linux-Benutzer-ID zwischen Apps, oder Apps können die Berechtigung zum Zugriff auf Gerätedaten wie SD-Karte, Kontakte usw. anfordern.

App-Komponenten

App-Komponenten sind die Bausteine einer Android-App. Jede Komponente spielt eine bestimmte Rolle in einer Android-App, die einem bestimmten Zweck dient und unterschiedliche Lebenszyklen hat (der Fluss, wie und wann die Komponente erstellt und zerstört wird). Hier sind die vier Arten von App-Komponenten:

1. **Aktivitäten:** Eine Aktivität repräsentiert einen einzelnen Bildschirm mit einer Benutzeroberfläche (UI). Eine Android-App kann mehrere Aktivitäten haben. (Eine E-Mail-App kann beispielsweise eine Aktivität enthalten, um alle E-Mails aufzulisten, eine andere, um den Inhalt jeder E-Mail anzuzeigen, und eine andere, um neue E-Mail zu erstellen.) Alle

Aktivitäten in einer App arbeiten zusammen, um ein User eXperience (UX) zu erstellen.
2. **Dienste:** Ein Dienst wird im Hintergrund ausgeführt, um lange andauernde Vorgänge auszuführen oder um Arbeit für Remote-Prozesse auszuführen. Ein Dienst stellt keine Benutzeroberfläche bereit, er wird nur im Hintergrund mit der Eingabe des Benutzers ausgeführt. (Ein Dienst kann beispielsweise Musik im Hintergrund abspielen, während sich der Benutzer in einer anderen App befindet, oder er kann Daten aus dem Internet herunterladen, ohne die Interaktion des Benutzers mit dem Android-Gerät zu blockieren.)
3. **Inhaltsanbieter:** Ein Inhaltsanbieter verwaltet freigegebene Anwendungsdaten. Es gibt vier Möglichkeiten, Daten in einer App zu speichern: Sie können Daten in eine Datei schreiben und im Dateisystem speichern, in eine SQLite-Datenbank einfügen oder aktualisieren, im Web bereitstellen oder an einem anderen dauerhaften Speicherort speichern, auf den die App zugreifen kann . Über Content Provider können andere Apps die Daten abfragen oder sogar ändern. (Ein Android-System stellt beispielsweise einen Inhaltsanbieter bereit, der die Kontaktinformationen des Benutzers verwaltet, sodass jede Anwendung, die über eine Berechtigung verfügt, die Kontakte abfragen kann.) Inhaltsanbieter können auch zum Speichern der für die Anwendung privaten Daten verwendet werden, um die Datenintegrität zu verbessern.
4. **Rundfunkempfänger:** Ein Rundfunkempfänger reagiert auf systemweite Rundsendungen von Ansagen (z. B. eine Sendung, die besagt, dass der Bildschirm ausgeschaltet ist, der Akku schwach ist usw.) oder von Apps (z. B., um andere Apps mitzuteilen, dass einige Daten vorhanden sind auf das Gerät heruntergeladen werden und steht ihnen zur Verfügung). Broadcast-Empfänger haben keine Benutzeroberflächen, aber sie können eine Benachrichtigung in der Statusleiste anzeigen, um den Benutzer zu warnen. Normalerweise werden Rundfunkempfänger als Gateway zu anderen Komponenten der App verwendet, die hauptsächlich aus Aktivitäten und Diensten bestehen.

Ein einzigartiger Aspekt des Android-Systems ist, dass jede App eine Komponente einer anderen App starten kann (z. B. wenn Sie einen Anruf tätigen, SMS senden, eine Webseite öffnen oder ein Foto anzeigen möchten, gibt es eine App, die dies bereits tut, und Ihre App kann dies.) nutzen Sie es, anstatt eine neue Aktivität für dieselbe Aufgabe zu entwickeln).

Wenn das System eine Komponente startet, startet es den Prozess für diese App (wenn sie noch nicht ausgeführt wird, dh auf einem Android-System kann zu jeder Zeit nur ein Vordergrundprozess pro App ausgeführt werden) und instanziiert die für diese Komponente erforderlichen Klassen. Somit läuft die Komponente auf dem Prozess der App, zu der sie gehört. Im Gegensatz zu Apps auf anderen Systemen verfügen Android-Apps daher nicht über einen einzigen Einstiegspunkt (es gibt keine `main()` -Methode).

Da das System jede App in einem separaten Prozess ausführt, kann eine App die Komponenten einer anderen App nicht direkt aktivieren. Dies kann jedoch das Android-System tun. Um eine andere App-Komponente zu starten, muss eine App eine Nachricht an das System senden, die die Absicht angibt, diese Komponente zu starten. Anschließend startet das System diese Komponente.

Kontext

Instanzen der Klasse `android.content.Context` stellen die Verbindung zum Android-System

`android.content.Context` , das die Anwendung ausführt. Die Kontextinstanz ist erforderlich, um Zugriff auf die Ressourcen des Projekts und die globalen Informationen zur Umgebung der App zu erhalten.

Lassen Sie uns ein leicht verdauliches Beispiel haben: Stellen Sie sich vor, Sie sind in einem Hotel und möchten etwas essen. Sie rufen den Zimmerservice an und bitten sie, Ihnen Dinge zu bringen oder Dinge für Sie aufzuräumen. Stellen Sie sich dieses Hotel nun als eine Android-App vor, sich selbst als eine Aktivität, und der Zimmerservice-Mitarbeiter ist dann Ihr Kontext, der Ihnen Zugriff auf die Hotelressourcen wie Zimmerservice, Lebensmittel usw. bietet.

Noch ein anderes Beispiel: Sie sitzen in einem Restaurant auf einem Tisch, jeder Tisch hat einen Begleiter. Wenn Sie Lebensmittel bestellen möchten, bitten Sie den Begleiter, dies zu tun. Der Betreuer gibt dann Ihre Bestellung auf und Ihre Lebensmittel werden auf Ihrem Tisch serviert. Auch in diesem Beispiel handelt es sich bei dem Restaurant um eine Android-App, die Tische oder die Kunden sind App-Komponenten, die Lebensmittel sind Ihre App-Ressourcen und die Telefonzentrale ist Ihr Kontext, sodass Sie auf Ressourcen wie Lebensmittel zugreifen können.

Die Aktivierung einer der oben genannten Komponenten erfordert die Instanz des Kontexts. Nicht nur das oben Genannte, sondern fast jede Systemressource: Erstellen der Benutzeroberfläche mithilfe von Ansichten (wird später erläutert), Erstellen von Instanzen von Systemdiensten, Starten neuer Aktivitäten oder Dienste - alles erfordert Kontext.

Eine ausführlichere Beschreibung wird hier geschrieben.

Einrichten einer AVD (Android Virtual Device)

TL; DR Damit können wir echte Geräte simulieren und unsere Apps ohne echtes Gerät testen.

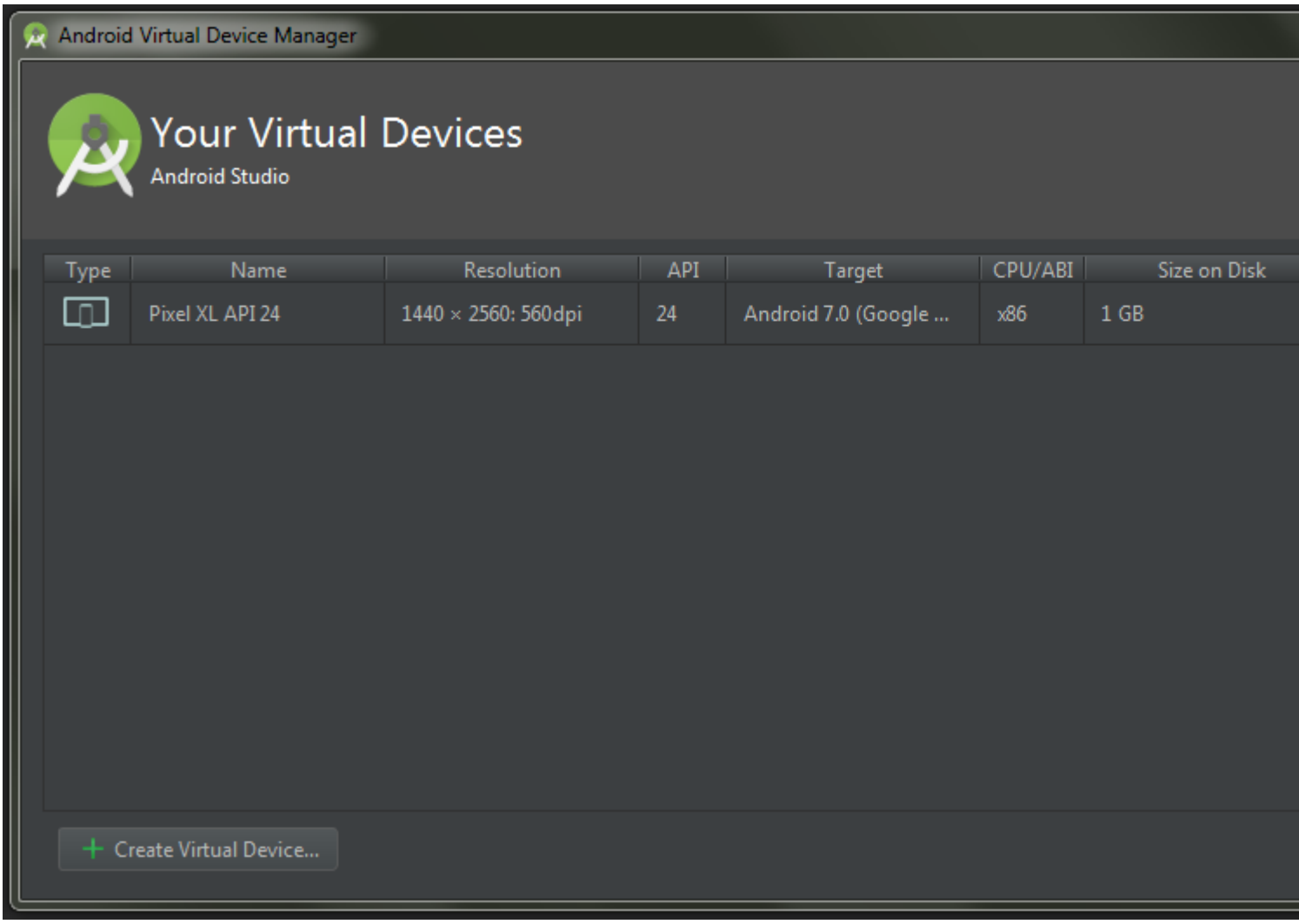
Laut der Android-Entwicklerdokumentation

> Mit einer *Definition für* ein ***virtuelles Android-Gerät (AVD)*** können Sie die Merkmale eines Android-Telefons, Tablets, Android Wear oder Android TV-Geräts definieren, die Sie im Android-Emulator simulieren möchten. Mit dem AVD Manager können Sie AVDs problemlos erstellen und verwalten.

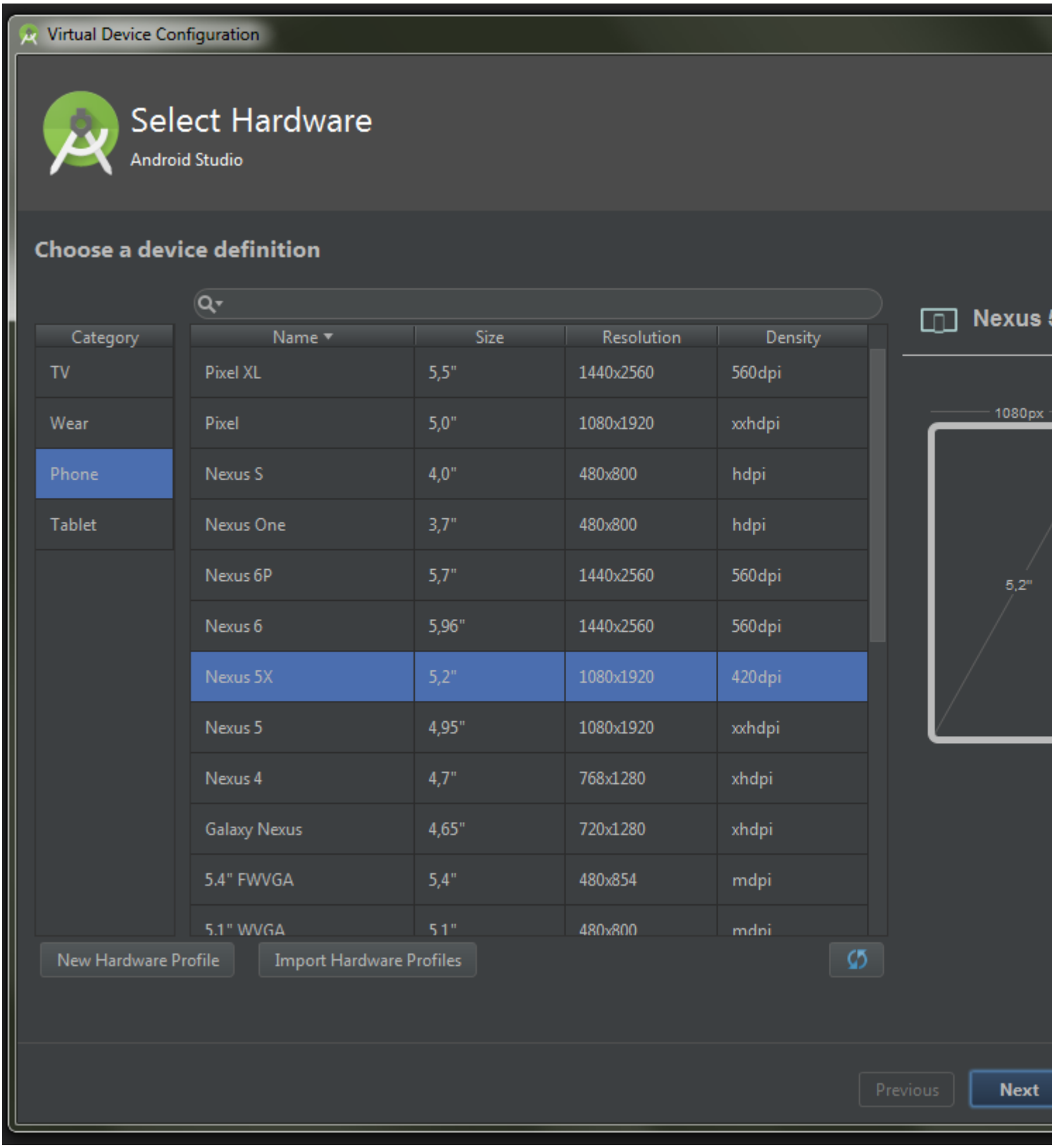
Führen Sie die folgenden Schritte aus, um eine AVD einzurichten:

1. Klicken Sie auf diese Schaltfläche, um den AVD Manager aufzurufen:

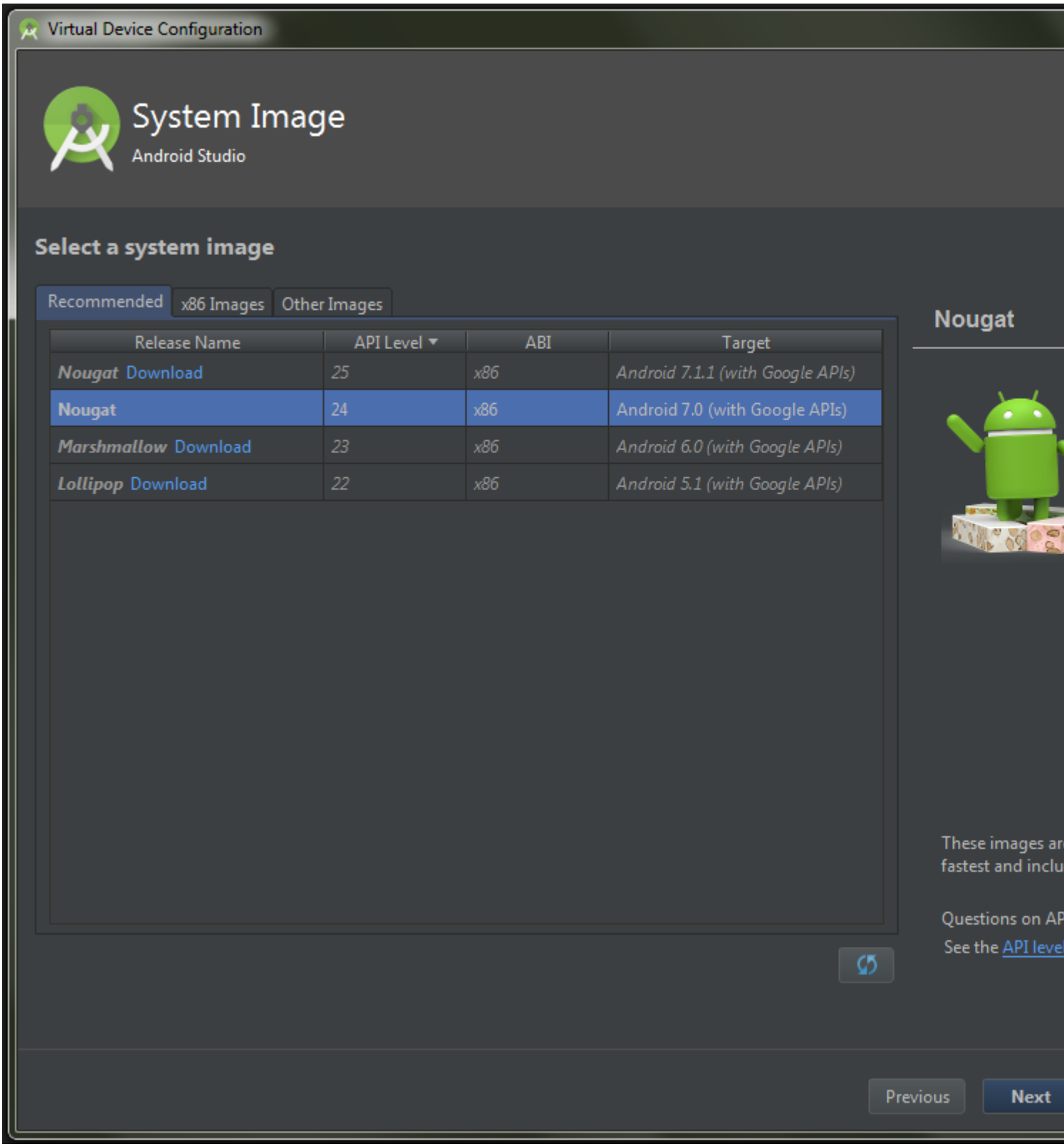
2. Sie sollten einen Dialog wie diesen sehen:



3. Klicken `+ Create Virtual Device...` nun auf die Schaltfläche `+ Create Virtual Device...` Dies öffnet den Konfigurationsdialog für virtuelle Geräte:



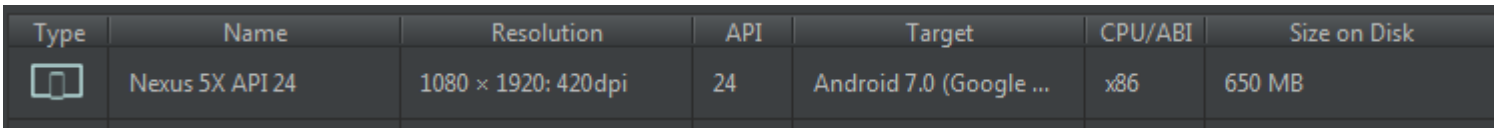
4. Wählen Sie ein beliebiges Gerät aus und klicken Sie auf `Next` :



5. Hier müssen Sie eine Android-Version für Ihren Emulator auswählen. Möglicherweise müssen Sie es auch zuerst herunterladen, indem Sie auf `Download` klicken. Nachdem Sie eine Version ausgewählt haben, klicken Sie auf `Next` .

6. Geben Sie hier einen Namen für Ihren Emulator ein, die anfängliche Ausrichtung und ob Sie einen Rahmen darum anzeigen möchten. Nachdem Sie alle ausgewählt haben, klicken Sie auf `Finish` .

7. Jetzt haben Sie eine neue AVD zum Starten Ihrer Apps bereit.

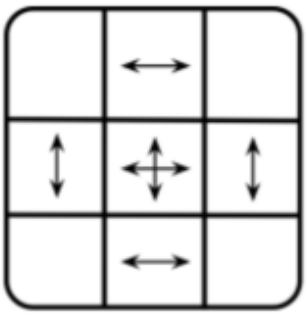


Erste Schritte mit Android online lesen: https://riptutorial.com/de/android/topic/85/erste-schritte-mit-android

Kapitel 2: 9-Patch-Bilder

Bemerkungen

Eine *9-Patch*- Bilddatei ist eine speziell formatierte Datei, damit Android weiß, welche Bereiche / Teile des Bildes skaliert werden können oder nicht. Das Bild wird in ein 3x3-Raster zerlegt. Die Ecken bleiben nicht skaliert, die Seiten werden in eine Richtung und der Mittelpunkt in beiden Dimensionen skaliert.



Ein Neun-Patch-Bild (9-Patch) ist eine Bitmap, die einen einzigen Pixel breiten Rand um das gesamte Bild herum aufweist. Ignoriert die 4 Pixel in den Ecken des Bildes. Dieser Rahmen stellt Metadaten für die Bitmap selbst bereit. Grenzen werden durch durchgezogene schwarze Linien markiert.

Ein Neun-Patch-Image wird mit der Erweiterung `.9.png` gespeichert.

Der obere Rand kennzeichnet Bereiche, die sich horizontal erstrecken. Der linke Rand kennzeichnet Bereiche, die sich vertikal erstrecken.

Der untere Rand zeigt das Auffüllen horizontal an. Der rechte Rand zeigt vertikales Auffüllen an.

Die Auffüllränder werden normalerweise verwendet, um zu bestimmen, wo Text gezeichnet werden soll.

Es gibt ein hervorragendes Werkzeug von Google, *das* die Erstellung dieser Dateien *erheblich* vereinfacht.

Befindet sich im Android SDK: `android-sdk\tools\lib\draw9patch.jar`

Examples

Grundgerundete Ecken

Der Schlüssel zum korrekten Strecken ist im oberen und linken Rand.

Der obere Rand steuert die horizontale Dehnung und der linke Rand die vertikale Dehnung.

In diesem Beispiel werden abgerundete Ecken erstellt, die für einen Toast geeignet sind.

Die Teile des Bildes, die sich unterhalb des *oberen Randes* und rechts vom *linken Rand befinden,* werden erweitert, um den gesamten nicht verwendeten Platz zu füllen.

Dieses Beispiel erstreckt sich auf alle Größenkombinationen (siehe unten):

Grundlegender Spinner

Der `Spinner` kann mit einem Neun-Patch nach Ihren eigenen Wünschen neu gestaltet werden.

Als Beispiel sehen Sie diesen neun Patch:

Wie Sie sehen können, sind drei extrem kleine Dehnungsbereiche markiert.

Der obere Rand hat nur das markierte Symbol verlassen. Dies bedeutet, dass die linke Seite (vollständige Transparenz) des Zeichenelements die `Spinner` Ansicht füllen soll, bis das Symbol erreicht ist.

Der linke Rand hat markierte transparente Segmente oben und unten am markierten Symbol. Dies bedeutet, dass sowohl die Ober- als auch die Unterseite auf die Größe der `Spinner` Ansicht erweitert werden. Dadurch wird das Symbol vertikal vertikal zentriert.

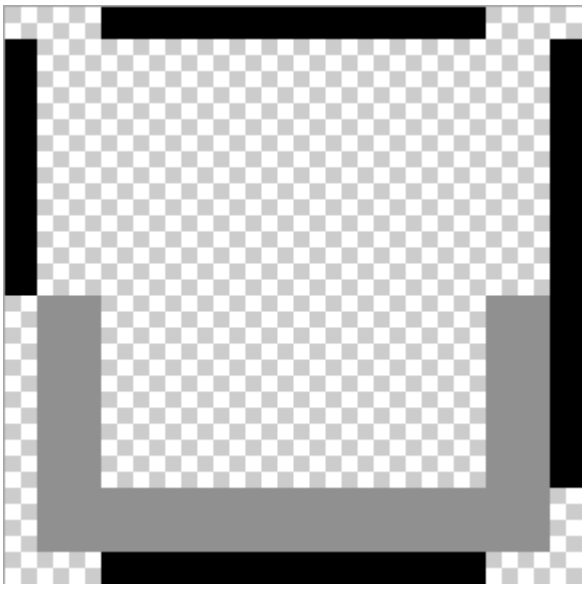
Verwenden des Bildes ohne Neun Patch-Metadaten:

Verwenden des Bildes mit den neun Patch-Metadaten:

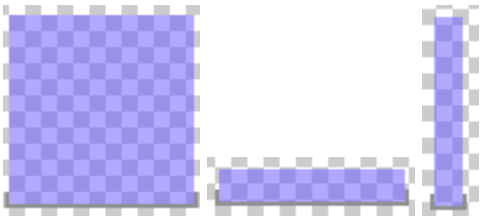
Optionale Auffüllzeilen

Bilder mit neun Patches ermöglichen die optionale Definition der Auffüllzeilen im Bild. Die Auffüllzeilen sind die Zeilen rechts und unten.

Wenn eine Ansicht das 9-Patch-Bild als Hintergrund festlegt, werden die Auffüllzeilen verwendet, um den Platz für den Inhalt der Ansicht festzulegen (z. B. die Texteingabe in einem `EditText`). Wenn die Auffüllzeilen nicht definiert sind, werden stattdessen die linke und obere Zeile verwendet.



Der Inhaltsbereich des gestreckten Bildes sieht dann so aus:



9-Patch-Bilder online lesen: https://riptutorial.com/de/android/topic/461/9-patch-bilder

Kapitel 3: Absicht

Einführung

Eine Absicht ist eine kleine Nachricht, die um das Android-System herumgeleitet wird. Diese Nachricht enthält möglicherweise Informationen zu unserer Absicht, eine Aufgabe auszuführen.

Es handelt sich im Wesentlichen um eine passive Datenstruktur, die eine abstrakte Beschreibung einer auszuführenden Aktion enthält.

Syntax

- Absicht Absicht ()
- Vorsatz (Vorsatz)
- Absicht Absicht (String-Aktion)
- Intent Intent (String-Aktion, Uri Uri)
- Intent Intent (Context packageContext, Klasse <?> Cls)
- Intent Intent (String-Aktion, Uri-Uri, KontextpaketContext, Klasse <?> Cls)
- void startActivity (Absichtsabsicht)
- void startActivity (Absichtsabsicht, Bundle-Optionen)
- void startActivityForResult (Absichtsabsicht, int requestCode)
- void startActivityForResult (Intent Intent, Int requestCode, Bundle-Optionen)
- Intention putExtra (Stringname, doppelter [] -Wert)
- Intention putExtra (Stringname, int-Wert)
- Intention putExtra (String-Name, CharSequence-Wert)
- Intention putExtra (String-Name, Char-Wert)
- Intention putExtra (Stringname, Bundle-Wert)
- Intention putExtra (String-Name, Wert für Parcelable [])
- Intention putExtra (Stringname, serialisierbarer Wert)
- Intention putExtra (Stringname, int [] -Wert)
- Intention putExtra (Stringname, Gleitkommawert)
- Intention putExtra (Stringname, Byte [] -Wert)
- Intention putExtra (Stringname, langer [] -Wert)
- Intention putExtra (Stringname, Wert für Parcelable)
- Intention putExtra (Stringname, Gleitkommawert [])
- Intention putExtra (Stringname, langer Wert)
- Intention putExtra (Stringname, String [] -Wert)
- Intention putExtra (Stringname, boolescher Wert)
- Intention putExtra (Stringname, boolescher [] -Wert)
- Intention putExtra (Stringname, kurzer Wert)
- Intention putExtra (Stringname, doppelter Wert)
- Intention putExtra (Stringname, kurzer [] -Wert)
- Intention putExtra (Stringname, Stringwert)
- Intention putExtra (Stringname, Byte-Wert)

- Intention putExtra (Stringname, char [] -Wert)
- Intention putExtra (Stringname, CharSequence [] -Wert)

Parameter

Parameter	Einzelheiten
Absicht	Die Absicht zu beginnen
Anfrage Code	Eindeutige Nummer zur Identifizierung der Anfrage
Optionen	Zusätzliche Optionen, wie die Aktivität gestartet werden soll
Name	Der Name der zusätzlichen Daten
Wert	Der Wert der zusätzlichen Daten
CHOOSE_CONTACT_REQUEST_CODE	der Code der Anforderung, um sie in der `onActivityResult` Methode zu identifizieren
Aktion	Jede über diese Absicht auszuführende Aktion, zB Intent.ACTION_VIEW
uri	data-uri, der von einer bestimmten Aktion verwendet werden soll
packageContext	Kontext zum Initialisieren der Absicht
cls	Klasse, die von dieser Absicht verwendet werden soll

Bemerkungen

Vorsichtsmaßnahmen bei der Verwendung impliziter Absichten

Beim Aufruf einer impliziten Absicht ist es immer hilfreich zu prüfen, ob es vom System möglich ist, damit umzugehen.

Dies kann durch Überprüfen mit `PackageManager.queryIntentActivities(Intent intent, int flags)`

```
PackageManager pm = getActivity().getPackageManager();
if (intent.resolveActivity(pm) != null) {
```

```
    //intent can be handled
    startActivity(intent);
} else {
     //intent can not be handled
}
```

`singleTask` der es sich um einen `singleTask` oder `singleTop`

Wenn der Startmodus der Aktivität `singleTask` oder `singleTop` , wird `singleTask` `singleTop` , sobald die Aktivität mit einem `singleTop` `onActivityResult` wird. Um dies zu verhindern, verwenden Sie `Intent.setFlags(0)` , um die Standardflags zurückzusetzen.

Examples

Eine Aktivität starten

In diesem Beispiel wird `DestinationActivity` von `OriginActivity` aus `OriginActivity` .

Hier nimmt der `Intent` Konstruktor zwei Parameter an:

1. Ein Kontext als erster Parameter (wird verwendet, da die Activity-Klasse eine Unterklasse von Context ist)
2. Die Klasse der App-Komponente, an die das System die Absicht liefern soll (in diesem Fall die Aktivität, die gestartet werden soll)

```
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.
    }
}
```

Eine andere Methode zum Erstellen des `Intent` zum Öffnen von `DestinationActivity` besteht darin, den Standardkonstruktor für den `Intent` und die `setClass()` Methode zu verwenden, um `setClass()` , welche Aktivität geöffnet werden soll:

```
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
```

Weitergabe von Daten zwischen Aktivitäten

Dieses Beispiel zeigt das Senden eines `String` mit dem Wert `"Some data!"` von `OriginActivity` zu `DestinationActivity` .

HINWEIS: Dies ist der einfachste Weg, Daten zwischen zwei Aktivitäten zu senden. Im Beispiel zur Verwendung des Startermusters finden Sie eine robustere Implementierung.

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);
    }
}
```

Es ist auch möglich , andere passieren `primitive` Datentypen sowie `arrays` , `Bundle` und `Parcelable` Daten. Das Passieren von `Serializable` ist ebenfalls möglich, sollte jedoch vermieden werden, da es mehr als dreimal langsamer ist als `Parcelable` .

Serializable ist eine Standard-Java- `interface` . Sie können eine Klasse einfach als `Serializable` markieren, indem Sie die `Serializable interface` implementieren `interface` und Java wird sie in den

erforderlichen Situationen automatisch serialisieren.

Parcelable ist eine Android-spezifische `interface` die für benutzerdefinierte Datentypen (z. B. eigene Objekte / POJO-Objekte) implementiert werden kann. **Dadurch** kann Ihr Objekt **reduziert** werden und sich selbst rekonstruieren, ohne dass das Ziel etwas tun muss. Es gibt ein Dokumentationsbeispiel , um ein Objekt parcelabel zu machen .

Sobald Sie ein `parcelable` Objekt haben, können Sie es wie ein primitiver Typ mit einem Intent-Objekt senden:

```
intent.putExtra(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

Oder in einem Bundle / als Argument für ein Fragment:

```
bundle.putParcelable(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

und dann mit getParcelableExtra aus dem Intent am Ziel lesen:

```
final MyParcelableType data = intent.getParcelableExtra(EXTRA_DATA);
```

Oder wenn Sie ein Fragment aus einem Bundle einlesen:

```
final MyParcelableType data = bundle.getParcelable(EXTRA_DATA);
```

Sobald Sie ein `Serializable` Objekt haben, können Sie es in ein Intent-Objekt einfügen:

```
bundle.putSerializable(DestinationActivity.EXTRA_DATA, mySerializableObject);
```

und dann lesen Sie es auch aus dem Intent-Objekt am Ziel, wie unten gezeigt:

```
final SerializableType data = (SerializableType)bundle.getSerializable(EXTRA_DATA);
```

E-Mails senden

```
// 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
```

```
}
```

Dadurch wird eine E-Mail in einer E-Mail-App nach Wahl des Benutzers ausgefüllt.

Wenn Sie einen Anhang hinzufügen müssen, können Sie `Intent.ACTION_SEND` anstelle von `Intent.ACTION_SENDTO` . Für mehrere Anhänge können Sie `ACTION_SEND_MULTIPLE`

Ein Wort der Vorsicht: Nicht jedes Gerät hat einen Provider für `ACTION_SENDTO` , und der Aufruf von `startActivity()` ohne `resolveActivity()` Überprüfung mit `resolveActivity()` kann eine ActivityNotFoundException `resolveActivity()` .

Ein Ergebnis aus einer anderen Aktivität erhalten

Mit `startActivityForResult(Intent intent, int requestCode)` Sie eine andere `Activity` starten und dann ein Ergebnis dieser `Activity` in der `onActivityResult(int requestCode, int resultCode, Intent data)` . Das Ergebnis wird als `Intent` . Ein Intent kann Daten über ein Bundle enthalten

---

In diesem Beispiel `MainActivity` eine `DetailActivity` und erwartet dann ein Ergebnis. Jeder `onActivityResult(int requestCode, int resultCode, Intent data)` sollte seinen eigenen `int` Anforderungscode haben, sodass in der *überschriebenen* `onActivityResult(int requestCode, int resultCode, Intent data)` in `MainActivity` werden kann, welche Anforderung verarbeitet werden soll, indem die Werte von `requestCode` und `REQUEST_CODE_EXAMPLE` (obwohl in diesem `REQUEST_CODE_EXAMPLE` verglichen werden Beispiel gibt es nur einen).

Hauptaktivität:

```
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.
            }
        }
    }
}
```

DetailAktivität:

```
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();
    }
```

```
}
```

Ein paar Dinge, die Sie beachten sollten:

- Daten werden nur zurückgegeben, wenn Sie `finish()` aufrufen. Sie müssen `setResult()` aufrufen, bevor Sie `finish()` aufrufen. Andernfalls wird kein Ergebnis zurückgegeben.
- `android:launchMode="singleTask"` Sie sicher, dass Ihre `Activity` `android:launchMode="singleTask"` nicht verwendet `android:launchMode="singleTask"` wird die `Activity` in einer separaten Task ausgeführt. Daher erhalten Sie kein Ergebnis. Wenn Ihre `Activity singleTask` als Startmodus verwendet, wird `onActivityResult()` sofort mit dem Ergebniscode `Activity.RESULT_CANCELED onActivityResult()` .
- Seien Sie vorsichtig bei der Verwendung von `android:launchMode="singleInstance"` . Auf Geräten vor Lollipop (Android 5.0, API Level 21) wird von Aktivitäten kein Ergebnis zurückgegeben.
- Sie können explizite oder implizite Absichten verwenden, wenn Sie `startActivityForResult()` aufrufen. Wenn Sie eine Ihrer eigenen Aktivitäten starten, um ein Ergebnis zu erhalten, sollten Sie eine explizite Absicht verwenden, um sicherzustellen, dass Sie das erwartete Ergebnis erhalten. Eine explizite `intent` wird immer an das Ziel geliefert, egal was es enthält. der `filter` wird nicht konsultiert. Eine implizite Absicht wird jedoch nur dann an eine Komponente übergeben, wenn sie einen der Filter der Komponente passieren kann.

Öffnen Sie eine URL in einem Browser

Mit dem Standardbrowser öffnen

Dieses Beispiel zeigt, wie Sie eine URL programmgesteuert im integrierten Webbrowser und nicht in Ihrer Anwendung öffnen können. Dadurch kann Ihre App eine Webseite öffnen, ohne dass die `INTERNET` Berechtigung in Ihre Manifestdatei eingefügt werden muss.

```
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);
    }
}
```

Aufforderung an den Benutzer, einen

Browser auszuwählen

Beachten Sie, dass dieses Beispiel die Methode `Intent.createChooser()` verwendet:

```
public void onBrowseClick(View v) {
    String url = "http://www.google.com";
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
    // Note the Chooser below. If no applications match,
    // Android displays a system message.So here there is no need for try-catch.
    startActivity(Intent.createChooser(intent, "Browse with"));

}
```

In einigen Fällen kann die URL mit **"www" beginnen** . In diesem Fall erhalten Sie diese Ausnahme:

`android.content.ActivityNotFoundException` : Es wurde keine Aktivität zum `android.content.ActivityNotFoundException` Absicht gefunden

Die URL muss immer mit **"http: //"** oder **"https: //" beginnen** . Ihr Code sollte daher prüfen, wie im folgenden Code-Snippet gezeigt:

```
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);
}
```

Best Practices

Prüfen Sie, ob sich auf dem Gerät keine Apps befinden, die die implizite Absicht erhalten können. Andernfalls stürzt Ihre App ab, wenn sie `startActivity()` . Um zunächst zu überprüfen, ob eine App vorhanden ist, um die Absicht zu erhalten, rufen Sie `resolveActivity()` für Ihr Intent-Objekt auf. Wenn das Ergebnis nicht null ist, gibt es mindestens eine App, die die Absicht verarbeiten kann, und es ist sicher, `startActivity()` . Wenn das Ergebnis null ist, sollten Sie die Absicht nicht verwenden und wenn möglich, sollten Sie die Funktion deaktivieren, die die Absicht aufruft.

Löschen eines Aktivitätsstapels

Manchmal möchten Sie möglicherweise eine neue Aktivität starten, während Sie vorherige Aktivitäten aus dem hinteren Stapel entfernen, sodass Sie mit der Zurück-Schaltfläche nicht wieder zu ihnen zurückkehren können. Ein Beispiel hierfür könnte das Starten einer App in der Login-Aktivität sein, die Sie zur Main-Aktivität Ihrer Anwendung führt. Beim Abmelden möchten Sie jedoch zurück zum Login geleitet werden, ohne die Möglichkeit zur erneuten Anmeldung. In einem `FLAG_ACTIVITY_CLEAR_TOP` Fall können Sie das Flag `FLAG_ACTIVITY_CLEAR_TOP` für die Absicht `FLAG_ACTIVITY_CLEAR_TOP` , d. `FLAG_ACTIVITY_CLEAR_TOP` , Wenn die gerade gestartete Aktivität bereits

in der aktuellen Task (LoginActivity) ausgeführt wird, und nicht eine neue Instanz dieser Aktivität, sondern alle anderen Aktivitäten oben Davon wird geschlossen, und diese Absicht wird als (neue) Aktivität an die (jetzt oben) alte Aktivität übergeben.

```
Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
```

Sie können auch die Flags `FLAG_ACTIVITY_NEW_TASK` zusammen mit `FLAG_ACTIVITY_CLEAR_TASK` wenn Sie alle Aktivitäten auf dem `FLAG_ACTIVITY_NEW_TASK` `FLAG_ACTIVITY_CLEAR_TASK` möchten:

```
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);
```

Beabsichtigte URI

Dieses Beispiel zeigt, wie Sie die Absicht vom Browser aus starten:

```
<a href="intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end">Start
intent</a>
```

Diese Absicht startet die App mit dem Paket `com.sample.test` oder öffnet Google Play mit diesem Paket.

Auch diese Absicht kann mit Javascript gestartet werden:

```
var intent = "intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end";
window.location.replace(intent)
```

In der Aktivität können dieser Host und dieser Pfad aus Absichtsdaten erhalten werden:

```
@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    Uri data = getIntent().getData(); // returns host.com/path
}
```

Intent URI-Syntax:

```
HOST/URI-path // Optional host
#Intent;
    package=[string];
    action=[string];
    category=[string];
    component=[string];
    scheme=[string];
end;
```

Senden von Nachrichten an andere Komponenten

Absichten können verwendet werden, um Nachrichten an andere Komponenten Ihrer Anwendung (z. B. einen laufenden Hintergrunddienst) oder an das gesamte Android-System zu senden.

Verwenden `LocalBroadcastManager` zum Senden einer Übertragung **innerhalb Ihrer Anwendung** die `LocalBroadcastManager` Klasse:

```
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);
```

Um ein Broadcast an Komponenten außerhalb Ihrer Anwendung zu senden, verwenden Sie die `sendBroadcast()` -Methode für ein `Context` Objekt.

```
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);
```

Informationen zum *Empfang von* Broadcasts finden Sie hier: Broadcast Receiver

CustomTabsIntent für benutzerdefinierte Registerkarten

4.0.3

Mit einem `CustomTabsIntent` ist es jetzt möglich, benutzerdefinierte Chrome-Registerkarten zu konfigurieren, um wichtige UI-Komponenten in dem Browser anzupassen, der über Ihre App geöffnet wird.

Dies ist in einigen Fällen eine gute Alternative zur Verwendung eines WebView. Es ermöglicht das Laden einer Webseite mit einem Intent, mit der zusätzlichen Möglichkeit, ein gewisses Maß an Aussehen und Verhalten Ihrer App in den Browser einzufügen.

Hier ein Beispiel, wie Sie eine URL mit `CustomTabsIntent`

```
String url = "https://www.google.pl/";
CustomTabsIntent intent = new CustomTabsIntent.Builder()
                    .setStartAnimations(getContext(), R.anim.slide_in_right,
R.anim.slide_out_left)
                    .setExitAnimations(getContext(), android.R.anim.slide_in_left,
android.R.anim.slide_out_right)
                    .setCloseButtonIcon(BitmapFactory.decodeResource(getResources(),
R.drawable.ic_arrow_back_white_24dp))
                    .setToolbarColor(Color.parseColor("#43A047"))
                    .enableUrlBarHiding()
                    .build();
            intent.launchUrl(getActivity(), Uri.parse(url));
```

Hinweis:

Um benutzerdefinierte Registerkarten verwenden zu können, müssen Sie diese Abhängigkeit zu Ihrem build.gradle hinzufügen

```
compile 'com.android.support:customtabs:24.1.1'
```

Gemeinsame Verwendung mehrerer Dateien

Die String-Liste, die der `share()` -Methode als Parameter übergeben wird, enthält die Pfade aller Dateien, die Sie freigeben möchten.

Es durchläuft im Wesentlichen die Pfade, fügt sie zu Uri hinzu und startet die Aktivität, die Dateien dieses Typs annehmen kann.

```
  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);
    }
```

Starter-Muster

Dieses Muster ist ein strengerer Ansatz, um eine `Activity` zu starten. Ihr Zweck besteht darin, die Lesbarkeit des Codes zu verbessern und gleichzeitig die Komplexität des Codes, die Wartungskosten und die Kopplung Ihrer Komponenten zu reduzieren.

Im folgenden Beispiel wird das Startermuster implementiert, das normalerweise als statische Methode für die `Activity` selbst implementiert wird. Diese statische Methode akzeptiert alle erforderlichen Parameter, erstellt aus diesen Daten eine gültige `Intent` und startet dann die `Activity` .

Ein `Intent` ist ein Objekt, das Laufzeitbindung zwischen separaten Komponenten, z. B. zwei Aktivitäten, bereitstellt. Die Absicht repräsentiert die Absicht einer App, etwas zu tun. Sie können Absichten für eine Vielzahl von Aufgaben verwenden, aber hier beginnt Ihre Absicht mit einer anderen Aktivität.

```
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);
    }
}
```

Mit diesem Muster können Sie auch erzwingen, dass zusätzliche Daten mit der Absicht übergeben werden.

Die `ExampleActivity` kann dann wie `ExampleActivity` gestartet werden, wobei `context` ein Aktivitätskontext ist:

```
ExampleActivity.start(context, "Some data!");
```

Starten Sie Unbound Service mit einer Absicht

Ein Service ist eine Komponente, die im Hintergrund (im UI-Thread) ohne direkte Interaktion mit dem Benutzer ausgeführt wird. Ein ungebundener Dienst wird gerade gestartet und ist nicht an den Lebenszyklus einer Aktivität gebunden.

Um einen Dienst zu starten, können Sie wie im folgenden Beispiel gezeigt:

```
// 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);
```

Sie können alle Extras der Absicht verwenden, indem Sie eine `onStartCommand()` Überschreibung verwenden:

```
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;
    }

}
```

Absicht teilen

Teilen Sie einfache Informationen mit verschiedenen Apps.

```
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)));
```

Teilen Sie ein Bild mit verschiedenen Apps.

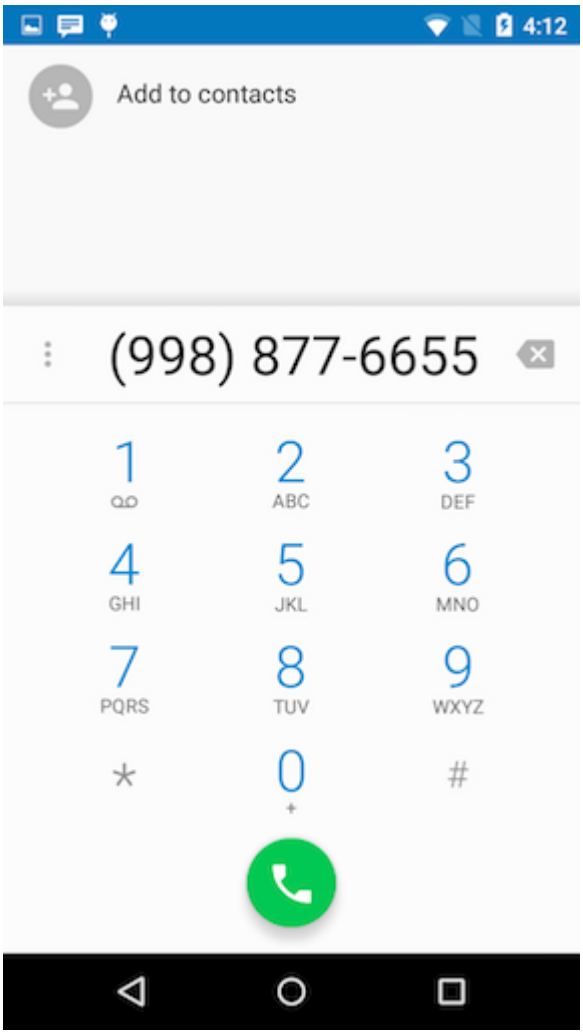
```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uriToImage);
shareIntent.setType("image/jpeg");
startActivity(Intent.createChooser(shareIntent, getResources().getText(R.string.send_to)));
```

Starten Sie den Dialer

In diesem Beispiel wird gezeigt, wie ein Standardwählgerät (eine App, die normale Anrufe tätigt) mit einer vorhandenen Telefonnummer geöffnet wird:

```
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);
```

Ergebnis durch Ausführen des obigen Codes:



Google-Karte mit angegebenem Breitengrad und Längengrad öffnen

Mit Intent können Sie Breiten- und Längengrade von Ihrer App an die Google-Karte übergeben

```
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);
```

Weitergabe verschiedener Daten durch Intent in Activity

1. Übergeben von Integer-Daten:

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. Doppeldaten übergeben:

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. String-Daten übergeben:

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");
```

oder

```
Intent mIntent = getIntent();
String stringValue = mIntent.getStringExtra("stringVariableName");
```

4. Übergeben von ArrayList-Daten:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putStringArrayListExtra("arrayListVariableName", arrayList);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
arrayList = mIntent.getStringArrayListExtra("arrayListVariableName");
```

5. Objektdaten übergeben:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("ObjectVariableName", yourObject);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
yourObj = mIntent.getSerializableExtra("ObjectVariableName");
```

> **Hinweis:** Denken Sie daran, dass Ihre benutzerdefinierte Klasse die `Serializable` Schnittstelle implementieren muss.

6. HashMap-Daten <String, String> übergeben:

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. Übergeben von Bitmap-Daten:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("image",bitmap);
startActivity(mIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
Bitmap bitmap = mIntent.getParcelableExtra("image");
```

Anzeigen einer Dateiauswahl und Lesen des Ergebnisses

Dateiauswahl-Aktivität starten

```
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);
}
```

Lesen Sie das Ergebnis

```
@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;
}
```

Benutzerdefiniertes Objekt zwischen Aktivitäten übergeben

Mit der `Bundle` Klasse können Sie Ihr benutzerdefiniertes Objekt auch an andere Aktivitäten übergeben.

Es gibt zwei Möglichkeiten:

- `Serializable` Schnittstelle - für Java und Android
- `Parcelable` Schnittstelle - speichereffizient, nur für Android (empfohlen)

Paketierbar

Die Verarbeitung per Paket ist viel schneller als die Serialisierung. Einer der Gründe dafür ist, dass wir den Serialisierungsprozess explizit machen, anstatt ihn anhand von Reflektionen abzuleiten. Es versteht sich auch, dass der Code für diesen Zweck stark optimiert wurde.

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

Code zum Senden von Aktivitäten

```
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);
```

Empfangen des Objekts in der Zielaktivität.

```
//Getting MyObjects
Intent mIntent = getIntent();
MyObjects workorder = (MyObjects) mIntent.getParcelable("UniqueKey");
```

Sie können das Arraylist of Parceble-Objekt wie folgt übergeben

```
//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");
```

> **Hinweis:** Es gibt Android Studio-Plugins wie dieses , um Parcelable-Code zu generieren

Serialisierbar

Code zum Senden von Aktivitäten

```
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);
```

Empfangen des Objekts in der Zielaktivität.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Bundle bundle = this.getIntent().getExtras();
    Product product = null;
    if (bundle != null) {
        product = (Product) bundle.getSerializable("product");
    }
```

`Arraylist` des serialisierbaren Objekts: wie beim Durchgang einzelner Objekte

Benutzerdefiniertes Objekt sollte die `Serializable` Schnittstelle implementieren.

Ergebnis von Aktivität in Fragment berechnen

Wie beim Abrufen eines Ergebnisses aus einer anderen Aktivität müssen Sie die Methode `startActivityForResult(Intent intent, int requestCode)` des `Fragment` `startActivityForResult(Intent intent, int requestCode)` . Beachten Sie, dass Sie `getActivity().startActivityForResult()` nicht

aufrufen `getActivity().startActivityForResult()` da dies das Ergebnis zur übergeordneten `Activity` des `Fragment` zurückgibt.

Das Ergebnis kann mit der `Fragment` `onActivityResult()` . Sie müssen sicherstellen, dass die übergeordnete Activity des Fragments auch `onActivityResult()` überschreibt und die Implementierung als `super` .

Im folgenden Beispiel enthält `ActivityOne` `FragmentOne` , das `ActivityTwo` startet und ein Ergebnis davon erwartet.

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
        }
    }
}
```

AktivitätZwei

```
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();
    }
}
```

Absicht online lesen: https://riptutorial.com/de/android/topic/103/absicht

Kapitel 4: ACRA

Syntax

- android: name = ". ACRAHandler"
- ACRA.init (this, config);
- öffentliche Klasse ACRAHandler erweitert Application {

Parameter

Parameter	Beschreibung
@ReportCrashes	Definiert die ACRA-Einstellungen, z. B. wo der Bericht gemeldet werden soll, benutzerdefinierte Inhalte usw
formUri	Der Pfad zu der Datei, die den Absturz meldet

Bemerkungen

- ACRA unterstützt keine Google-Formulare mehr, daher benötigen Sie ein Backend: https://github.com/ACRA/acra/wiki/Backends

Examples

ACRAHandler

Beispielanwendungserweiterende Klasse für die Handhabung des Berichtswesens:

```
@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);


    }

}
```

Beispiel Manifest

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

>



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

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



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

</manifest>
```

Installation

Maven

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

Gradle

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

ACRA online lesen: https://riptutorial.com/de/android/topic/1324/acra

Kapitel 5: ADB (Android Debug Bridge)

Einführung

ADB (Android Debug Bridge) ist ein Befehlszeilenprogramm, das zur Kommunikation mit einer Emulatorinstanz oder einem angeschlossenen Android-Gerät verwendet wird.

Überblick über ADB

Ein großer Teil dieses Themas wurde in die Adb-Shell aufgeteilt

Bemerkungen

Liste der Beispiele, die in die Adb-Shell verschoben wurden:

- Erteilen und Widerrufen von API 23+ -Berechtigungen
- Senden Sie Text, drücken Sie die Taste und berühren Sie Ereignisse über ADB an das Android-Gerät
- Pakete auflisten
- Aufzeichnen der Anzeige
- Öffnen Sie die Entwickleroptionen
- Datum / Uhrzeit über adb einstellen
- Ändern der Dateiberechtigungen mit dem Befehl chmod
- Generieren einer "Boot Complete" -Übertragung
- Anwendungsdaten drucken
- Anzeigen externer / sekundärer Speicherinhalte
- http://stackoverflow.com/documentation/android/9408/adb-shell/29140/adb-shell
- Beenden Sie einen Prozess in einem Android-Gerät

Examples

Ausführliche Liste der verbundenen Geräte drucken

Um eine ausführliche Liste aller an `adb` angeschlossenen Geräte zu erhalten, schreiben Sie den folgenden Befehl in Ihr Terminal:

```
adb devices -l
```

Beispielausgabe

```
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
```

- Die erste Spalte enthält die Seriennummer des Geräts. Wenn es mit dem `emulator-` beginnt, ist dieses Gerät ein Emulator.
- `usb:` der Pfad des Geräts im USB-Subsystem.
- `product:` Der Produktcode des Geräts. Dies ist sehr herstellerspezifisch. Wie Sie im Fall des Archos-Geräts `A50PL` oben sehen können, kann es leer sein.
- `model:` das Gerätemodell. Kann wie das `product` leer sein.
- `device:` der Gerätecode. Dies ist auch sehr herstellerspezifisch und kann leer sein.

Geräteinformationen lesen

Schreiben Sie den folgenden Befehl in Ihr Terminal:

```
adb shell getprop
```

Dadurch werden alle verfügbaren Informationen in Form von Schlüssel / Wert-Paaren gedruckt.

Sie können nur bestimmte Informationen lesen, indem Sie den Namen einer bestimmten Taste an den Befehl anhängen. Zum Beispiel:

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

Hier sind ein paar interessante Informationen, die Sie bekommen:

- `ro.product.model` : Modellname des Geräts (zB Nexus 6P)
- `ro.build.version.sdk` : API-Ebene des Geräts (zB 23)
- `ro.product.brand` : Branding des Geräts (zB Samsung)

Volle Beispielausgabe

```
[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]
```

Verbinden Sie ADB über WLAN mit einem Gerät

Die Standard-ADB-Konfiguration beinhaltet eine USB-Verbindung zu einem physischen Gerät. Wenn Sie möchten, können Sie in den TCP / IP-Modus wechseln und ADB stattdessen über WLAN verbinden.

Nicht gerootetes Gerät

1. **Holen Sie sich im selben Netzwerk:**
    - Stellen Sie sicher, dass sich Ihr Gerät und Ihr Computer im selben Netzwerk befinden.
2. **Schließen Sie das Gerät mit einem USB-Kabel an den Host-Computer an.**
3. **Verbinden Sie `adb` mit dem Gerät über das Netzwerk:**

    Wenn Ihr Gerät über USB mit `adb` verbunden ist, führen Sie den folgenden Befehl aus, um eine TCP / IP-Verbindung an einem Port zu überwachen (Standardeinstellung 5555):

    - `adb tcpip <port>` (wechseln Sie in den TCP / IP-Modus).
    - Trennen Sie das USB-Kabel vom Zielgerät.
    - `adb connect <ip address>:<port>` (Port ist optional; Standardeinstellung 5555).

    Zum Beispiel:

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

    Wenn Sie die IP-Adresse Ihres Geräts nicht kennen, können Sie:

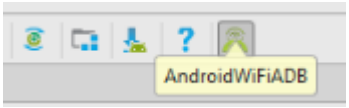
    - Überprüfen Sie die IP-Adresse in den WLAN-Einstellungen Ihres Geräts.
    - Verwenden Sie ADB zum Ermitteln von IP (über USB):
        1. Verbinden Sie das Gerät über USB mit dem Computer
        2. `adb shell ifconfig` in einer Befehlszeile `adb shell ifconfig` und kopieren Sie die IP-Adresse Ihres Geräts

    Um **wieder** zum Debuggen über **USB zurückzukehren,** verwenden Sie den folgenden

Befehl:

```
adb usb
```

Sie können ADB auch über WLAN verbinden, indem Sie ein Plugin für Android Studio installieren. *Wechseln* Sie dazu zu *Einstellungen> Plugins* und durchsuchen Sie Repositorys, suchen Sie nach *ADB WiFi* , installieren Sie es und öffnen Sie Android Studio erneut. In der Symbolleiste wird ein neues Symbol angezeigt, wie in der folgenden Abbildung dargestellt. Verbinden Sie das Gerät über USB mit dem Host-Computer und klicken Sie auf dieses *AndroidWiFiADB*- Symbol. Es wird eine Meldung angezeigt, ob Ihr Gerät angeschlossen ist oder nicht. Sobald die Verbindung hergestellt ist, können Sie den USB-Stecker entfernen.



Gerootetes Gerät

Hinweis: Einige Geräte , die **verwurzelt sind** , kann die ADB WiFi App aus dem Play Store verwenden , dies auf einfache Art und Weise zu ermöglichen. Für bestimmte Geräte (insbesondere solche mit CyanogenMod-ROMs) ist diese Option auch in den Entwickleroptionen unter den Einstellungen vorhanden. Wenn Sie diese `adb adb connect <ip address>:<port>` , erhalten Sie die IP-Adresse und die Portnummer, die für die Verbindung zu `adb` erforderlich sind, indem Sie einfach `adb connect <ip address>:<port>` ausführen.

Wenn Sie über ein gerootetes Gerät verfügen, aber keinen Zugriff auf ein USB-Kabel haben

Der Vorgang wird in der folgenden Antwort ausführlich erläutert: http://stackoverflow.com/questions/2604727/how-can-i-connect-to-android-with-adb-over-tcp/3623727#3623727 sind unten gezeigt.

Öffnen Sie ein Terminal im Gerät und geben Sie Folgendes ein:

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

Zum Beispiel:

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

Und auf deinem Computer:

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

Zum Beispiel:

```
adb connect 192.168.1.2:5555
```

Um es auszuschalten:

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

Timeout vermeiden

Standardmäßig tritt `adb` nach 5000 ms auf. Dies kann in einigen Fällen wie langsames WLAN oder große APK vorkommen.

Eine einfache Änderung in der Gradle-Konfiguration kann den Trick ausführen:

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

Ziehen Sie Dateien von (zum) Gerät

Sie können Dateien vom Gerät abrufen (herunterladen), indem Sie den folgenden Befehl ausführen:

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

Zum Beispiel:

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

Sie können auch Dateien von Ihrem Computer auf das Gerät übertragen (hochladen):

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

Zum Beispiel:

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

Beispiel zum Abrufen der Datenbank vom Gerät

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

Gerät neustarten

Sie können Ihr Gerät neu starten, indem Sie den folgenden Befehl ausführen:

```
adb reboot
```

Führen Sie diesen Befehl aus, um den Bootloader neu zu starten:

```
adb reboot bootloader
```

Neustart im Wiederherstellungsmodus:

```
adb reboot recovery
```

Beachten Sie, dass das Gerät nicht zuerst heruntergefahren wird!

Schalten Sie Wifi ein / aus

Anschalten:

```
adb shell svc wifi enable
```

Abschalten:

```
adb shell svc wifi disable
```

Verfügbare Geräte anzeigen

Befehl:

```
 adb devices
```

Ergebnisbeispiel:

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

Erste Spalte - Geräteseriennummer

Zweite Spalte - Verbindungsstatus

Android-Dokumentation

Gerät per IP verbinden

Geben Sie diese Befehle in Android - Gerät - Terminal

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

Danach können Sie **CMD** und **ADB verwenden** , um die Verbindung mit dem folgenden Befehl herzustellen

```
adb connect 192.168.0.101.5555
```

Sie können es deaktivieren und mit ADB wieder auf USB hören

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

Von einem Computer aus, wenn Sie bereits über einen USB-Zugriff verfügen (kein Root erforderlich)

Wenn Sie bereits über USB verfügen, ist es noch einfacher, auf WLAN zu wechseln. Geben Sie die Befehle über eine Befehlszeile auf dem Computer aus, an den das Gerät über USB angeschlossen ist

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

Ersetzen Sie 192.168.0.101 durch die Geräte-IP

Adb starten / stoppen

ADB starten:

```
adb kill-server
```

Stop ADB:

```
adb start-server
```

Logcat anzeigen

Sie können `logcat` als `logcat` Befehl oder direkt in einer Shell-Eingabeaufforderung Ihres Emulators oder eines verbundenen Geräts `logcat` . Um die Protokollausgabe mit `adb` anzuzeigen, navigieren Sie zu Ihrem SDK platform-tools / -Verzeichnis und führen Sie Folgendes aus

```
$ adb logcat
```

Alternativ können Sie eine Shell-Verbindung zu einem Gerät erstellen und dann Folgendes ausführen:

```
$ adb shell
$ logcat
```

Ein nützlicher Befehl ist:

```
adb logcat -v threadtime
```

Daraufhin werden Datum, Aufrufzeit, Priorität, Tag sowie PID und TID des ausstellenden Threads in einem langen Nachrichtenformat angezeigt.

Filterung

Logcat-Protokolle erhielten so genannte Protokollierungsstufen:

> **V** - Verbose, **D** - Debug, **I** - Info, **W** - Warnung, **E** - Fehler, **F** - Fatal, **S** - Stumm

Sie können Logcat auch nach Log-Level filtern. Zum Beispiel, wenn Sie nur den Debug-Level ausgeben möchten:

```
adb logcat *:D
```

Logcat kann nach einem Paketnamen gefiltert werden, natürlich können Sie es mit dem Log-Level-Filter kombinieren:

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

Sie können das Protokoll auch mit grep filtern (mehr zum Filtern der Logcat-Ausgabe hier):

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

In Windows kann der Filter beispielsweise mit findstr verwendet werden:

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

Um den alternativen Protokollpuffer [main | events | radio] `logcat` , führen Sie `logcat` mit der Option `-b` :

```
adb logcat -b radio
```

Ausgabe in Datei speichern:

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

Speichern Sie die Ausgabe in einer Datei und beobachten Sie sie gleichzeitig:

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

Reinigung der Protokolle:

```
adb logcat -c
```

Direkter ADB-Befehl an ein bestimmtes Gerät in einer Einstellung für mehrere Geräte

1. Richten Sie ein Gerät anhand der Seriennummer aus

Verwenden Sie die Option `-s` gefolgt von einem Gerätenamen, um auszuwählen, auf welchem Gerät der Befehl `adb` ausgeführt werden soll. Die `-s` Optionen sollten **vor** dem Befehl an erster Stelle stehen.

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

Beispiel:

```
adb devices

List of devices attached
emulator-5554       device
02157df2d1faeb33    device

adb -s emulator-5554 shell
```

Beispiel # 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. Zielen Sie auf ein Gerät, wenn nur ein Gerätetyp angeschlossen ist

Sie können den einzigen laufenden Emulator mit -e anvisieren

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

Oder Sie können das einzige angeschlossene USB-Gerät mit -d anvisieren

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

Screenshot: Option 1 (reine Adb)

Mit dem `shell` Befehl adb können Sie Befehle unter Verwendung der integrierten Shell eines Geräts ausführen. Mit dem Befehl `screencap` shell wird der aktuell auf einem Gerät sichtbare Inhalt erfasst und in einer bestimmten Bilddatei gespeichert, z. B. `/sdcard/screen.png` :

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

Sie können dann den Pull-Befehl verwenden , um die Datei vom Gerät in das aktuelle Verzeichnis auf Ihrem Computer herunterzuladen:

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

Screenshot: Option 2 (schneller)

Führen Sie den folgenden Einzeiler aus:

(Marshmallow und früher):

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

(Nougat und später):

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

Die `-p` - Flag leitet den Ausgang des `screencap` Befehl stdout. Der Perl-Ausdruck, in den dieser Pipe geleitet wird, bereinigt einige End-of-Line-Probleme in Marshmallow und früheren Versionen. Der Stream wird dann in eine Datei mit dem Namen `screen.png` im aktuellen Verzeichnis geschrieben. Weitere Informationen finden Sie in diesem Artikel und in diesem Artikel .

Video

Dies funktioniert nur in KitKat und nur über ADB. Dies funktioniert nicht unter Kitkat Führen Sie den folgenden Befehl aus, um mit der Aufnahme des Bildschirms Ihres Geräts zu beginnen:

`adb shell screenrecord /sdcard/example.mp4` , Mit diesem Befehl wird der Bildschirm Ihres Geräts mit den Standardeinstellungen `/sdcard/example.mp4` und das resultierende Video in einer Datei unter `/sdcard/example.mp4` auf Ihrem Gerät gespeichert.

Wenn Sie mit der Aufnahme fertig sind, drücken Sie im Eingabeaufforderungsfenster Strg + C (z in Linux), um die Bildschirmaufnahme zu stoppen. Sie können die Bildschirmaufzeichnungsdatei

dann an dem von Ihnen angegebenen Ort finden. Beachten Sie, dass die Bildschirmaufnahme im internen Speicher Ihres Geräts gespeichert wird, nicht auf Ihrem Computer.

In der Standardeinstellung verwenden Sie die Standard-Bildschirmauflösung Ihres Geräts, kodieren Sie das Video mit einer Bitrate von 4 MBit / s und stellen Sie die maximale Bildschirmaufnahmezeit auf 180 Sekunden ein. Führen Sie den folgenden Befehl aus, um weitere Informationen zu den Befehlszeilenoptionen zu erhalten:

`adb shell screenrecord –help` , Dies funktioniert ohne das Gerät zu `adb shell screenrecord –help` .
Hoffe das hilft.

Anwendungsdaten löschen

Mit `adb` können Sie die Benutzerdaten einer bestimmten App `adb` :

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

Dies ist das gleiche wie beim Durchsuchen der Einstellungen auf dem Telefon. Wählen Sie die App aus und drücken Sie die Taste zum Löschen der Daten.

- `pm` ruft den Paketmanager auf dem Gerät auf
- `clear` löscht alle Daten, die einem Paket zugeordnet sind

Sendung senden

Es ist möglich, `BroadcastReceiver` mit `adb` an `BroadcastReceiver` zu senden.

In diesem Beispiel senden wir Broadcast mit der Aktion `com.test.app.ACTION` und der Zeichenfolge extra im Bundle `'foo'='bar'` :

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

Sie können jeden anderen unterstützten Typ zum Bündeln verwenden, nicht nur Strings:

--ez - boolean
--ei - ganze Zahl
--el - lang
--ef - Schwimmer
--eu - uri
--eia - int array (getrennt durch ',')
--ela - long array (getrennt durch ',')
--efa - float array (getrennt durch ',')
--esa - String-Array (getrennt durch ',')

Um Intents an bestimmte Pakete / Klassen zu senden, kann der Parameter `–n` oder `–p` verwendet werden.
Versand ins Paket:

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

Senden an eine bestimmte Komponente (`SomeReceiver` Klasse im `com.test.app package`):

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

Nützliche Beispiele:

- Senden einer "Boot Complete" -Sendung
- Senden einer "Zeit geändert" -Sendung nach dem Einstellen der Zeit mit dem Befehl adb

Installieren Sie eine Anwendung und führen Sie sie aus

Verwenden Sie den folgenden Befehl, **um eine APK-Datei zu installieren** :

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

oder wenn die App vorhanden ist und wir neu installieren möchten

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

Um eine Anwendung zu deinstallieren , müssen wir ihr Paket angeben

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

Verwenden Sie den folgenden Befehl, um eine App mit einem angegebenen Paketnamen (oder einer bestimmten Aktivität in einer App) zu starten:

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

Zum Beispiel, um Waze zu starten:

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

Backup

Sie können den Befehl `adb backup` um Ihr Gerät zu sichern.

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

`-f <filename>` Angabe Dateiname **Standard:** *erstellt backup.ab im aktuellen Verzeichnis*

`-apk|noapk` Aktiviert / deaktiviert die Sicherung von .apks. **Standardwert:** *-noapk*

`-obb|noobb` Aktivieren / Deaktivieren der Sicherung zusätzlicher Dateien **Standardwert:** *-noobb*

`-shared|noshared`

des gemeinsam genutzten Speicher- / SD-Karteninhalts des `-shared|noshared` Backup-Geräts **:** *-noshared*

`-all` Sicherung aller installierten Anwendungen

`-system|nosystem` include Systemanwendungen **Standard:** *-system*

`<packages>` eine Liste der zu sichernden Pakete (z. B. com.example.android.myapp) (nicht erforderlich, wenn `-all` angegeben ist)

Verwenden Sie für eine vollständige Gerätesicherung, einschließlich alles

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

> **Hinweis: Das** Durchführen einer vollständigen Sicherung kann lange dauern.

Um eine Sicherung wiederherzustellen, verwenden Sie

```
adb restore backup.ab
```

Installieren Sie ADB auf einem Linux-System

So installieren Sie die Android Debugging Bridge (ADB) auf einem Linux-System, wobei das Terminal die Repositorys Ihrer Distribution verwendet.

Installation auf Ubuntu / Debian-System über apt:

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

Installation auf Fedora / CentOS-System über yum:

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

Installation auf dem Gentoo-System mit portage:

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

Installieren Sie das OpenSUSE-System mit Zypper:

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

Installation in Arch-System mit Pacman:

```
sudo pacman -Syyu
```

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

Listen Sie alle Berechtigungen auf, für die Benutzer unter Android 6.0 eine Laufzeitgewährung benötigen

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

Anzeigen der internen Daten einer App (Daten / Daten /) auf einem Gerät

`AndroidManifest.xml` sicher, dass Ihre App in `AndroidManifest.xml` gesichert werden `AndroidManifest.xml` , dh `android:allowBackup` ist nicht `false` .

Sicherungsbefehl:

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

Erstellen Sie einen tar mit dem Befehl dd:

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

Extrahieren Sie den Teer:

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

Sie können dann den extrahierten Inhalt anzeigen.

Aktivitätsstapel anzeigen

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

Sehr nützlich in Verbindung mit dem Befehl `watch` unix:

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

Anzeigen und Abrufen von Cache-Dateien einer App

Sie können diesen Befehl verwenden, um die Dateien für Ihre eigene debuggable apk aufzulisten:

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

Und dieses Skript zum Abrufen aus dem Cache-Speicher, kopieren Sie den Inhalt zuerst auf die SD-Karte, ziehen Sie ihn ab und entfernen Sie ihn am Ende:

```
#!/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'"
```

Dann können Sie eine Datei wie folgt aus dem Cache ziehen:

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

Datenbankdatei über ADB abrufen

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

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

Kapitel 6: ADB Shell

Einführung

`adb shell` öffnet eine Linux-Shell in einem Zielgerät oder Emulator. Es ist die leistungsfähigste und vielseitigste Möglichkeit, ein Android-Gerät über `adb` zu steuern.

Dieses Thema wurde von ADB (Android Debug Bridge) getrennt, da die Anzahl der Beispiele erreicht wurde, von denen viele den Befehl `adb shell` .

Syntax

- Adb-Shell [-e Escape] [-n] [-Tt] [-x] [Befehl]

Parameter

Parameter	Einzelheiten
-e	Fluchtzeichen oder "none" wählen; Standardeinstellung "~"
-n	lese nicht von stdin
-T	Deaktivieren Sie die PTY-Zuordnung
-t	PTY-Zuordnung erzwingen
-x	Deaktivieren Sie Remote-Exit-Codes und stdout / stderr-Trennung

Examples

Senden Sie Text, drücken Sie die Taste und berühren Sie Ereignisse über ADB an das Android-Gerät

Führen Sie den folgenden Befehl aus, um den Text in eine Ansicht mit einem Fokus einzufügen (wenn er Texteingabe unterstützt).

6,0

Senden Sie Text auf SDK 23+

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

Wenn Sie bereits über `adb` mit Ihrem Gerät verbunden sind:

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

6,0

Senden Sie Text vor dem SDK 23

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

Leerzeichen werden nicht als Eingabe akzeptiert, sondern durch% s ersetzt.

Ereignisse senden

So simulieren Sie das Drücken der Hardwaretaste

```
adb shell input keyevent 26
```

oder alternativ

```
adb shell input keyevent POWER
```

Selbst wenn Sie keinen Hardwareschlüssel haben, können Sie immer noch ein `keyevent` , um die entsprechende Aktion auszuführen

```
adb shell input keyevent CAMERA
```

Berührungsereignis als Eingabe senden

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

Swipe-Event als Eingabe senden

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

* DAUER ist optional, Standard = 300ms. Quelle

Erhalten Sie X- und Y-Punkte, indem Sie die Zeigerposition in der Entwickleroption aktivieren.

ADB-Beispielshellskript

> Um ein Skript in Ubuntu auszuführen, klicken Sie auf Create script.sh, klicken Sie mit der rechten Maustaste auf die Datei, fügen Sie die Lese- / Schreibberechtigung hinzu und aktivieren Sie das Kontrollkästchen **Ausführen als Programm zulassen** .
>
> Öffnen Sie den Terminalemulator und führen Sie den Befehl ./script.sh aus

Script.sh

```
 for (( c=1; c<=5; c++ ))
    do
```

```
    adb shell input tap X Y
    echo "Clicked $c times"
    sleep 5s
done
```

Für eine umfassende Liste der Veranstaltungsnummern

- Auswahlliste mehrerer interessanter Ereignisse ADB Shell Input Events
- Referenzdokumentation https://developer.android.com/reference/android/view/KeyEvent.html#KEYCODE_POWER .

Pakete auflisten

Druckt alle Pakete, optional nur diejenigen, deren Paketname den Text in <FILTER> enthält.

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

All <FILTER>

adb shell pm list packages
```

Attribute:

`-f` , um die zugehörige Datei `-f` .

`-i` Informationen zu den Paketen finden Sie im Installationsprogramm.

`-u` um auch deinstallierte Pakete aufzunehmen.

`-u` auch deinstallierte Pakete.

Attribute, die filtern:

`-d` für deaktivierte Pakete.

`-e` für aktivierte Pakete.

`-s` für Systempakete.

`-3` für Pakete von Drittanbietern.

`--user <USER_ID>` für einen bestimmten Benutzerbereich zum Abfragen.

Erteilen und Widerrufen von API 23+ -Berechtigungen

Ein Einzeiler, mit dem verwundbare Berechtigungen erteilt oder widerrufen werden.

- **Gewährung**

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

- **widerrufen**

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

- **Alle Laufzeitberechtigungen bei der Installation erteilen (-g)**

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

Anwendungsdaten drucken

Dieser Befehl druckt alle relevanten Anwendungsdaten:

- Versionscode
- Versionsname
- erteilte Berechtigungen (Android API 23+)
- usw..

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

Aufzeichnen der Anzeige

4.4

Aufzeichnen der Anzeige von Geräten, auf denen Android 4.4 (API Level 19) und höher ausgeführt wird:

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

(drücken Sie Strg-C, um die Aufnahme zu stoppen)

Laden Sie die Datei vom Gerät herunter:

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

> Hinweis: Stoppen Sie die Bildschirmaufnahme, indem Sie die Tastenkombination Strg-C drücken. Andernfalls wird die Aufnahme automatisch nach drei Minuten oder nach der `--time-limit` .

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

Legt die Videogröße fest: 1280x720. Der Standardwert ist die native Anzeigeauflösung des Geräts (falls unterstützt), 1280x720, falls nicht. Um optimale Ergebnisse zu erzielen, verwenden Sie eine vom AVC-Encoder (Advanced Video Coding) Ihres Geräts unterstützte Größe.

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

Legt die Video-Bitrate für das Video in Megabit pro Sekunde fest. Der Standardwert ist 4 MBit / s. Sie können die Bitrate erhöhen, um die Videoqualität zu verbessern. Dies führt jedoch zu größeren Filmdateien. Im folgenden Beispiel wird die Aufzeichnungsbitrate auf 5 MBit / s festgelegt:

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

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

Legt die maximale Aufnahmezeit in Sekunden fest. Der voreingestellte und maximale Wert beträgt 180 (3 Minuten).

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

Dreht die Ausgabe um 90 Grad. Diese Funktion ist experimentell.

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

Zeigt Protokollinformationen auf dem Befehlszeilenbildschirm an. Wenn Sie diese Option nicht festlegen, zeigt das Dienstprogramm während der Ausführung keine Informationen an.

> Hinweis: Dies funktioniert möglicherweise auf einigen Geräten nicht.

4.4

Der Bildschirmaufzeichnungsbefehl ist nicht mit Android-Versionen vor 4.4 kompatibel

> Der screenrecord-Befehl ist ein Shell-Dienstprogramm zum Aufzeichnen der Anzeige von Geräten, auf denen Android 4.4 (API Level 19) und höher ausgeführt wird. Das Dienstprogramm zeichnet die Bildschirmaktivität in einer MPEG-4-Datei auf.

Ändern der Dateiberechtigungen mit dem Befehl chmod

Beachten Sie, dass Ihr Gerät verwurzelt sein muss, um die Dateiübertragung zu ändern. *`su` binary wird nicht mit werkseitig gelieferten Geräten* ***geliefert*** *!*

Konvention:

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

Numerische Berechtigung, erstellt aus Benutzer-, Gruppen- und Weltabschnitten.

Wenn Sie beispielsweise die Datei so ändern möchten, dass sie von allen lesbar, schreibbar und ausführbar ist, ist dies Ihr Befehl:

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

Oder

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

wenn Sie beabsichtigen, Berechtigungen dafür zu verweigern.

1. Stelle - Gibt die Benutzererlaubnis an, **2. Stelle** - Gibt die Gruppenerlaubnis an, **3. Stelle** - Gibt die Weltberechtigung (andere) an.

Zugriffsberechtigungen:

```
--- :   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)
```

Datum / Uhrzeit über adb einstellen

6,0

Das `MMDDhhmm[[CC]YY][.ss]` SET-Format ist `MMDDhhmm[[CC]YY][.ss]` , das ist (jeweils 2 Ziffern).

Um beispielsweise den 17. Juli um 10:10 Uhr einzustellen, ohne das aktuelle Jahr zu ändern, geben Sie Folgendes ein:

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

Tipp 1: Die Datumsänderung wird nicht sofort angezeigt, und eine merkliche Änderung wird erst nach der nächsten Uhr der Systemuhr vorgenommen.
Sie können eine Aktualisierung erzwingen, indem `TIME_SET` Ihrem Aufruf eine `TIME_SET` Intent-Broadcast `TIME_SET` :

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

Tipp 2: So synchronisieren Sie die Uhr von Android mit Ihrem lokalen Computer:

Linux:

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

Windows (PowerShell):

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

Beide Tipps zusammen:

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

6,0

Das Standard-SET-Format ist 'JJJJMMTT.HHmmss'.

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

Tipp: So synchronisieren Sie die Uhr von Android mit Ihrem lokalen (Linux-basierten) Computer:

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

Öffnen Sie die Entwickleroptionen

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

Navigiert mit Ihrem Gerät / Emulator zum Abschnitt " `Developer Options` ".

Generieren einer "Boot Complete" -Übertragung

Dies ist relevant für Apps, die einen `BootListener` implementieren. Testen Sie Ihre App, indem Sie Ihre App beenden und testen Sie dann mit:

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

(Ersetzen Sie `your.package/your.app.BootListener` durch die richtigen Werte).

Anzeigen externer / sekundärer Speicherinhalte

Inhalt anzeigen:

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

Pfad anzeigen:

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

Beenden Sie einen Prozess in einem Android-Gerät

Manchmal läuft das Logcat von Android unendlich oft mit Fehlern, die auf einen Prozess zurückzuführen sind, der nicht Ihnen gehört, die Batterie entleert oder nur das Debuggen Ihres Codes erschwert.

Um das Problem ohne Neustart des Geräts zu beheben, können Sie den Prozess, der das Problem verursacht, lokalisieren und beenden.

Von Logcat

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

Beachten Sie die Prozessnummer: 1550

Jetzt können wir eine Shell öffnen und den Prozess beenden. Beachten Sie, dass wir den `root` Prozess nicht beenden können.

```
adb shell
```

In der Shell können wir mehr über den Prozess überprüfen

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

und töte es, wenn wir wollen:

```
kill -9 1550
```

ADB Shell online lesen: https://riptutorial.com/de/android/topic/9408/adb-shell

Kapitel 7: AdMob

Syntax

- 'com.google.firebase: firebase-ads: 10.2.1' kompilieren // HINWEIS: AUF NEUESTE VERSION BEI VERFÜGBARER SETZEN
- `<uses-permission android:name="android.permission.INTERNET" />` Erforderlich, um die Anzeige abzurufen
- AdRequest AdRequest = new AdRequest.Builder (). Build (); // Bannerwerbung
- AdView mAdView = (AdView) findViewById (R.id.adView); // Banneranzeige
- mAdView.loadAd (adRequest); // Banneranzeige

Parameter

Param	Einzelheiten
ads: adUnitId = "@ string / main_screen_ad"	Die ID Ihrer Anzeige Erhalten Sie Ihre ID von der Admob-Site. *"Zwar ist das Speichern Ihrer ID-Werte für Anzeigenblöcke in einer Ressourcendatei nicht zwingend erforderlich. Wenn Ihre App wächst und Ihre Anzeigenveröffentlichung erforderlich ist, müssen Sie möglicherweise die ID-Werte ändern. Wenn Sie sie in einer Ressource behalten Datei, Sie müssen Ihren Code niemals durchsuchen und danach suchen. "* [1]

Bemerkungen

- Erfordert ein gültiges Admob-Konto
- Lesen Sie die Admob-Richtlinie . Stellen Sie sicher, dass Sie nichts tun, was Ihren Admob-Account suspendieren kann

Examples

Umsetzung

Hinweis: Für dieses Beispiel sind ein gültiges Admob-Konto und ein gültiger Admob-Anzeigencode erforderlich.

Build.gradle auf App-Ebene

Wechseln Sie zur neuesten Version, falls vorhanden:

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

Manifest

Für den Zugriff auf die Daten ist eine Internet-Genehmigung erforderlich. Beachten Sie, dass diese Berechtigung nicht angefordert werden muss (mit API 23+), da es sich um eine normale Berechtigung handelt, die nicht gefährlich ist:

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

XML

Das folgende XML-Beispiel zeigt eine Banneranzeige:

```
<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" />
```

Den Code anderer Typen finden Sie in der Google AdMob-Hilfe .

Java

Der folgende Code ist für die Integration von Werbebanner. Beachten Sie, dass andere Anzeigentypen möglicherweise eine andere Integration erfordern:

```
// 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);
```

Fügen Sie die `AdView` Lebenszyklusmethoden in den Methoden `onResume()` , `onPause()` und `onDestroy()` Ihrer Aktivität hinzu:

```
@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();
}
```

AdMob online lesen: https://riptutorial.com/de/android/topic/5334/admob

Kapitel 8: AIDL

Einführung

AIDL ist die Sprache der Android-Benutzeroberfläche.

Was? Warum? Wie ?

Was? Es ist eine beschränkte Dienstleistung. Dieser AIDL-Dienst ist so lange aktiv, bis mindestens ein Client vorhanden ist. Es basiert auf dem Marshalling- und Unmarshaling-Konzept.

Warum? Remote-Anwendungen können auf Ihren Dienst + Multi-Threading zugreifen (Anforderung für Remote-Anwendung).

Wie? Erstellen Sie die .aidl-Datei. Implementieren Sie die Schnittstelle. Machen Sie die Schnittstelle für die Clients verfügbar

Examples

AIDL-Service

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;
        }
    };

}
```

Service-Verbindung

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

AIDL online lesen: https://riptutorial.com/de/android/topic/9504/aidl

Kapitel 9: Aktivität

Einführung

Eine Aktivität steht für einen einzelnen Bildschirm mit einer Benutzeroberfläche **(UI)** . Eine Android-App kann mehrere Aktivitäten haben, z. B. Eine E-Mail-App kann eine Aktivität zum Auflisten aller E-Mails, eine weitere Aktivität zum Anzeigen von E-Mail-Inhalten und eine weitere Aktivität zum Erstellen neuer E-Mails enthalten. Alle Aktivitäten in einer App arbeiten zusammen, um eine perfekte Benutzererfahrung zu schaffen.

Syntax

- void onCreate (Bundle savedInstanceState) // Wird aufgerufen, wenn die Aktivität beginnt.
- void onPostCreate (Bundle savedInstanceState) // Wird aufgerufen, wenn der Aktivitätsstart abgeschlossen ist (nachdem onStart () und onRestoreInstanceState (Bundle) aufgerufen wurden).
- void onStart () // Wird nach onCreate (Bundle) aufgerufen - oder nach onRestart (), wenn die Aktivität angehalten wurde, dem Benutzer jedoch jetzt wieder angezeigt wird.
- void onResume () // Wird nach onRestoreInstanceState (Bundle), onRestart () oder onPause () aufgerufen, damit Ihre Aktivität mit dem Benutzer interagieren kann.
- void onPostResume () // Wird aufgerufen, wenn die Wiederaufnahme der Aktivität abgeschlossen ist (nachdem onResume () aufgerufen wurde).
- void onRestart () // Wird nach onStop () aufgerufen, wenn die aktuelle Aktivität dem Benutzer erneut angezeigt wird (der Benutzer hat wieder zu ihm navigiert).
- void onPause () // Wird als Teil des Aktivitätslebenszyklus aufgerufen, wenn eine Aktivität in den Hintergrund geht, aber (noch) nicht beendet wurde.
- void onStop () // Wird aufgerufen, wenn Sie für den Benutzer nicht mehr sichtbar sind.
- void onDestroy () // Führen Sie eine letzte Bereinigung durch, bevor eine Aktivität zerstört wird.
- void onNewIntent (Intent Intent) // Dies wird für Aktivitäten aufgerufen, bei denen launchMode in ihrem Paket auf "singleTop" gesetzt ist oder wenn ein Client beim Aufruf von startActivity (Intent) das Flag FLAG_ACTIVITY_SINGLE_TOP verwendet hat.
- void onSaveInstanceState (Bundle outState) // Wird aufgerufen, um den Instanzstatus von einer Aktivität abzurufen, bevor er beendet wird, damit der Status in onCreate (Bundle) oder onRestoreInstanceState (Bundle) wiederhergestellt werden kann (das von dieser Methode aufgefüllte Bundle wird an beide übergeben).

- void onRestoreInstanceState (Bundle savedInstanceState) // Diese Methode wird nach onStart () aufgerufen, wenn die Aktivität aus einem zuvor gespeicherten Status neu initialisiert wird, der hier in SavedInstanceState angegeben ist.

Parameter

Parameter	Einzelheiten
Absicht	Kann mit startActivity verwendet werden , um eine Aktivität zu starten
Bündeln	Eine Zuordnung von String-Schlüsseln zu verschiedenen Parcelable- Werten.
Kontext	Schnittstelle zu globalen Informationen zu einer Anwendungsumgebung.

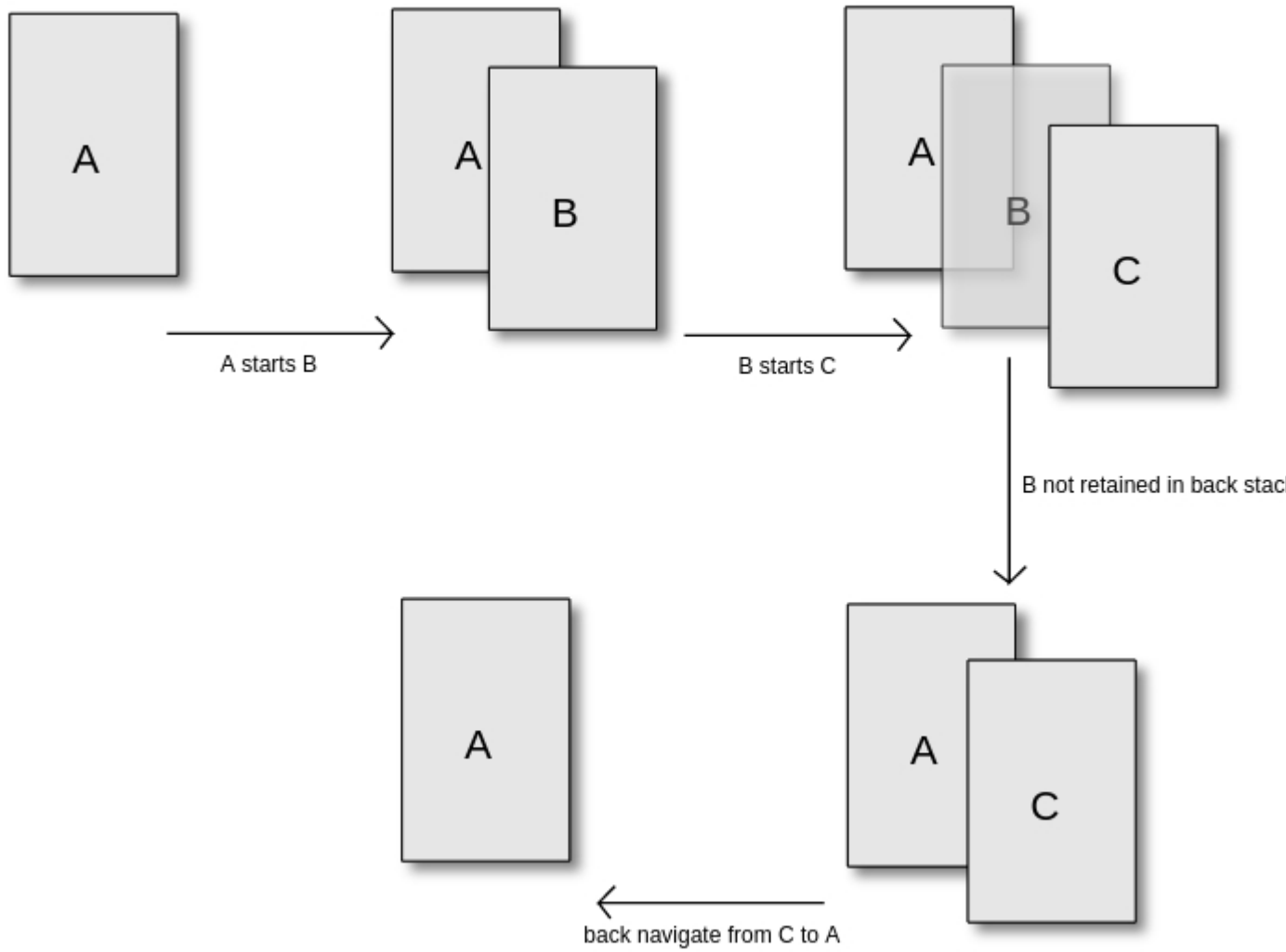
Bemerkungen

Eine Aktivität ist eine Anwendungskomponente, die einen Bildschirm bietet, auf dem Benutzer interagieren können, um etwas zu tun, z. B. das Telefon wählen, ein Foto aufnehmen, eine E-Mail senden oder eine Karte anzeigen. Jede Aktivität erhält ein Fenster, in dem sie ihre Benutzeroberfläche zeichnen kann. Das Fenster füllt normalerweise den Bildschirm, kann jedoch kleiner als der Bildschirm sein und über anderen Fenstern schweben.

Examples

Schließen Sie eine Aktivität aus dem Back-Stack-Verlauf aus

Es sei die Aktivität `B` , die geöffnet werden kann und weitere Aktivitäten starten kann. Der Benutzer sollte jedoch nicht darauf stoßen, wenn er in Aufgabenaktivitäten zurück navigiert.



Die einfachste Lösung besteht darin, das Attribut `noHistory` für dieses `<activity>` -Tag in `AndroidManifest.xml` auf `true` zu `AndroidManifest.xml` :

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

Dasselbe Verhalten ist auch vom Code aus möglich, wenn `B` vor dem Start der nächsten Aktivität `finish()` aufruft:

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

Das `noHistory` Flag wird `noHistory` mit "Splash Screen" oder Login-Aktivitäten verwendet.

Android Activity LifeCycle erklärt

Angenommen, eine Anwendung mit einer MainActivity, die die nächste Aktivität mit einem Klick

auf die Schaltfläche aufrufen kann.

```
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);
    } }
```

und

```
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");
    } }
```

Wenn die App zum ersten Mal erstellt wird
D / MainActivity: Aufruf vonCreate from MainActivity
D / MainActivity: Aufruf von MainActivity auf Start
D / MainActivity: Aufruf anAusgabe von MainActivity
werden genannt

Wenn der Bildschirm schläft
08: 11: 03.142 D / MainActivity: Aufruf onPause von MainActivity
08: 11: 03.192 D / MainActivity: Aufruf vonStopp von MainActivity
werden genannt. Und wieder, wenn es aufwacht
08: 11: 55.922 D / MainActivity: Aufruf vonRestart von MainActivity
08: 11: 55.962 D / MainActivity: Aufruf anStarten von MainActivity
08: 11: 55.962 D / MainActivity: Aufruf anAusgabe von MainActivity
werden genannt

Fall 1: Wenn die nächste Aktivität aus der ***Hauptaktivität*** aufgerufen wird
D / MainActivity: Nächste Aktivität aufrufen
D / MainActivity: Aufruf anPause von MainActivity

D / NextActivity: Aufruf vonAus der nächsten Aktivität erstellen
D / NextActivity: Aufruf vonStart from Next Activity
D / NextActivity: Aufrufen vonArbeit von nächster Aktivität
D / MainActivity: Aufruf von MainActivity auf onStop

Wenn Sie mit der Zurück-Taste zur nächsten Hauptaktivität zurückkehren
D / NextActivity: Aufruf onPause from Next Activity
D / MainActivity: Aufruf von Neustart von MainActivity
D / MainActivity: Aufruf von MainActivity auf Start
D / MainActivity: Aufruf anAusgabe von MainActivity
D / NextActivity: Aufruf von onStop from Next Activity
D / NextActivity: Aufruf von onDestroy aus Next Activity

Fall2: Wenn die Aktivität teilweise verdeckt ist (Wenn die Übersichtstaste gedrückt wird) oder Wenn die App in den Hintergrund wechselt und eine andere App dies völlig verdeckt
D / MainActivity: Aufruf anPause von MainActivity
D / MainActivity: Aufruf von MainActivity auf onStop
und wenn die App wieder im Vordergrund ist, um Benutzereingaben zu akzeptieren,
D / MainActivity: Aufruf von Neustart von MainActivity
D / MainActivity: Aufruf von MainActivity auf Start
D / MainActivity: Aufruf anAusgabe von MainActivity
werden genannt

Fall3: Wenn eine Aktivität aufgerufen wird, um eine implizite Absicht zu erfüllen, und der Benutzer eine Auswahl getroffen hat. Zum Beispiel, wenn die Share-Taste gedrückt wird und der Benutzer eine App aus der Liste der angezeigten Anwendungen auswählen muss
D / MainActivity: Aufruf anPause von MainActivity

Die Aktivität ist sichtbar, aber jetzt nicht aktiv. Wenn die Auswahl abgeschlossen ist und die App aktiv ist
D / MainActivity: Aufruf anAusgabe von MainActivity
wird genannt

Fall4:
Wenn die App im Hintergrund beendet wird (um Ressourcen für eine andere Vordergrund-App *freizugeben*), werden *onPause* (für Pre-Honeycomb-Gerät) oder *onStop* (für da Honeycomb-Gerät) das letzte Mal aufgerufen, bevor die App beendet wird.

onCreate und onDestroy werden bei jeder Ausführung der Anwendung einmalig als utmost bezeichnet. OnPause, onStop, onRestart, onStart, onResume werden jedoch oft während des Lebenszyklus aufgerufen.

Aktivitätsstartmodus

Der Startmodus definiert das Verhalten einer neuen oder vorhandenen Aktivität in der Aufgabe. Es gibt mögliche Startmodi:

- Standard

- singleTop
- singleTask
- Einzelinstanz

Es sollte im Android-Manifest im `<activity/>` -Element als `android:launchMode` Attribut definiert werden.

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

Standard:

Standardwert. Wenn dieser Modus festgelegt ist, wird für jede neue Absicht immer eine neue Aktivität erstellt. So ist es möglich, viele Aktivitäten des gleichen Typs zu erhalten. Neue Aktivität wird auf die Aufgabe gesetzt. Es gibt einige Unterschiede bei den verschiedenen Android-Versionen: Wenn die Aktivität von einer anderen Anwendung aus gestartet wird, wird sie bei Androiden <= 4.4 derselben Aufgabe als Starter-Anwendung zugeordnet, aber bei> = 5.0 wird eine neue Aufgabe erstellt.

SingleTop:

Dieser Modus entspricht fast dem `standard` . Es können viele Instanzen von singleTop-Aktivitäten erstellt werden. Der Unterschied besteht darin, dass `onNewIntent()` aufgerufen wird, anstatt eine neue Instanz zu erstellen, wenn auf dem aktuellen Stack bereits eine Aktivitätsinstanz vorhanden ist.

SingleTask:

Die Aktivität bei diesem Startmodus kann nur eine Instanz **im System haben** . Neue Aufgabe für Aktivität wird erstellt, wenn sie nicht existiert. Andernfalls wird die Task mit der Aktivität nach vorne `onNewIntent` und `onNewIntent` wird aufgerufen.

Einzelinstanz:

Dieser Modus ähnelt `singleTask` . Der Unterschied ist die Aufgabe, die eine Aktivität mit `singleInstance` kann nur diese Aktivität haben und nichts weiter. Wenn die `singleInstance` Aktivität eine andere Aktivität erstellt, wird eine neue Aufgabe erstellt, um diese Aktivität zu platzieren.

Präsentieren der Benutzeroberfläche mit setContentView

Die Aktivitätsklasse erstellt für Sie ein Fenster, in dem Sie Ihre Benutzeroberfläche mit

`setContentView` .

Es gibt drei `setContentView` Methoden:

- `setContentView(int layoutResID)` - `setContentView(int layoutResID)` den Aktivitätsinhalt einer `setContentView(int layoutResID)` .
- `setContentView(View view)` - Setzt den Aktivitätsinhalt auf eine explizite Ansicht.
- `setContentView(View view, ViewGroup.LayoutParams params)` - `setContentView(View view, ViewGroup.LayoutParams params)` den Aktivitätsinhalt auf eine explizite Ansicht mit den bereitgestellten Parametern fest.

Wenn `setContentView` aufgerufen wird, wird diese Ansicht direkt in die Ansichtshierarchie der Aktivität `setContentView` . Es kann selbst eine komplexe Ansichtshierarchie sein.

Beispiele

Inhalt aus Ressourcendatei setzen:

Fügen Sie die Ressourcendatei (in diesem Beispiel main.xml) mit der Ansichtshierarchie hinzu:

```
<?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>
```

Legen Sie es als Inhalt in Aktivität fest:

```
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);
    }
}
```

Setzen Sie den Inhalt auf eine explizite Ansicht:

```
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);
    }
}
```

Löschen Sie Ihren aktuellen Aktivitätsstapel und starten Sie eine neue Aktivität

Wenn Sie Ihren aktuellen Aktivitätsstapel löschen und eine neue Aktivität starten möchten (z. B. Abmeldung von der App und Starten einer Protokollaktivität), gibt es zwei Ansätze.

1. Ziel (API> = 16)

`finishAffinity()` von einer Aktivität aus `finishAffinity()`

2. Ziel (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();
```

Bewerbung beenden mit Ausnahmen ausschließen

Definieren Sie zunächst eine ExitActivity in der AndroidManifest.xml

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

Danach die ExitActivity-Klasse

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

Up Navigation für Aktivitäten

Die Navigation in Android erfolgt durch Hinzufügen von `android:parentActivityName=""` in Manifest.xml zum Activity-Tag. Mit diesem Tag informieren Sie das System grundsätzlich über die übergeordnete Aktivität einer Aktivität.

Wie wird es gemacht?

```
<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>
```

Wenn ich jetzt auf den Pfeil in der Symbolleiste von HomeActivity klicke, werde ich zur übergeordneten Aktivität zurückkehren.

Java-Code

Hier werde ich den entsprechenden Java-Code schreiben, der für diese Funktionalität erforderlich ist.

```
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.
}
}
```

Wenn Sie diesen Code ausführen, sehen Sie, wenn Sie die Zurück-Taste drücken, um zur Hauptaktivität zurückzukehren. Für ein besseres Verständnis von Up Navigation würde ich das Lesen von Dokumenten empfehlen

Sie können dieses Verhalten an Ihre Bedürfnisse anpassen, indem Sie es überschreiben

```
@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);
}
```

Einfacher Hack

Dies ist ein einfacher Hack, der meistens verwendet wird, um zur übergeordneten Aktivität zu navigieren, wenn das übergeordnete Element im Hintergrund ist. Durch Aufruf von `onBackPressed()` wenn id gleich `android.R.id.home`

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

Aktivität online lesen: https://riptutorial.com/de/android/topic/1481/aktivitat

Kapitel 10: Aktivitätserkennung

Einführung

Aktivitätserkennung ist die Erkennung der körperlichen Aktivität eines Benutzers, um bestimmte Aktionen auf dem Gerät auszuführen, z. B. das Erfassen von Punkten, wenn ein Laufwerk erkannt wird, das WLAN ausschalten, wenn das Telefon noch aktiv ist, oder die Ruflautstärke auf Maximum stellen, wenn der Benutzer aktiv ist Gehen.

Examples

Google Play-AktivitätRecognitionAPI

Dies ist nur ein einfaches Beispiel für die Verwendung der ActivityRecognitionApi von GooglePlay Service. Obwohl dies eine großartige Bibliothek ist, funktioniert sie nicht auf Geräten, auf denen keine Google Play-Dienste installiert sind.

Dokumente für die ActivityRecognition-API

Manifest

```
<!-- 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) {

    }
}
```

Aktivitätsempfänger

```
public class ActivityReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

LocalBroadcastManager.getInstance(context).sendBroadcast(intent.setAction("activity"));
    }
}
```

PathSense-Aktivitätserkennung

Die PathSense- Aktivitätserkennung ist eine weitere gute Bibliothek für Geräte, die über keine Google Play-Dienste verfügen, da sie ein eigenes Aktivitätserkennungsmodell erstellt haben. Entwickler müssen sich jedoch unter http://developer.pathsense.com registrieren, um einen API-Schlüssel und eine Client-ID zu erhalten .

Manifest

```
<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);
    }
}
```

Aktivitätserkennung online lesen: https://riptutorial.com/de/android/topic/9831/aktivitatserkennung

Kapitel 11: AlarmManager

Examples

Führen Sie eine Absicht später aus

1. Erstellen Sie einen Empfänger. Diese Klasse erhält die Absicht und behandelt sie wie gewünscht.

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

2. Weisen Sie AlarmManager eine Absicht zu. In diesem Beispiel wird die Absicht ausgelöst, dass sie nach 1 Minute an AlarmReceiver gesendet wird.

```
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 );
```

So löschen Sie einen Alarm

Wenn Sie einen Alarm abbrechen möchten und keinen Verweis auf das ursprüngliche PendingIntent haben, das zum Einstellen des Alarms verwendet wurde, müssen Sie ein PendingIntent genau so erstellen, wie es bei seiner ursprünglichen Erstellung vorlag.

Eine Absicht wird vom AlarmManager als gleich betrachtet :

> wenn ihre Aktion, Daten, Typ, Klasse und Kategorien gleich sind. Dies vergleicht keine zusätzlichen Daten, die in den Absichten enthalten sind.

Normalerweise ist der Anforderungscode für jeden Alarm als eine Konstante definiert:

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

Also für einen einfachen Alarm wie folgt einrichten:

```
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);
```

So erstellen Sie eine neue PendingIntent-Referenz, mit der Sie den Alarm mit einer neuen AlarmManager-Referenz abbrechen können:

```
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);
}
```

Erstellen Sie genaue Alarme für alle Android-Versionen

Da mit der Zeit immer mehr Batterieoptimierungen in das Android-System `AlarmManager` haben sich auch die Methoden des `AlarmManager` erheblich geändert (um ein milderes Timing zu ermöglichen). Für einige Anwendungen ist es jedoch erforderlich, auf allen Android-Versionen so genau wie möglich zu sein. Der folgende Helper verwendet die genaueste Methode, die auf allen Plattformen verfügbar ist, um ein `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);
    }
}
```

Der API23 + Doze-Modus greift in AlarmManager ein

Android 6 (API23) hat den Doze-Modus eingeführt, der den AlarmManager stört. Es verwendet bestimmte Wartungsfenster, um Alarme zu behandeln. Wenn Sie also `setExactAndAllowWhileIdle()` , können Sie nicht sicherstellen, dass Ihr Alarm zum gewünschten Zeitpunkt ausgelöst wird.

Sie können dieses Verhalten für Ihre App mithilfe der Einstellungen Ihres Telefons deaktivieren (`Settings/General/Battery & power saving/Battery usage/Ignore optimizations` oder ähnliches).

In Ihrer App können Sie diese Einstellung überprüfen ...

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

... und zeigen ggf. den jeweiligen Einstellungsdialog:

```
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);
```

AlarmManager online lesen: https://riptutorial.com/de/android/topic/1361/alarmmanager

Kapitel 12: Alert-Dialoge verbessern

Einführung

In diesem Thema wird ein `AlertDialog` mit zusätzlichen Funktionen erweitert.

Examples

Benachrichtigungsdialog mit einem anklickbaren Link

Um einen Alarmdialog anzuzeigen, der einen Link enthält, der durch Klicken geöffnet werden kann, können Sie folgenden Code verwenden:

```
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
```

Alert-Dialoge verbessern online lesen: https://riptutorial.com/de/android/topic/10163/alert-dialoge-verbessern

Kapitel 13: Android Authenticator

Examples

Basic Account Authenticator-Dienst

Das Android Account Authenticator-System kann dazu verwendet werden, den Client bei einem Remote-Server zu authentifizieren. Drei Informationen sind erforderlich:

- Ein Dienst, ausgelöst durch den `android.accounts.AccountAuthenticator` . Die `onBind` Methode sollte eine Unterklasse von `AbstractAccountAuthenticator` .
- Eine Aktivität, um den Benutzer zur Eingabe von Anmeldeinformationen aufzufordern (Login-Aktivität)
- Eine XML-Ressourcendatei zur Beschreibung des Kontos

1. Der Dienst:

Fügen Sie die folgenden Berechtigungen in Ihre AndroidManifest.xml ein:

```
<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" />
```

Deklarieren Sie den Dienst in der Manifestdatei:

```
<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>
```

Beachten Sie, dass der `android.accounts.AccountAuthenticator` im `intent-filter` Tag enthalten ist. Die XML-Ressource (hier als `authenticator`) ist im `meta-data` Tag angegeben.

Die Serviceklasse:

```
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. Die XML-Ressource:

```
<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" />
```

Weisen Sie dem `android:label` keine Zeichenfolge zu `android:label` fehlende Zeichen oder weisen Sie ihnen fehlende Zeichen zu. Es wird ohne Warnung abstürzen.

3. Erweitern Sie die AbstractAccountAuthenticator-Klasse:

```
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;
    }
}
```

Die `addAccount()` -Methode in der `AbstractAccountAuthenticator` Klasse ist wichtig, da diese Methode aufgerufen wird, wenn Sie unter "Einstellungen" ein Konto im Bildschirm "Konto hinzufügen" hinzufügen. `AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE` ist wichtig, da es das AccountAuthenticatorResponse-Objekt enthält, das erforderlich ist, um die Kontoschlüssel bei erfolgreicher Benutzerverifizierung zurückzugeben.

Android Authenticator online lesen: https://riptutorial.com/de/android/topic/6759/android-authenticator

Kapitel 14: Android Java Native Interface (JNI)

Einführung

JNI (Java Native Interface) ist ein leistungsstarkes Tool, mit dem Android-Entwickler das NDK und den nativen C ++ - Code in ihren Anwendungen verwenden können. In diesem Thema wird die Verwendung der Java-C ++ - Schnittstelle beschrieben.

Examples

Aufrufen von Funktionen in einer systemeigenen Bibliothek über die JNI-Schnittstelle

Mit der Java Native Interface (JNI) können Sie native Funktionen aus Java-Code aufrufen und umgekehrt. In diesem Beispiel wird gezeigt, wie eine native Funktion über JNI geladen und aufgerufen wird. Sie greift nicht auf Java-Methoden und -Felder aus nativem Code über JNI-Funktionen zu .

Angenommen, Sie verfügen über eine native Bibliothek namens `libjniexample.so` im Ordner `project/libs/<architecture>` und möchten eine Funktion aus der `JNITest` Java-Klasse im Paket `com.example.jniexample` .

Deklarieren Sie die Funktion in der JNITest-Klasse folgendermaßen:

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

Definieren Sie die Funktion in Ihrem eigenen Code folgendermaßen:

```
#include <jni.h>

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

Das Argument `pEnv` ist ein Zeiger auf die JNI-Umgebung, die Sie an JNI-Funktionen übergeben können, um auf Methoden und Felder von Java-Objekten und -Klassen zuzugreifen. Der `thiz` Zeiger ist eine `jobject` auf das Java-Objekt, für das die native Methode aufgerufen wurde (oder die Klasse, wenn es sich um eine statische Methode handelt).

Laden Sie die Bibliothek in Ihrem Java-Code in `JNITest` :

```
static{
```

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

Beachten Sie die `lib` am Anfang und die `.so` am Ende des Dateinamens werden weggelassen.

Rufen Sie die native Funktion von Java aus folgendermaßen auf:

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

So rufen Sie eine Java-Methode aus nativem Code auf

Mit der Java Native Interface (JNI) können Sie Java-Funktionen aus nativem Code aufrufen. Hier ist ein einfaches Beispiel, wie es geht:

Java-Code:

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

Nativen Code:

```
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-Methodensignatur für Java-Typ:

JNI-Signatur	Java-Typ
Z	boolean
B	Byte
C	verkohlen
S	kurz

JNI-Signatur	Java-Typ
ich	int
J	lange
F	schweben
D	doppelt
L vollqualifizierte Klasse;	Vollqualifizierte Klasse
[Art	Art[]

Für unser Beispiel haben wir (Z) I verwendet - was bedeutet, dass die Funktion einen Boolean erhält und ein Int zurückgibt.

Dienstprogrammmethode in der JNI-Schicht

Diese Methode hilft beim Abrufen der Java-Zeichenfolge aus der C ++ - Zeichenfolge.

```
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;
    }
```

Mit dieser Methode können Sie jbyteArray in char konvertieren

```
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;
}
```

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

Kapitel 15: Android NDK

Examples

Native ausführbare Dateien für Android erstellen

Projekt / jni / main.c

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

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

Projekt / jni / Android.mk

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

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

Projekt / jni / Anwendung.mk

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

Wenn Sie Geräte unterstützen möchten, auf denen Android-Versionen niedriger als 5.0 (API 21) ausgeführt werden, müssen Sie Ihre Binärdatei mit `APP_PLATFORM` auf eine ältere API, z. B. `android-8` , kompilieren. Dies ist eine Folge von Android 5.0, bei dem *Position Independent Binaries* (PIE) erzwungen wird, während ältere Geräte PIEs nicht unbedingt unterstützen. Daher müssen Sie je nach Geräteversion entweder die PIE oder die Nicht-PIE verwenden. Wenn Sie die Binärdatei in Ihrer Android-Anwendung verwenden möchten, müssen Sie die API-Ebene überprüfen und die korrekte Binärdatei extrahieren.

`APP_ABI` kann auf bestimmte Plattformen wie `armeabi` , um die Binärdatei nur für diese Architekturen zu erstellen.

Im schlimmsten Fall haben Sie für jede Architektur eine PIE- und eine Nicht-PIE-Binärdatei (etwa 14 verschiedene Binärdateien mit ndk-r10e).

Um die ausführbare Datei zu erstellen:

```
cd project
ndk-build
```

Sie finden die Binärdateien unter `project/libs/<architecture>/hello_world` . Sie können sie über `ADB` (`push` und `chmod` mit ausführbarer Berechtigung) oder von Ihrer Anwendung (Extrahieren und `chmod` mit ausführbarer Berechtigung) verwenden.

Um die Architektur der CPU zu ermitteln, `ro.product.cpu.abi` die Build-Eigenschaft `ro.product.cpu.abi` für die primäre Architektur oder `ro.product.cpu.abilist` (auf neueren Geräten) ab, um eine vollständige Liste der unterstützten Architekturen zu erhalten. Sie können dies mit der `android.os.Build` Klasse in Ihrer Anwendung oder mit `getprop <name>` über ADB `getprop <name>` .

So reinigen Sie den Build

Wenn Sie den Build reinigen müssen:

```
ndk-build clean
```

So verwenden Sie ein anderes Makefile als Android.mk

ndk-build NDK_PROJECT_PATH = PROJECT_PATH APP_BUILD_SCRIPT = MeinAndroid.mk

Wie melde ich mich an?

`Android.mk` Sie zunächst sicher, dass Sie eine Verbindung zur Protokollierungsbibliothek in Ihrer `Android.mk` Datei herstellen:

```
LOCAL_LDLIBS := -llog
```

Verwenden Sie dann einen der folgenden `__android_log_print()` Aufrufe:

```
#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)
```

Oder verwenden Sie diese auf bequemere Weise, indem Sie entsprechende Makros definieren:

```
#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__)
```

Beispiel :

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

Android NDK online lesen: https://riptutorial.com/de/android/topic/492/android-ndk

Kapitel 16: Android Paypal Gateway-Integration

Bemerkungen

Paypal stellt uns eine eigene Bibliothek zur Verfügung, so dass es jetzt sehr sicher und einfach in unserer Anwendung implementiert werden kann. Nachfolgend finden Sie den wichtigen Schritt.

Examples

Richten Sie paypal in Ihrem Android-Code ein

1) Gehen Sie zunächst die Paypal Developer-Website durch und erstellen Sie eine Anwendung.

2) Öffnen Sie nun Ihre Manifestdatei und erteilen Sie die folgenden Berechtigungen

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

3) Und einige erforderliche Aktivitäten und Dienstleistungen

```
 <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) Öffnen Sie Ihre Aktivitätsklasse und legen Sie die Konfiguration für Ihre Anwendung fest.

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

5) Legen Sie nun die Client-ID aus dem Paypal-Entwicklerkonto fest. 6) Innerhalb der onCreate-Methode rufen Sie den Paypal-Service auf. Intent Intent = New Intent (diese PayPalService.class); intent.putExtra (PayPalService.EXTRA_PAYPAL_CONFIGURATION, config); startService (Absicht);

7) Nun können Sie eine Zahlung tätigen, indem Sie einfach die

```
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) Und schließlich von der onActivityResult erhalten Sie die Zahlungsantwort-

```
 @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.");
                   }
               }


           }
```

Android Paypal Gateway-Integration online lesen:
https://riptutorial.com/de/android/topic/5895/android-paypal-gateway-integration

Kapitel 17: Android Places-API

Examples

Verwendungsbeispiel für Picker

Place Picker ist ein wirklich einfaches Benutzeroberflächen-Widget, das von Places API bereitgestellt wird. Es bietet eine integrierte Karte, den aktuellen Standort, Orte in der Nähe, Suchfunktionen und eine automatische Vervollständigung.

Dies ist ein Beispiel für die Verwendung des Place Picker UI-Widgets.

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

```
}
```

Aktuelle Orte mithilfe der Places-API abrufen

Sie können den aktuellen Standort und die lokalen Orte des Nutzers über das Google Places-API abrufen .

Zuerst sollten Sie die `PlaceDetectionApi.getCurrentPlace()` Methode aufrufen, um lokale Geschäfte oder andere `PlaceDetectionApi.getCurrentPlace()` abzurufen. Diese Methode gibt ein `PlaceLikelihoodBuffer` Objekt zurück, das eine Liste von `PlaceLikelihood` Objekten enthält. Anschließend können Sie ein `Place` Objekt `PlaceLikelihood.getPlace()` indem Sie die `PlaceLikelihood.getPlace()` Methode aufrufen.

Wichtig: Sie müssen die `ACCESS_FINE_LOCATION` anfordern und erhalten, damit Ihre App auf genaue Standortinformationen zugreifen kann.

```
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());
}
```

Platzieren Sie die Autocomplete-Integration

Die Autovervollständigungsfunktion in der Google Places-API für Android bietet dem Nutzer Ortsvorhersagen. Während der Benutzer in das Suchfeld eingibt, zeigt Autocomplete die Orte entsprechend den Benutzerabfragen an.

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"/>
```

Hinzufügen mehrerer automatischer Google-Komplettaktivitäten

```
 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.
           }
        }
    }
```

Festlegen von Platztypfiltern für PlaceAutocomplete

In einigen Szenarien möchten wir möglicherweise die Ergebnisse, die von **PlaceAutocomplete angezeigt werden,** auf ein bestimmtes Land beschränken oder möglicherweise nur Regionen **anzeigen** . Dies kann erreicht werden, indem ein **AutocompleteFilter** in der Absicht festgelegt wird. Wenn ich beispielsweise nur nach Orten vom Typ REGION suchen möchte, die nur zu Indien gehören, würde ich Folgendes tun:

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>
```

Der **PlaceAutocomplete** wird automatisch gestartet und Sie können aus den Ergebnissen einen Ort auswählen, der nur vom Typ **REGION ist** und nur dem angegebenen Land angehört. Die Absicht kann auch auf Knopfdruck gestartet werden.

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

Kapitel 18: Android Sound und Medien

Examples

So wählen Sie ein Bild und ein Video für API> 19 aus

Hier ist ein getesteter Code für Bilder und Videos. Er funktioniert auch für alle APIs mit weniger als 19 und mehr als 19.

Bild:

```
if (Build.VERSION.SDK_INT <= 19) {
                        Intent i = new Intent();
                        i.setType("image/*");
                        i.setAction(Intent.ACTION_GET_CONTENT);
                        i.addCategory(Intent.CATEGORY_OPENABLE);
                        startActivityForResult(i, 10);
                    } else if (Build.VERSION.SDK_INT > 19) {
                        Intent intent = new Intent(Intent.ACTION_PICK,
android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                        startActivityForResult(intent, 10);
                    }
```

Video:

```
if (Build.VERSION.SDK_INT <= 19) {
                        Intent i = new Intent();
                        i.setType("video/*");
                        i.setAction(Intent.ACTION_GET_CONTENT);
                        i.addCategory(Intent.CATEGORY_OPENABLE);
                        startActivityForResult(i, 20);
                    } else if (Build.VERSION.SDK_INT > 19) {
                        Intent intent = new Intent(Intent.ACTION_PICK,
android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI);
                        startActivityForResult(intent, 20);
                    }
```

.

```
     @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == Activity.RESULT_OK) {

            if (requestCode == 10) {
                Uri selectedImageUri = data.getData();
                String selectedImagePath = getRealPathFromURI(selectedImageUri);
            } else if (requestCode == 20) {
                Uri selectedVideoUri = data.getData();
                String selectedVideoPath = getRealPathFromURI(selectedVideoUri);
            }

     public String getRealPathFromURI(Uri uri) {
```

```
            if (uri == null) {
                return null;
            }
            String[] projection = {MediaStore.Images.Media.DATA};
            Cursor cursor = getActivity().getContentResolver().query(uri, projection, null,
null, null);
            if (cursor != null) {
                int column_index = cursor
                        .getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
                cursor.moveToFirst();
                return cursor.getString(column_index);
            }
            return uri.getPath();
        }
```

Sounds über SoundPool abspielen

```
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;
    }
}
```

Android Sound und Medien online lesen: https://riptutorial.com/de/android/topic/4730/android-sound-und-medien

Kapitel 19: Android Studio

Examples

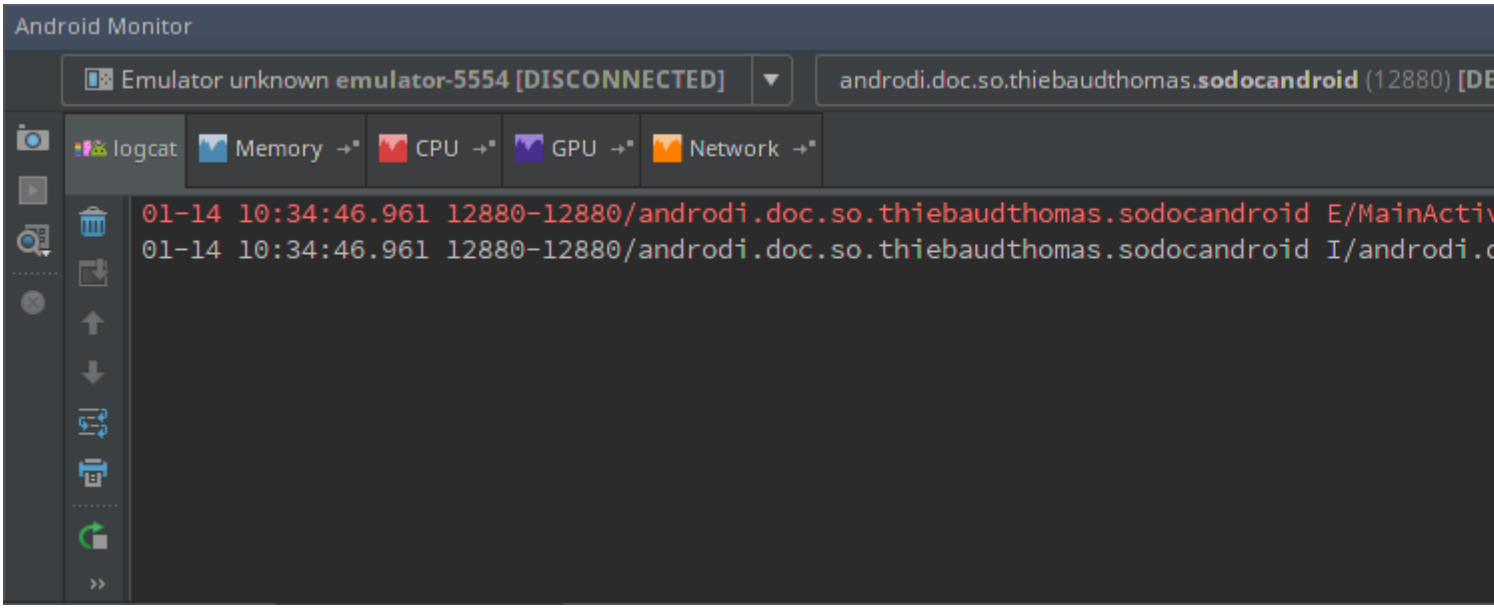
Filtern Sie Protokolle von der Benutzeroberfläche

Android-Protokolle können direkt aus der Benutzeroberfläche gefiltert werden. Verwendung dieses Codes

```
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");
    }
}
```

Wenn ich den Regex `TAG1|TAG2` und die Ebene `verbose` bekomme, bekomme ich

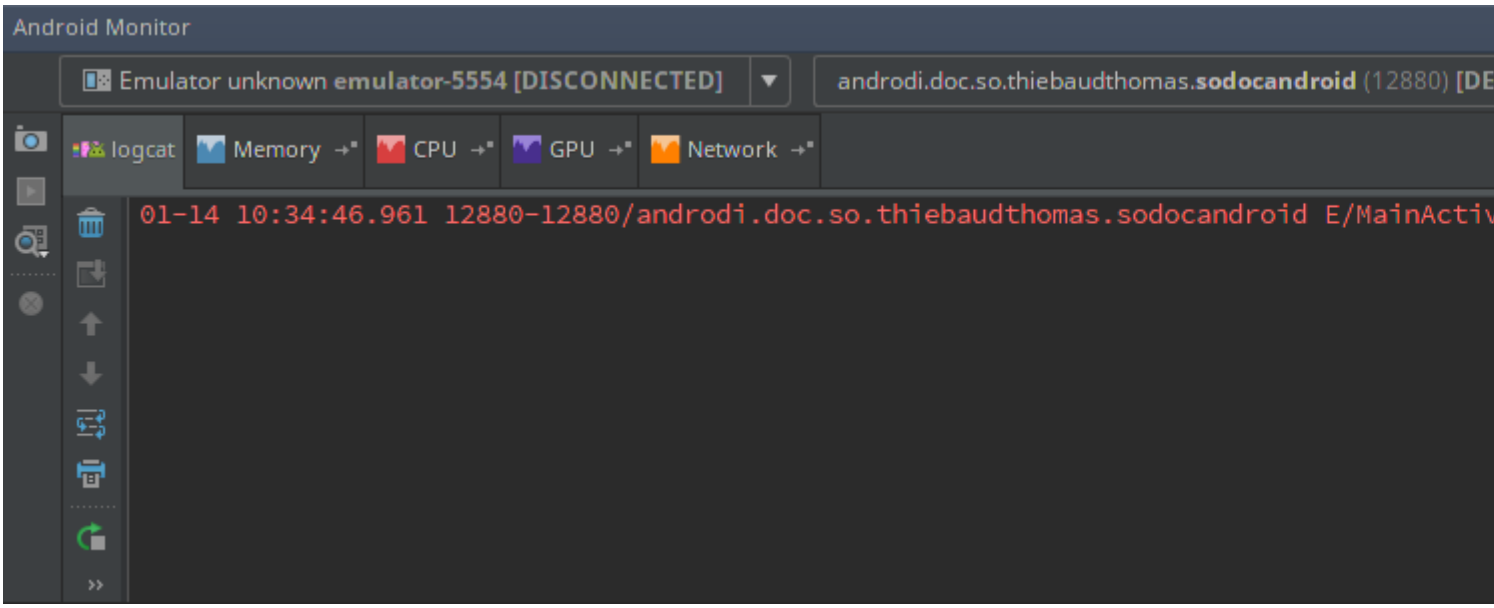
```
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log
from onCreate method with TAG1
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid
I/androdi.doc.so.thiebaudthomas.sodocandroid.MainActivity: Log from onCreate method with TAG2
```



Die Stufe kann so eingestellt werden, dass Protokolle mit einer bestimmten Stufe und darüber abgerufen werden. Zum Beispiel fängt die `verbose` Ebene `verbose, debug, info, warn, error and assert` Protokolle `verbose, debug, info, warn, error and assert` .

Nach dem gleichen Beispiel bekomme ich nur, wenn ich die Stufe auf `error` habe

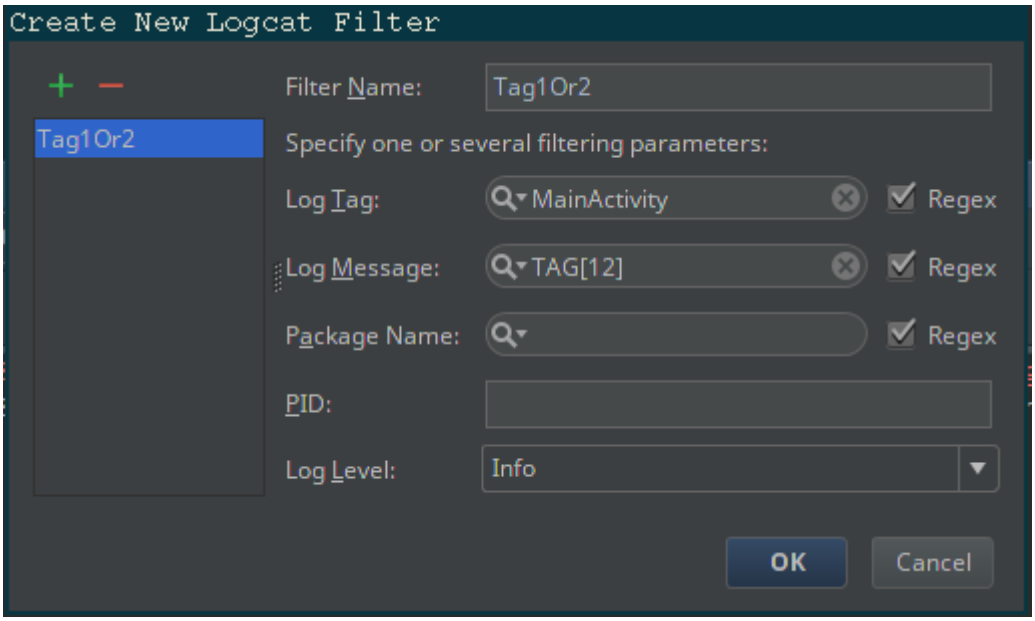
```
01-14 10:34:46.961 12880-12880/androdi.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log
from onCreate method with TAG1
```



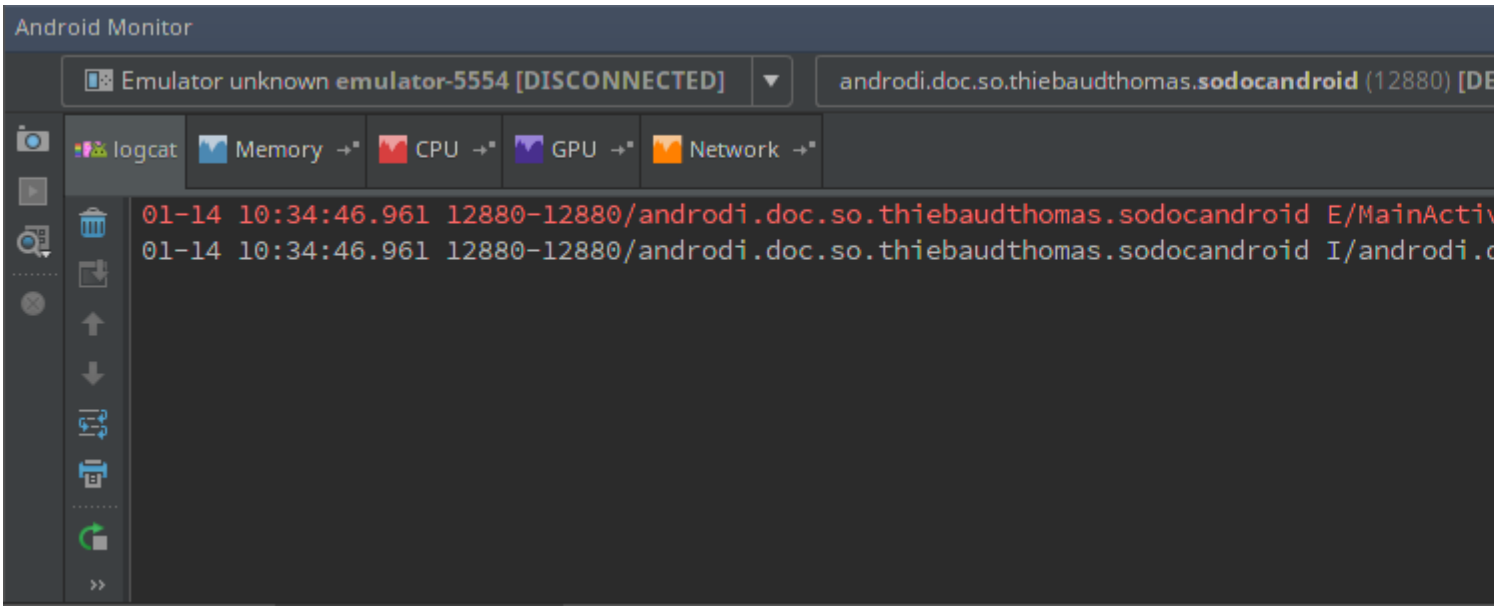
Erstellen Sie eine Filterkonfiguration

Benutzerdefinierte Filter können über die Benutzeroberfläche eingestellt und gespeichert werden. `AndroidMonitor` Sie auf der Registerkarte `AndroidMonitor` auf das rechte Dropdown-Menü (muss `Show only selected application` anzeigen oder `No filters` anzeigen), und wählen Sie `Edit filter configuration` .

Geben Sie den gewünschten Filter ein

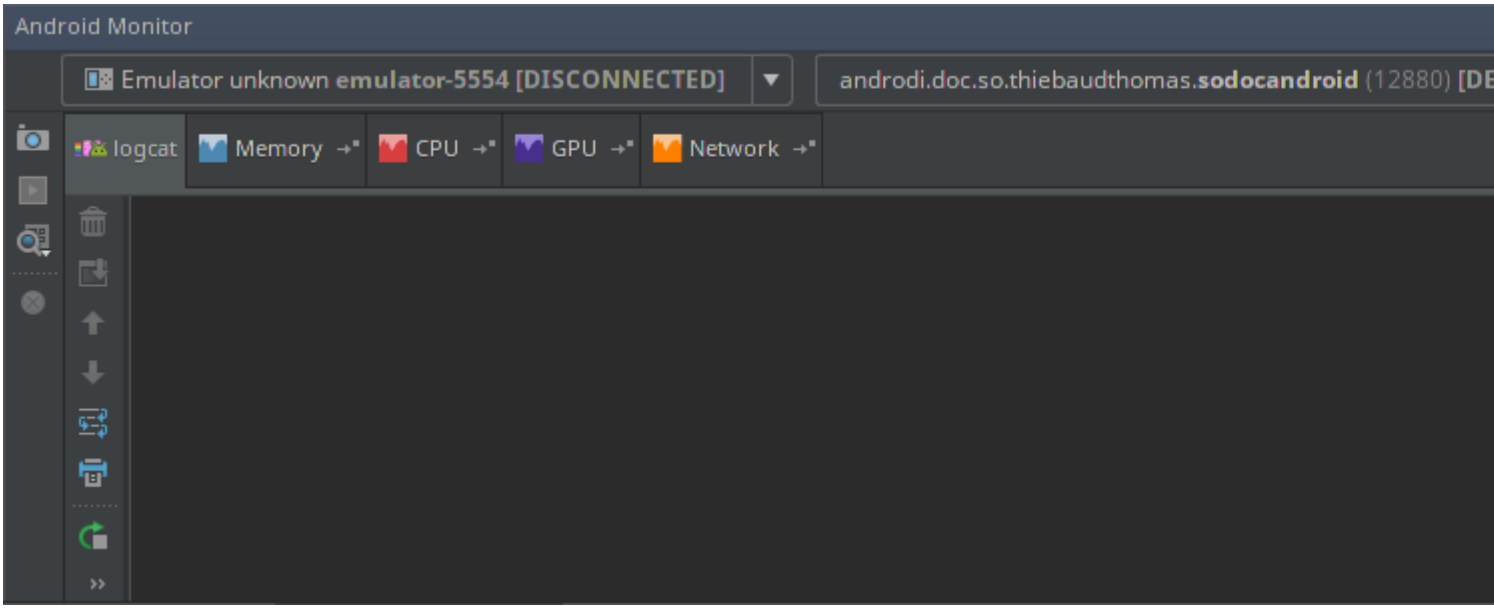


Und verwenden Sie es (Sie können es aus dem gleichen Dropdown auswählen)

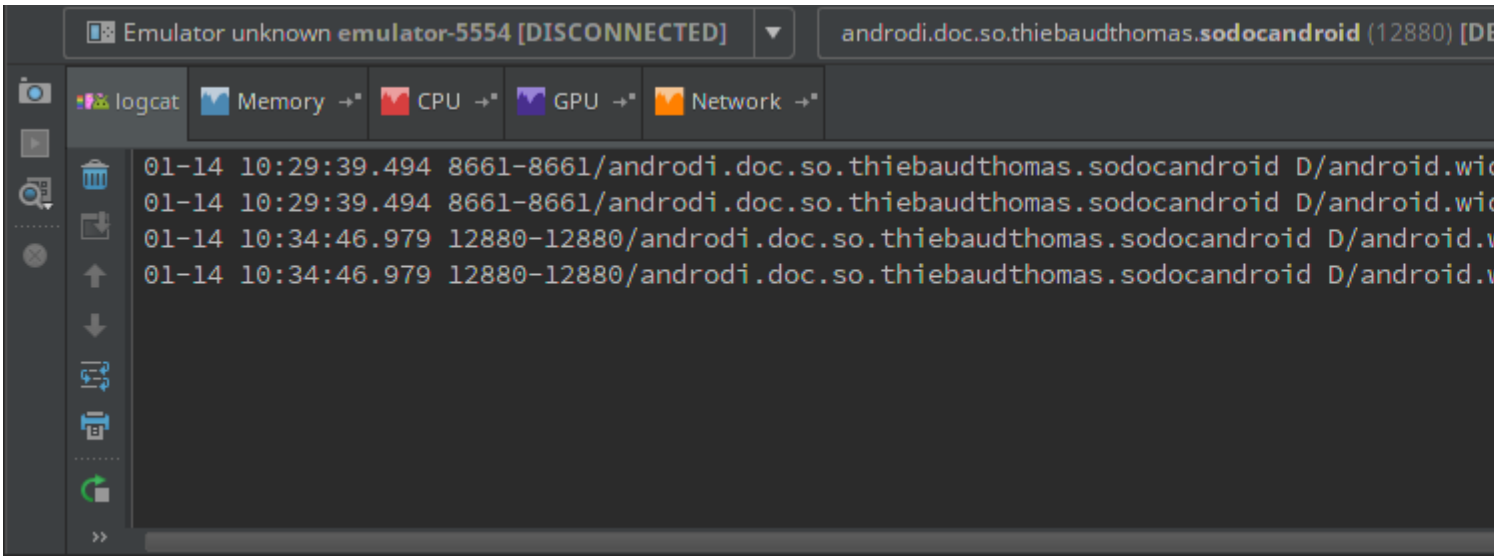


Wichtig Wenn Sie einen Eingang in die Filterleiste einfügen, berücksichtigt Android Studio sowohl Ihren Filter als auch Ihre Eingabe.

Sowohl beim Eingang als auch beim Filter gibt es keinen Ausgang



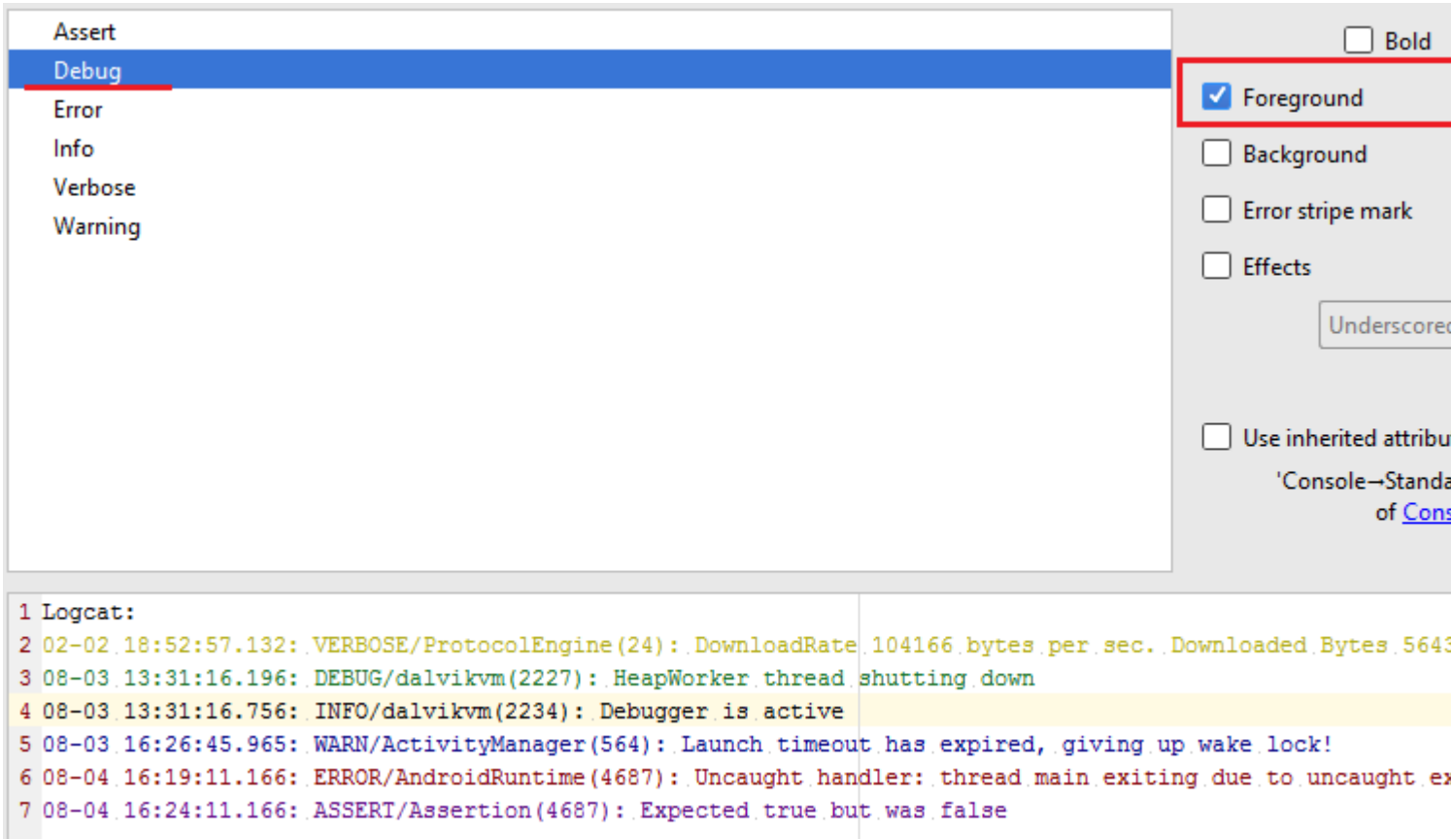
Ohne Filter gibt es einige Ausgänge



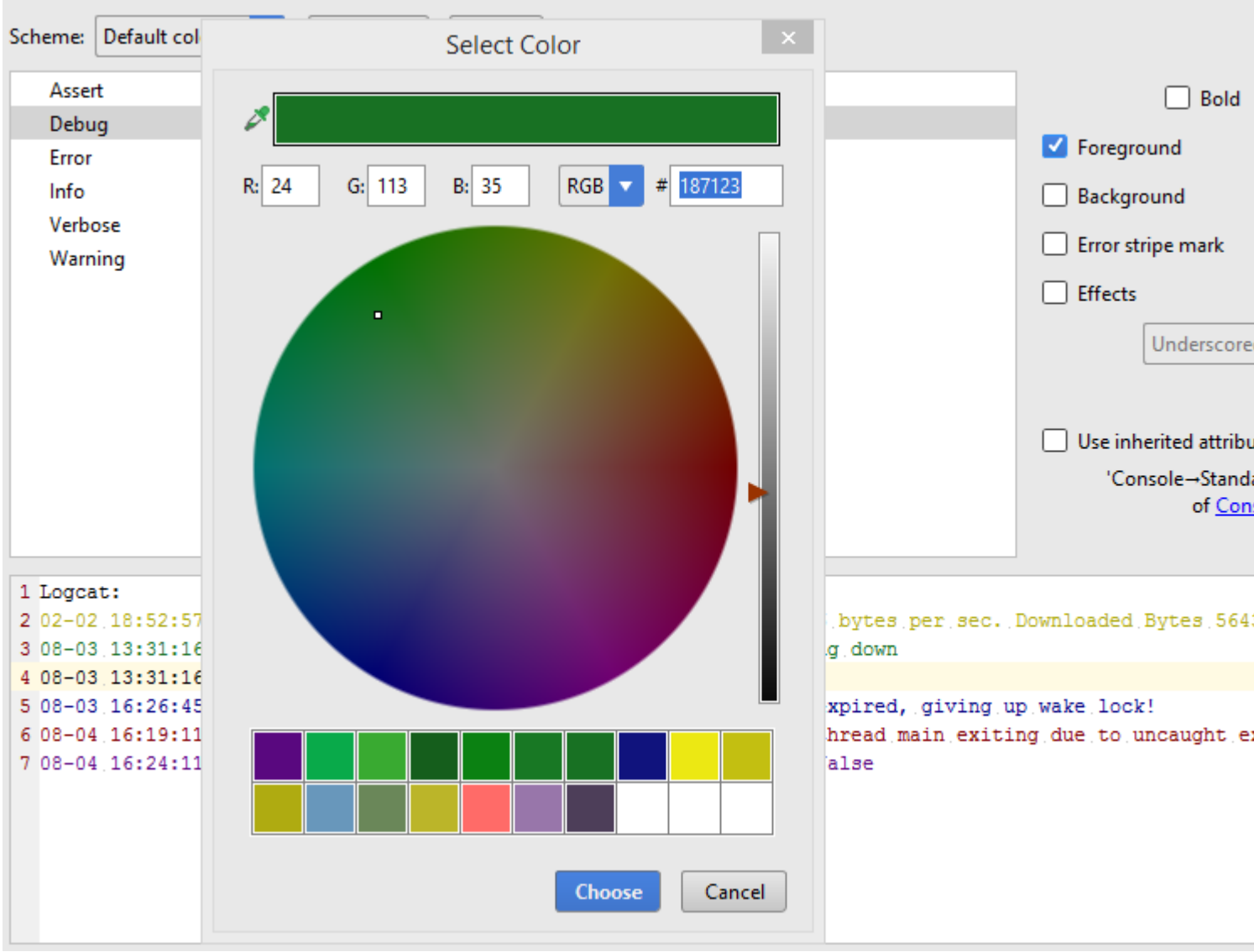
Benutzerdefinierte Farben der Logcat-Nachricht basierend auf der Wichtigkeit der Nachricht

Gehen Sie zu Datei -> Einstellungen -> Editor -> Farben und Schriftarten -> Android Logcat

Ändern Sie die Farben nach Bedarf:

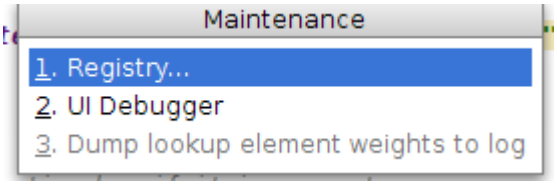


Wähle die passende Farbe:

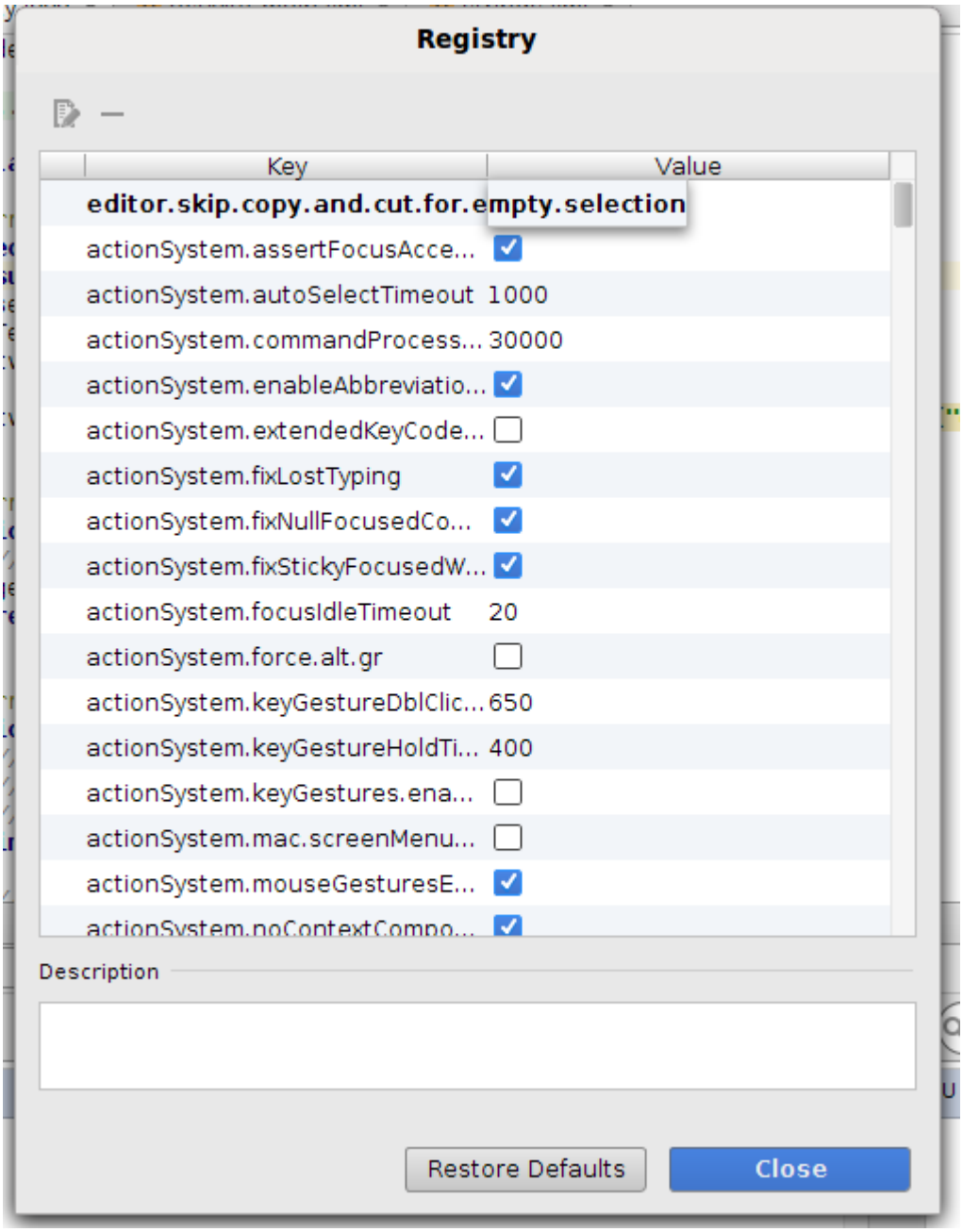


Leerzeilenkopie aktivieren / deaktivieren

`ctrl + alt + shift + /` (`cmd + alt + shift + /` unter `MacOS`) sollte den folgenden Dialog anzeigen:



Klicken Sie auf `Registry` , um zu gelangen



Der Schlüssel, den Sie aktivieren / deaktivieren möchten, ist

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

Getestet unter `Linux Ubuntu` und `MacOS` .

Android Studio nützliche Verknüpfungen

Im Folgenden sind einige der gebräuchlicheren / nützlichsten Verknüpfungen aufgeführt.

Diese basieren auf der standardmäßigen IntelliJ-Verknüpfungskarte. Sie können über `File -> Settings -> Keymap -> <Choose Eclipse/Visual Studio/etc from Keymaps dropdown>` zu anderen gängigen IDE-Shortcut-Maps `File -> Settings -> Keymap -> <Choose Eclipse/Visual Studio/etc from Keymaps dropdown>`

Aktion	Abkürzung
Code formatieren	STRG + ALT + L
Fügen Sie nicht implementierte Methoden hinzu	STRG + I
Logcat anzeigen	ALT + 6
Bauen	STRG + F9
Bauen und ausführen	STRG + F10
Finden	STRG + F
Im Projekt finden	STRG + UMSCHALT + F
Suchen und Ersetzen	STRG + R
Im Projekt suchen und ersetzen	STRG + UMSCHALT + R
Methoden überschreiben	STRG + O
Projekt anzeigen	ALT + 1
Projekt ausblenden - Logcat	UMSCHALT + ESC
Alles einklappen	STRG + UMSCHALT + NumPad +
Debug-Punkte anzeigen	STRG + UMSCHALT + F8
Alle erweitern	STRG + UMSCHALT + NumPad -
Einstellungen öffnen	ALT + s
Ziel auswählen (aktuelle Datei in der Projektansicht öffnen)	ALT + F1 → ENTER
Überall suchen	SHIFT → SHIFT (Doppelschicht)
Code | Surround mit	STRG → ALT + T
Erstellen Sie eine Methode aus dem ausgewählten Code	ALT + STRG

Refactor:

Aktion	Abkürzung
Refactor Dies (Menü / Auswahl für alle anwendbaren Refactor-Aktionen des aktuellen Elements)	Mac CTRL + T - Win / Linux CTRL + ALT + T
Umbenennen	UMSCHALT + F6

Aktion	Abkürzung
Methode extrahieren	Mac `CMD` + `ALT` + `M` - Win / Linux `STRG` + `ALT` + `M`
Parameter extrahieren	Mac `CMD` + `ALT` + `P` - Win / Linux `STRG` + `ALT` + `P`
Variable extrahieren	Mac `CMD` + `ALT` + `V` - Win / Linux `STRG` + `ALT` + `V`

Android Studio Verbessern Sie den Leistungstipp

Offline-Arbeit aktivieren:

1. Klicken Sie auf Datei -> Einstellungen. Suchen Sie nach "gradle" und klicken Sie in das Feld für die `Offline work` .
2. Gehen Sie zum Compiler (im gleichen Einstellungsdialog unterhalb von `Gradle`) und fügen Sie `--offline` zum `--offline` `Command-line Options` .

Steigfähigkeit verbessern

Fügen Sie in der Datei gradle.properties die folgende zwei Codezeile hinzu.

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

Erhöhen des Werts von `-Xmx` und `-Xms` in der Datei `studio.vmoptions`

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

Fenster

> % USERPROFILE%. {FOLDER_NAME} \ studio.exe.vmoptions und / oder% USERPROFILE%. {FOLDER_NAME} \ studio64.exe.vmoptions

Mac

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

Linux

> ~ /. {FOLDER_NAME} /studio.vmoptions und / oder ~ /. {FOLDER_NAME} /studio64.vmoptions

Richten Sie Android Studio ein

System Anforderungen

- Microsoft® Windows® 8/7 / Vista / 2003 (32 oder 64 Bit).
- Mac® OS X® 10.8.5 oder höher, bis zu 10,9 (Mavericks)
- GNOME- oder KDE-Desktop

Installation

Fenster

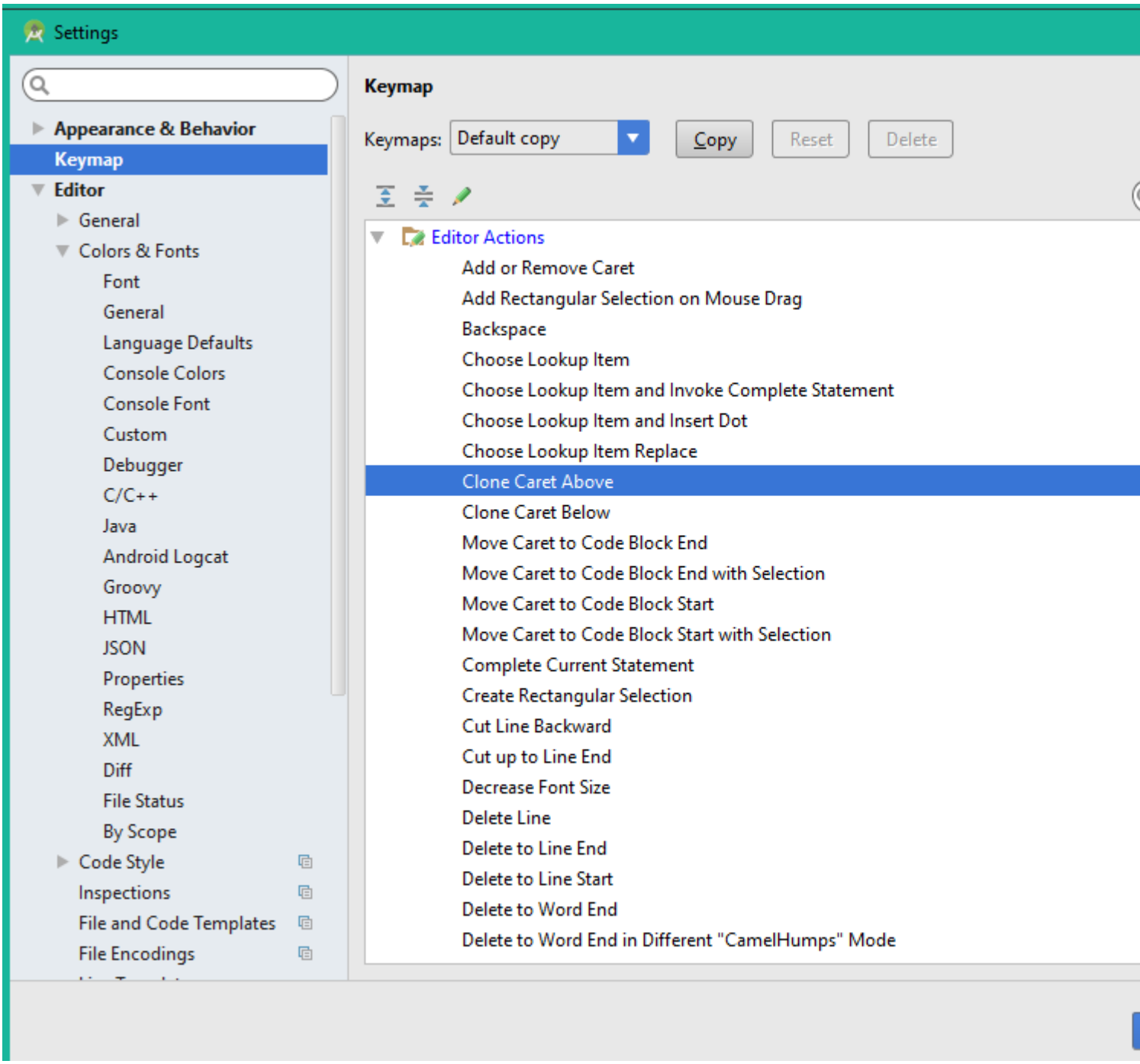
1. Laden Sie JDK (Java Development Kit) Version 8 herunter und installieren Sie es
2. Laden Sie Android Studio herunter
3. Starten Sie `Android Studio.exe` JDK-Pfad an und laden Sie das neueste SDK herunter

Linux

1. Laden Sie JDK (Java Development Kit) Version 8 herunter und installieren Sie es
2. Laden Sie Android Studio herunter
3. Extrahieren Sie die ZIP-Datei
4. Terminal öffnen, CD in den extrahierten Ordner, CD in Bin (Beispiel `cd android-studio/bin`)
5. Führen Sie ./studio.sh aus

Anzeigen und Hinzufügen von Verknüpfungen in Android Studio

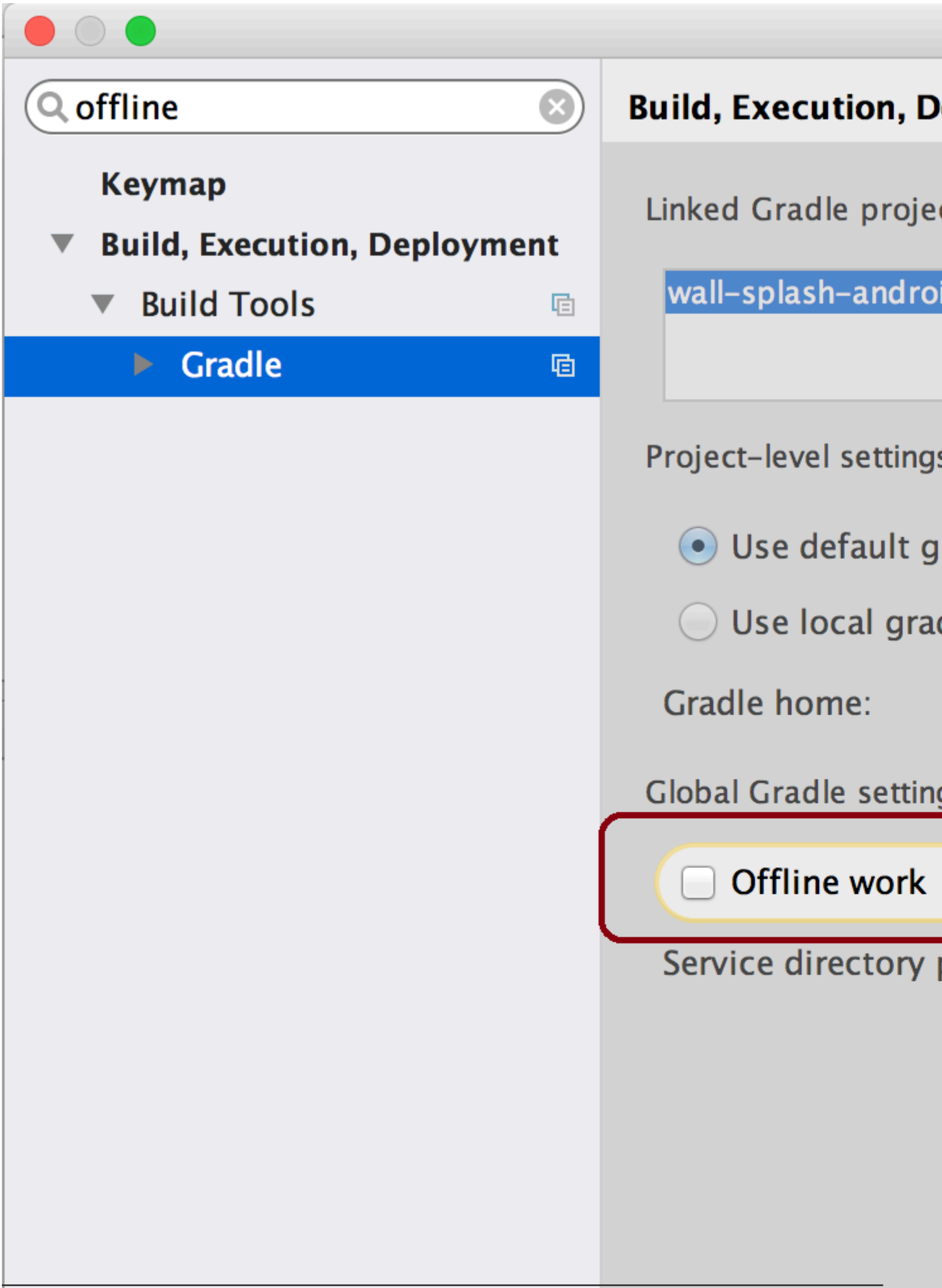
Gehen Sie zu Einstellungen >> Tastaturbeleg. Ein Fenster zeigt alle `Editor Actions` mit ihrem Namen und ihren Verknüpfungen. Einige der `Editor Actions` haben keine Verknüpfungen. Klicken Sie also mit der rechten Maustaste darauf und fügen Sie eine neue Verknüpfung hinzu. Überprüfen Sie das Bild unten



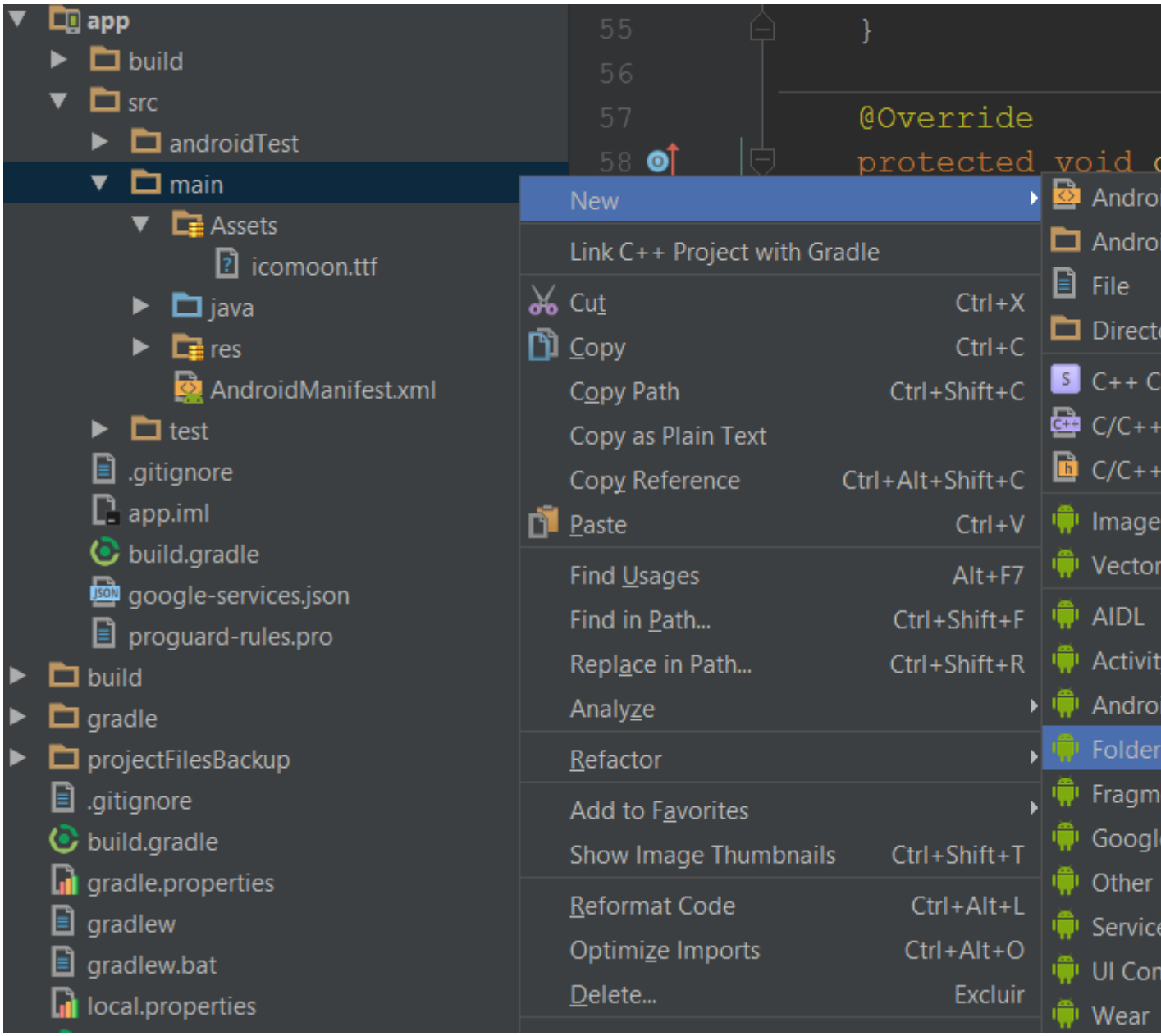
Gradle Build-Projekt dauert ewig

Android Studio -> **Einstellungen** -> **Gradle** -> Aktivieren Sie **Offline-Arbeit** und starten Sie Ihr Android Studio neu.

Referenz-Screenshot:


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

- Der Ordner "Assets" befindet sich im Ordner MAIN mit dem gleichen Symbol wie der Ordner "RES".
- In diesem Beispiel habe ich eine Schriftartdatei eingefügt.



Android Studio online lesen: https://riptutorial.com/de/android/topic/107/android-studio

Kapitel 20: Android Vk Sdk

Examples

Initialisierung und Login

1. Erstellen Sie hier eine neue Anwendung: Anwendung erstellen
2. Wählen Sie eine eigenständige Anwendung und bestätigen Sie die Erstellung der App per SMS.
3. Füllen Sie den **Paketnamen für Android** als Ihren aktuellen Paketnamen. *Sie können Ihren Paketnamen in der Android-Manifestdatei erhalten, und zwar ganz am Anfang.*
4. Holen Sie sich Ihren **Zertifikat-Fingerabdruck,** indem Sie diesen Befehl in Ihrer Shell / cmd ausführen:

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

Sie können diesen Fingerabdruck auch vom SDK selbst erhalten:

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

5. Fügen Sie den erhaltenen Fingerabdruck in den Fingerabdruck Ihres **Signaturzertifikats für Android ein:**

6. Dann fügen Sie dies Ihrer Gradle-Datei hinzu:

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

8. Initialisieren Sie das SDK beim Start mit der folgenden Methode. Am besten rufen Sie ihn in der Applications-Methode onCreate auf.

```
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);
}
```

Dies ist der beste Weg, um VKSdk zu initiieren. Verwenden Sie nicht die methid, bei der VK_ID in strings.xml eingefügt werden sollte, da api danach nicht richtig funktioniert.

9. Der letzte Schritt besteht darin, sich mit vksdk anzumelden.

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

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

Kapitel 21: Android-Architekturkomponenten

Einführung

Android Architecture Components ist eine neue Sammlung von Bibliotheken, mit deren Hilfe Sie robuste, testbare und wartbare Apps erstellen können. Hauptbestandteile sind: Lebenszyklen, ViewModel, LiveData, Room.

Examples

Fügen Sie Architekturkomponenten hinzu

Projekt 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'
}
```

Anwendungsaufbau Gradle

```
// For Lifecycles, LiveData, and ViewModel
compile "android.arch.lifecycle:runtime:$archVersion"
compile "android.arch.lifecycle:extensions:$archVersion"
annotationProcessor "android.arch.lifecycle:compiler:$archVersion"

// For Room
compile "android.arch.persistence.room:runtime:$archVersion"
annotationProcessor "android.arch.persistence.room:compiler:$archVersion"

// For testing Room migrations
testCompile "android.arch.persistence.room:testing:$archVersion"

// For Room RxJava support
compile "android.arch.persistence.room:rxjava2:$archVersion"
```

Lebenszyklus in AppCompatActivity verwenden

Erweitern Sie Ihre Aktivität aus dieser Aktivität

```
public abstract class BaseCompatLifecycleActivity extends AppCompatActivity implements
```

```
LifecycleRegistryOwner {
    // We need this class, because LifecycleActivity extends FragmentActivity not
AppCompatActivity

    @NonNull
    private final LifecycleRegistry lifecycleRegistry = new LifecycleRegistry(this);

    @NonNull
    @Override
    public LifecycleRegistry getLifecycle() {
        return lifecycleRegistry;
    }
}
```

ViewModel mit LiveData-Umwandlungen

```
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;
    }
}
```

Irgendwo in der Benutzeroberfläche:

```
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);
        }
}
```

Raumdurchlässigkeit

Raum erfordert vier Teile: Datenbankklasse, DAO-Klassen, Entitätsklassen und Migrationsklassen (jetzt können Sie **nur DDL-Methoden verwenden**):

Entity-Klassen

```
// 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-Klassen

```
@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);
}
```

Datenbankklasse

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

Migrationen

```
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() {
    }
}
```

Verwendung in der Anwendungsklasse oder über Dolch bereitstellen

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

Schreiben Sie Ihr Repository:

```
public final class ContentRepository {
    private final ContentDatabase db;
    private final VideoDao videoDao;

    public ContentRepository(ContentDatabase contentDatabase, VideoDao videoDao) {
        this.db = contentDatabase;
        this.videoDao = videoDao;
    }

    public LiveData<List<VideoItem>> getVideoByTag(@Nullable String tag, int limit) {
        // you may fetch from network, save to database
        ....
        return videoDao.getVideosByTag(tag, limit);
    }
}
```

In ViewModel verwenden:

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

Benutzerdefinierte LiveData

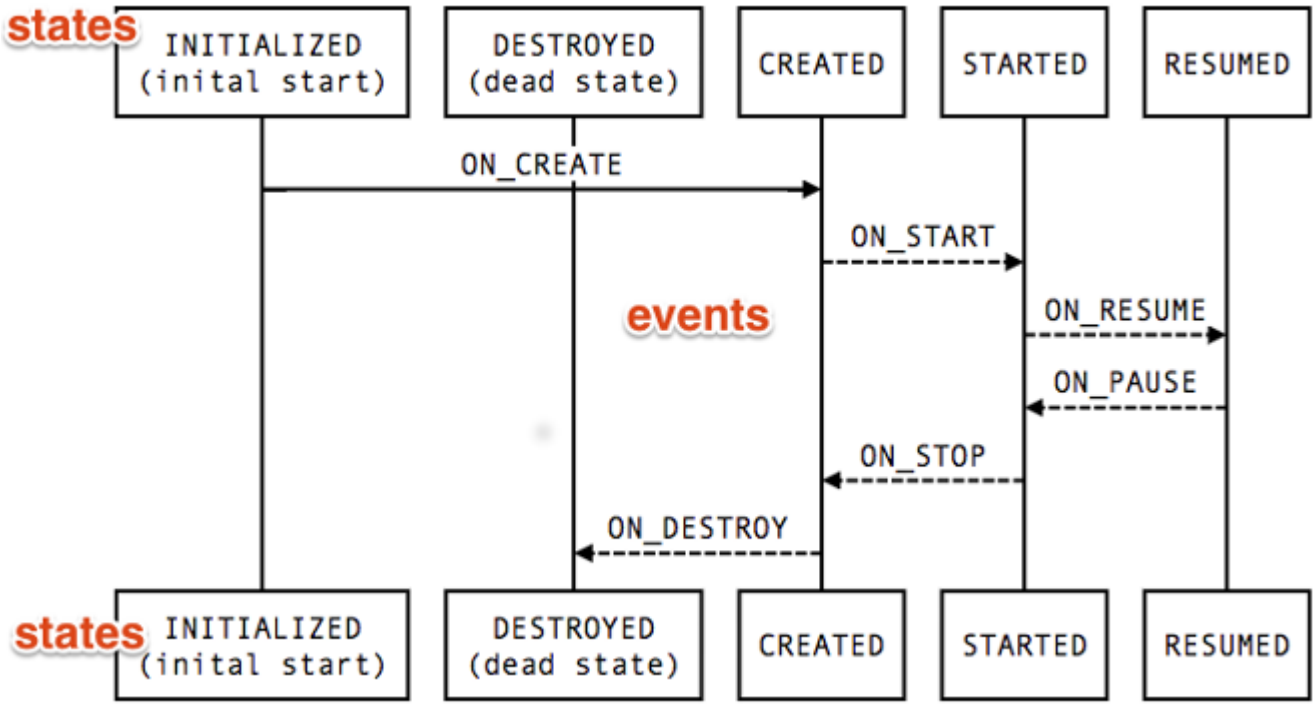
Sie können benutzerdefinierte LiveData schreiben, wenn Sie eine benutzerdefinierte Logik benötigen.
Schreiben Sie keine benutzerdefinierte Klasse, wenn Sie nur Daten transformieren müssen (verwenden Sie die Transformations-Klasse).

```
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);
    }
}
```

Benutzerdefinierte Komponente für den Lebenszyklus

Jeder Lebenszyklus der UI-Komponenten hat sich wie im Bild gezeigt geändert.



Sie können eine Komponente erstellen, die bei Änderung des Lebenszyklusstatus benachrichtigt wird:

```
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
    }
}
```

Android-Architekturkomponenten online lesen:
https://riptutorial.com/de/android/topic/10872/android-architekturkomponenten

Kapitel 22: Android-Dinge

Examples

Steuern eines Servomotors

In diesem Beispiel wird davon ausgegangen, dass Sie ein Servo mit den folgenden typischen Eigenschaften haben:

- Bewegung zwischen 0 und 180 Grad
- Impulsdauer von 20 ms
- Mindestimpulslänge von 0,5 ms
- maximale Impulslänge von 2,5 ms

Sie müssen überprüfen, ob diese Werte mit Ihrer Hardware übereinstimmen, da das Servo dadurch beschädigt werden kann, wenn es dazu gezwungen wird, den angegebenen Betriebsbereich zu verlassen. Ein beschädigtes Servo kann wiederum Ihr Android Things-Gerät beschädigen. Die Beispiel- `ServoController` Klasse besteht aus zwei Methoden, `setup()` und `setPosition()` :

```
public class ServoController {
    private double periodMs, maxTimeMs, minTimeMs;
    private Pwm pin;

    public void setup(String pinName) throws IOException {
        periodMs  = 20;
        maxTimeMs = 2.5;
        minTimeMs = 0.5;

        PeripheralManagerService service = new PeripheralManagerService();
        pin = service.openPwm(pinName);

        pin.setPwmFrequencyHz(1000.0d / periodMs);
        setPosition(90);
        pin.setEnabled(true);
    }

    public void setPosition(double degrees) {
        double pulseLengthMs = (degrees / 180.0 * (maxTimeMs - minTimeMs)) + minTimeMs;

        if (pulseLengthMs < minTimeMs) {
            pulseLengthMs = minTimeMs;
        } else if (pulseLengthMs > maxTimeMs) {
            pulseLengthMs = maxTimeMs;
        }

        double dutyCycle = pulseLengthMs / periodMs * 100.0;

        Log.i(TAG, "Duty cycle = " + dutyCycle + " pulse length = " + pulseLengthMs);

        try {
            pin.setPwmDutyCycle(dutyCycle);
        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

Sie können Pin-Namen, die PWM auf Ihrem Gerät unterstützen, wie folgt ermitteln:

```
PeripheralManagerService service = new PeripheralManagerService();

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

Damit Ihr Servo für immer zwischen 80 Grad und 100 Grad schwingt, können Sie einfach den folgenden Code verwenden:

```
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();
```

Sie können den gesamten obigen Code kompilieren und implementieren, ohne Servomotoren tatsächlich an das Computergerät anzuschließen. Informationen zur Verdrahtung finden Sie in der Pinbelegungstabelle Ihres Computers (z. B. eine Pinbelegungstabelle des Raspberry Pi 3 ist hier verfügbar).

Dann müssen Sie Ihr Servo an Vcc, Gnd und Signal anschließen.

Android-Dinge online lesen: https://riptutorial.com/de/android/topic/8938/android-dinge

Kapitel 23: Android-Kernel-Optimierung

Examples

Niedrige RAM-Konfiguration

Android unterstützt jetzt Geräte mit 512 MB RAM. Diese Dokumentation soll OEMs helfen, Android 4.4 für Geräte mit geringem Speicherplatz zu optimieren und zu konfigurieren. Einige dieser Optimierungen sind generisch genug, um sie auch auf frühere Versionen anwenden zu können.

Low Ram Device Flag aktivieren

Wir führen eine neue API mit dem Namen ActivityManager.isLowRamDevice () für Anwendungen ein, um zu bestimmen, ob bestimmte speicherintensive Funktionen deaktiviert werden sollen, die auf Geräten mit wenig Arbeitsspeicher schlecht funktionieren.

Für 512-MB-Geräte wird erwartet, dass diese API zurückgibt: "true". Sie kann durch die folgende Systemeigenschaft im Geräte-Makefile aktiviert werden.

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

Deaktivieren Sie JIT

Die systemweite Nutzung des JIT-Speichers hängt von der Anzahl der laufenden Anwendungen und dem Code-Footprint dieser Anwendungen ab. Die JIT legt eine maximale Cache-Größe für übersetzten Code fest und berührt die darin enthaltenen Seiten nach Bedarf. JIT kostet bei einem typischen laufenden System zwischen 3 und 6 Millionen.

Die großen Apps neigen dazu, den Code-Cache relativ schnell zu maximieren (standardmäßig 1M). Im Durchschnitt wird die JIT-Cache-Nutzung zwischen 100 KB und 200 KB Bytes pro App ausgeführt. Das Reduzieren der maximalen Cache-Größe kann bei der Speichernutzung etwas hilfreich sein. Wenn Sie jedoch den Wert für "zu niedrig" festlegen, wird die JIT in einen Thrash-Modus versetzt. Bei Geräten mit sehr geringem Speicher empfehlen wir, die JIT vollständig zu deaktivieren.

Dies kann durch Hinzufügen der folgenden Zeile zum Produkt-Makefile erreicht werden:

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

So fügen Sie einen CPU-Governor hinzu

Der CPU-Governor selbst ist nur eine C-Datei, die sich in Kernel_source / drivers / cpufreq / befindet, zum Beispiel: cpufreq_smartass2.c. Sie sind selbst dafür verantwortlich, den Governor zu finden (in einem vorhandenen Kernel-Repo nach Ihrem Gerät suchen). Um diese Datei jedoch erfolgreich aufrufen und in Ihrem Kernel kompilieren zu können, müssen Sie folgende Änderungen

vornehmen:

1. Kopieren Sie Ihre Governor-Datei (cpufreq_govname.c) und navigieren Sie zu kernel_source / drivers / cpufreq. Fügen Sie sie nun ein.
2. und öffnen Sie Kconfig (dies ist die Oberfläche des Konfigurationsmenü-Layouts), wenn Sie einen Kernel hinzufügen, der in Ihrer Konfiguration angezeigt werden soll. Sie können dies tun, indem Sie die Wahl des Governors hinzufügen.

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

Zum Beispiel für smartassV2.

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

Neben dem Hinzufügen der Auswahl müssen Sie auch die Möglichkeit angeben, dass der Governor als Standard-Governor ausgewählt wird.

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

Zum Beispiel für smartassV2.

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

- kann nicht den richtigen Ort finden, um es zu platzieren? Suchen Sie einfach nach `"CPU_FREQ_GOV_CONSERVATIVE"` und `"CPU_FREQ_GOV_CONSERVATIVE"` den Code darunter ein. Gleiches gilt für `"CPU_FREQ_DEFAULT_GOV_CONSERVATIVE"`

Nachdem Kconfig fertig ist, können Sie die Datei speichern und schließen.

3. Öffnen Sie Makefile, während Sie sich noch im Ordner `/drivers/cpufreq` . Fügen Sie in Makefile die Zeile hinzu, die Ihrem CPU-Regler entspricht. zum Beispiel:

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

Seien Sie sich jedoch bewusst, dass Sie nicht die native C-Datei, sondern die O-Datei aufrufen!

Welches ist die kompilierte C-Datei. Speicher die Datei.

4. `kernel_source/includes/linux` zu: `kernel_source/includes/linux` . Öffnen `cpufreq.h` nun `cpufreq.h` unten, bis `cpufreq.h` :

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

(andere CPU-Gouverneure sind auch dort aufgeführt)

Fügen Sie nun Ihren Eintrag mit dem ausgewählten CPU-Governor hinzu, Beispiel:

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

Speichern Sie die Datei und schließen Sie sie.

Das anfängliche CPU Governor-Setup ist nun abgeschlossen. Wenn Sie alle Schritte erfolgreich durchgeführt haben, sollten Sie in der Lage sein, Ihren Governor aus dem Menü `menuconfig` (`menuconfig` , `xconfig` , `gconfig` , `nconfig`). Nach dem Einchecken im Menü wird es in den Kernel aufgenommen.

Festschreiben, das fast genauso ist wie die obigen Anweisungen: Fügen Sie smartassV2 und lulzactive gouverneur fest

E / A-Scheduler

Sie können Ihren Kernel erweitern, indem Sie bei Bedarf neue E / A-Scheduler hinzufügen. Weltweit sind Governors und Scheduler gleich. beide bieten eine Möglichkeit, wie das System funktionieren sollte. Bei den Schedulern geht es jedoch ausschließlich um den Ein- / Ausgabedatenstrom mit Ausnahme der CPU-Einstellungen. E / A-Scheduler entscheiden, wie eine bevorstehende E / A-Aktivität geplant wird. Die Standard-Scheduler wie *noop* oder *cfq* arbeiten sehr vernünftig.

E / A-Scheduler finden Sie im *Kernel_source / -Block* .

1. Kopieren Sie Ihre E / A-Scheduler-Datei (z. B. " *sio-iosched.c"*) und *navigieren Sie* zu " *kernel_source / block"* . Fügen Sie die Scheduler-Datei dort ein.

2. Öffnen *Sie* nun *Kconfig.iosched* und fügen Sie Ihre Auswahl zum *Kconfig hinzu* , zum Beispiel für *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. Legen Sie dann die Standardauswahloption wie folgt fest:

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

   Speicher die Datei.

4. Öffnen Sie das *Makefile* in *kernel_source / block /* und fügen Sie einfach die folgende Zeile für *SIO hinzu* :

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

   Speichern Sie die Datei und Sie sind fertig! Die I / O-Scheduler sollten jetzt im Menü config erscheinen.

Ähnliches Commit für GitHub: Simple I / O-Scheduler hinzugefügt .

Android-Kernel-Optimierung online lesen: https://riptutorial.com/de/android/topic/9106/android-kernel-optimierung

Kapitel 24: Android-Programmierung mit Kotlin

Einführung

Die Verwendung von Kotlin mit Android Studio ist eine einfache Aufgabe, da Kotlin von JetBrains entwickelt wird. Es ist das gleiche Unternehmen, das hinter IntelliJ IDEA steht - einer Basis-IDE für Android Studio. Deshalb gibt es fast keine Probleme mit der Kompatibilität.

Bemerkungen

Wenn Sie mehr über die Programmiersprache Kotlin erfahren möchten, lesen Sie die Dokumentation .

Examples

Kotlin-Plugin installieren

Zuerst müssen Sie das Kotlin-Plugin installieren.

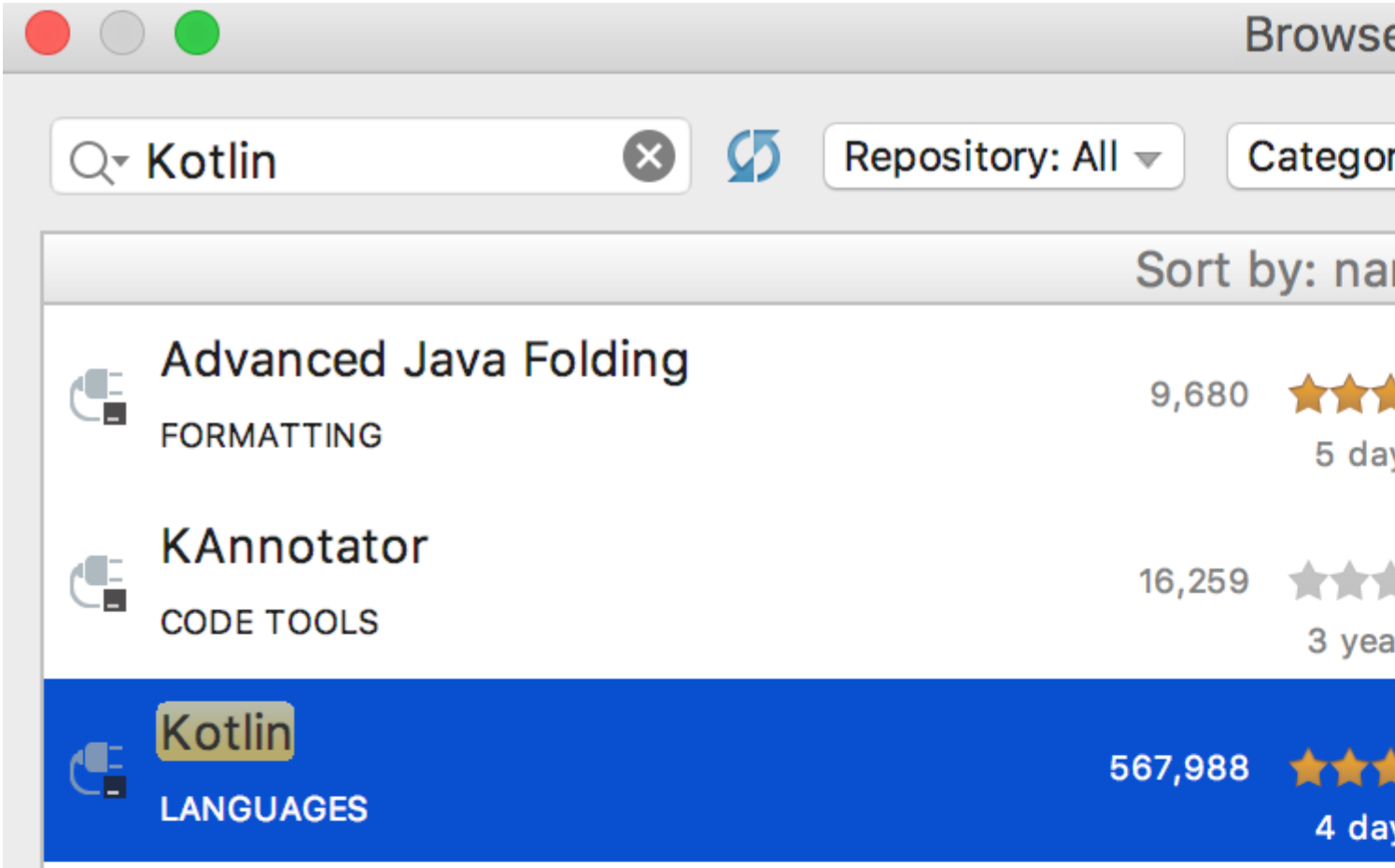
Für Windows:

- Navigieren Sie zu `File` → `Settings` → `Plugins` → `Install JetBrains plugin`

Für Mac:

- Navigieren Sie zu `Android Studio` → `Preferences` → `Plugins` → `Install JetBrains plugin`

Suchen Sie dann nach Kotlin und installieren Sie es. Sie müssen die IDE danach neu starten.


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

erstellen und dann die Kotlin-Unterstützung hinzufügen oder Ihr vorhandenes Projekt ändern. Dazu müssen Sie:

1. **Hinzufügen einer Abhängigkeit zu einer Stamm-Gradle-Datei** - Sie müssen die Abhängigkeit für `kotlin-android` Plugin zu einer Stamm- `build.gradle` Datei `build.gradle` .

```
buildscript {

    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.1'
        classpath 'org.jetbrains.kotlin:kotlin-gradle-plugin:1.1.2'
    }
}

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

2. **Apply Kotlin Android Plugin** - fügen Sie einfach `apply plugin: 'kotlin-android'` zu einer Modul- `build.gradle` Datei hinzu.

3. **Fügen Sie der Kotlin stdlib** eine Abhängigkeit **hinzu** - fügen Sie die Abhängigkeit zu `'org.jetbrains.kotlin:kotlin-stdlib:1.1.2'` zum Abhängigkeitsabschnitt in einer Modul- `build.gradle` Datei hinzu.

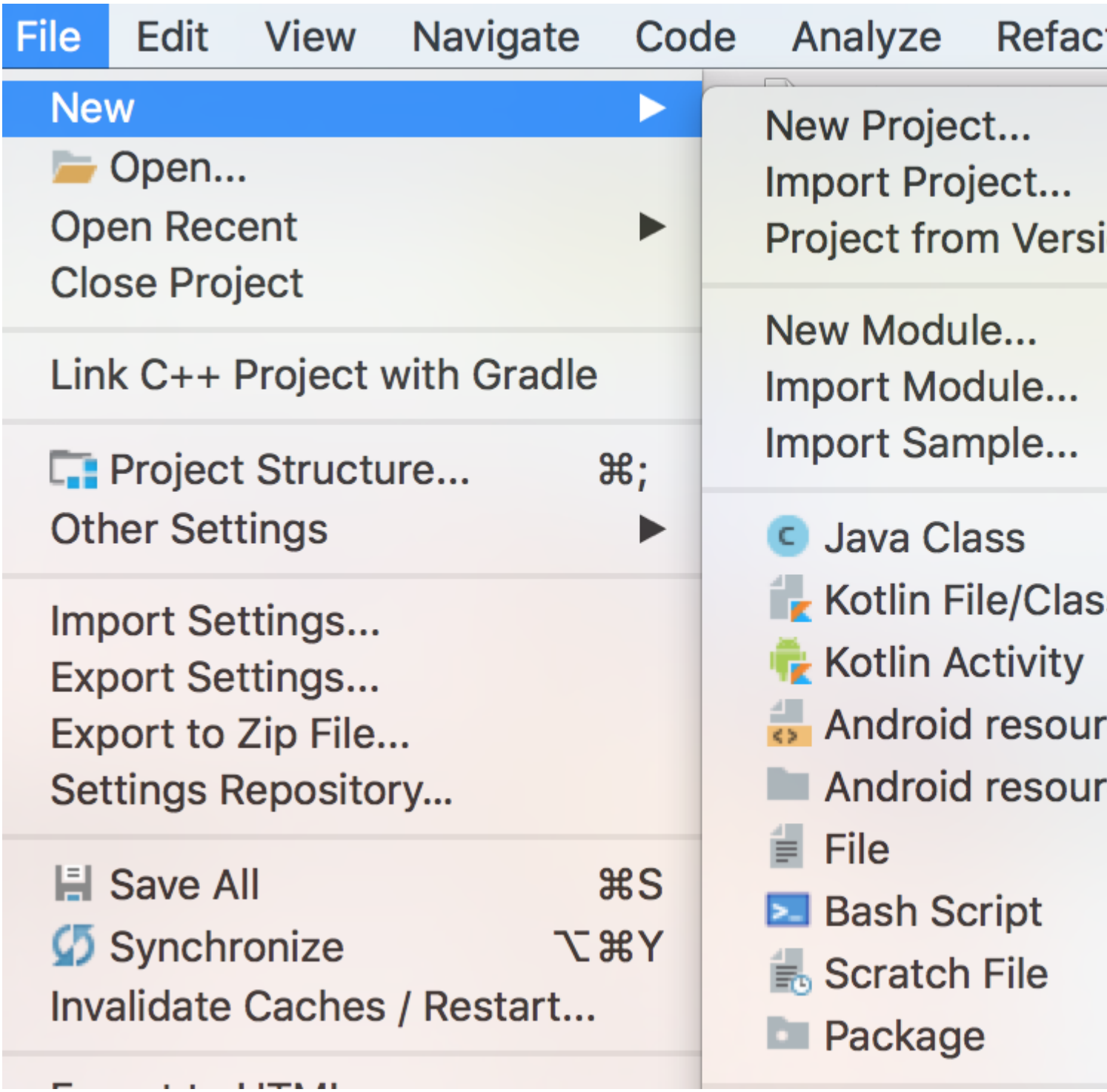
Für ein neues Projekt könnte die `build.gradle` Datei folgendermaßen aussehen:

```
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'
}
```

Neue Kotlin-Aktivität erstellen

1. Klicken Sie auf `File` → `New` → `Kotlin Activity` .
2. Wählen Sie eine Art der Aktivität.
3. Wählen Sie den Namen und andere Parameter für die Aktivität aus.
4. Fertig.

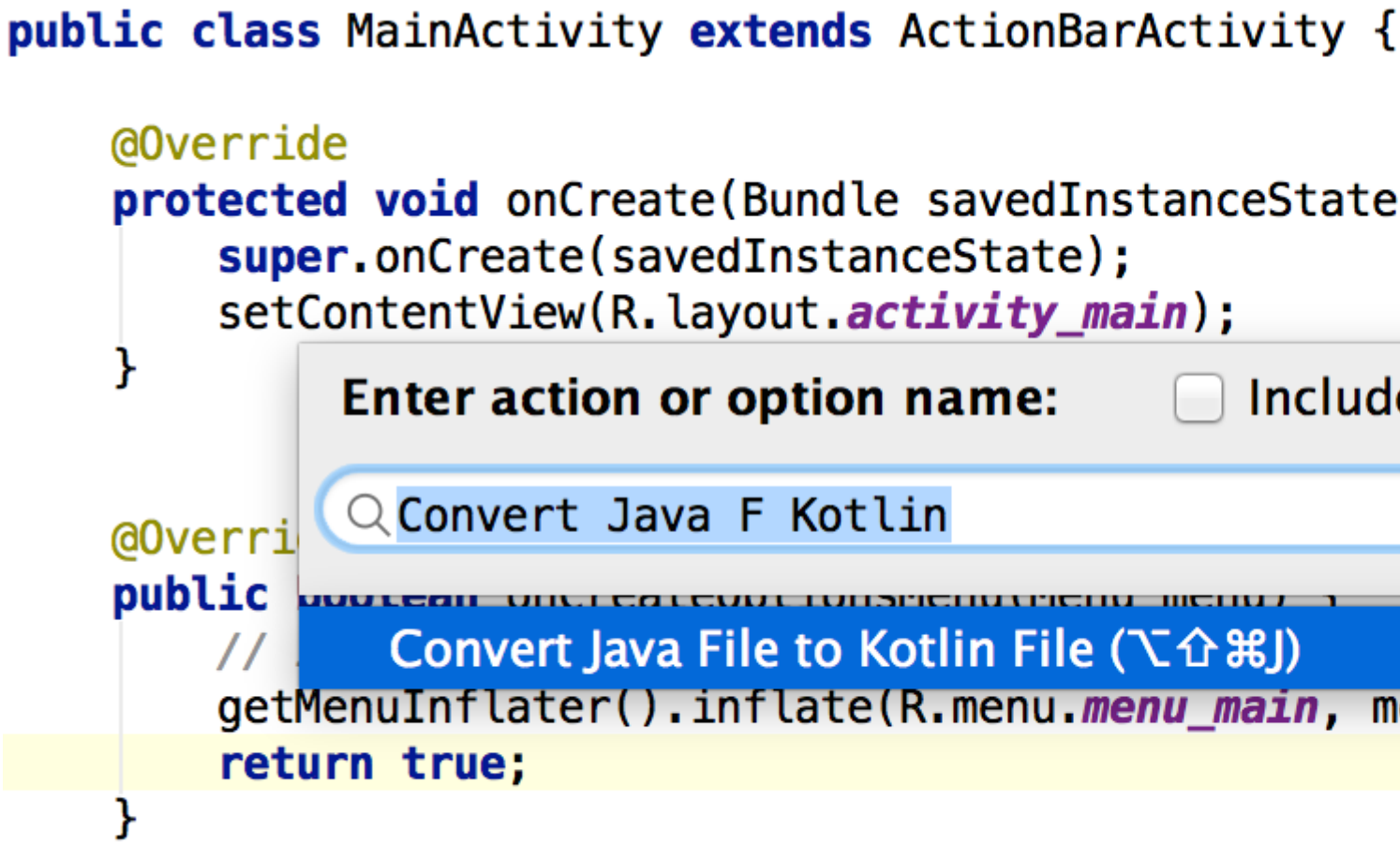


Die Abschlussklasse könnte so aussehen:

```
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)
    }
}
```

Bestehenden Java-Code in Kotlin konvertieren

Kotlin Plugin für Android Studio unterstützt das Konvertieren vorhandener Java-Dateien in Kotlin-Dateien. Wählen Sie eine Java-Datei aus und rufen Sie die Aktion auf. Konvertieren Sie Java-Datei in Kotlin-Datei:



Eine neue Aktivität starten

```
fun startNewActivity(){
    val intent: Intent = Intent(context, Activity::class.java)
    startActivity(intent)
}
```

Sie können wie in Java der Intent Extras hinzufügen.

```
fun startNewActivityWithIntents(){
    val intent: Intent = Intent(context, Activity::class.java)
    intent.putExtra(KEY_NAME, KEY_VALUE)
    startActivity(intent)
}
```

Android-Programmierung mit Kotlin online lesen:
https://riptutorial.com/de/android/topic/9623/android-programmierung-mit-kotlin

Kapitel 25: Android-Spieleentwicklung

Einführung

Eine kurze Einführung in die Erstellung eines Spiels auf der Android-Plattform mit Java

Bemerkungen

- Das erste Beispiel behandelt die Grundlagen: Es gibt keine Ziele, aber es zeigt Ihnen, wie Sie einen grundlegenden Teil eines 2D-Spiels mit SurfaceView erstellen.
- Stellen Sie sicher, dass Sie wichtige Daten speichern, wenn Sie ein Spiel erstellen. alles andere wird verloren gehen

Examples

Spiel mit Canvas und SurfaceView

Hier erfahren Sie, wie Sie mit SurfaceView ein einfaches 2D-Spiel erstellen können.

Zuerst brauchen wir eine Aktivität:

```
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.
    }

}
```

Die Aktivität muss auch im Android-Manifest deklariert werden.

Nun zum Spiel selbst. Zuerst beginnen wir mit der Implementierung eines Game-Threads:

```
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");
    }
}
```

Das ist der grundlegende Teil. Jetzt haben Sie die Möglichkeit, auf den Bildschirm zu zeichnen.

Beginnen wir mit dem Hinzufügen von Ganzzahlen:

```
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;
```

Für den nächsten Teil benötigen Sie ein Bild. Sie sollte etwa 100x100 betragen, kann jedoch größer oder kleiner sein. Zum Lernen kann auch ein Rect verwendet werden (dies erfordert jedoch eine geringfügige Änderung des Codes).

Jetzt deklarieren wir eine Bitmap:

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

Beim Rendern müssen wir diese Bitmap zeichnen.

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

Vor dem Start gibt es noch einige Dinge zu tun

Wir brauchen zuerst einen Boolean:

```
boolean up = false;
```

In onTouchEvent fügen wir hinzu:

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

Und in tick brauchen wir das, um den Spieler zu bewegen:

```
if(up){
```

```
    velY -=1;
}
else{
    velY +=1;
}
if(velY >14)velY = 14;
if(velY <-14)velY = -14;
y += velY *2;
```

und jetzt brauchen wir das in init:

```
WindowManager wm = (WindowManager) c.getSystemService(Context.WINDOW_SERVICE);
Display display = wm.getDefaultDisplay();
Point size = new Point();
display.getSize(size);
WIDTH = size.x;
HEIGHT = size.y;
y = HEIGHT/ 2 - PLAYER_BMP.getHeight();
```

Und wir brauchen diese Variablen:

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

Zu diesem Zeitpunkt ist das Spiel lauffähig. Das heißt, Sie können es starten und testen.

Jetzt sollten Sie ein Player-Image haben oder den Bildschirm nach oben und unten bewegen. Der Player kann bei Bedarf als benutzerdefinierte Klasse erstellt werden. Dann können alle spielerbezogenen Dinge in diese Klasse verschoben werden und eine Instanz dieser Klasse zum Verschieben, Rendern und Ausführen anderer Logik verwenden.

Nun, wie Sie wahrscheinlich beim Testen gesehen haben, fliegt es vom Bildschirm. Wir müssen es also einschränken.

Zuerst müssen wir das Rect deklarieren:

```
private Rect screen;
```

In init erstellen wir nach dem Initialisieren von Breite und Höhe ein neues Rechteck, das der Bildschirm ist.

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

Nun brauchen wir ein weiteres Rect in Form einer Methode:

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

und in tick:

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

Die Implementierung von gameOVer kann auch verwendet werden, um den Start eines Spiels zu zeigen.

Andere Aspekte eines Spiels, die es zu beachten gilt:

Speichern (derzeit fehlt in der Dokumentation)

Android-Spieleentwicklung online lesen: https://riptutorial.com/de/android/topic/10011/android-spieleentwicklung

Kapitel 26: Android-Versionen

Bemerkungen

Name	Android-Version	Veröffentlichungsdatum	API-Ebene	Build.VERSION_CODES
Engelskuchen (Alpha)	1,0	23. September 2008	1	BASE
Battenberg (Beta)	1.1	9. Februar 2009	2	BASE_1_1
Cupcake	1,5	30. April 2009	3	CUPCAKE
Krapfen	1.6	15. September 2009	4	KRAPFEN
Eclair	2,0	26. Oktober 2009	5	ECLAIR
	2.0.1	3. Dezember 2009	6	ECLAIR_0_1
	2.1	12. Januar 2010	7	ECLAIR_MR1
Froyo	2.2	20. Mai 2010	8	FROYO
Lebkuchen	2.3	6. Dezember 2010	9	LEBKUCHEN
	2.3.3	9. Februar 2011	10	GINGERBREAD_MR1
Bienenwabe	3,0	22. Februar 2011	11	BIENENWABE
	3.1	10. Mai 2011	12	HONEYCOMB_MR2
	3.2	15. Juli 2011	13	HONEYCOMB_MR1
Eiscreme-Sandwich	4,0	19. Oktober 2011	14	EISCREME-SANDWICH
	4.0.3	16. Dezember 2011	fünfzehn	ICE_CREAM_SANDWICH_MR1
Geleebohne	4.1	9. Juli 2012	16	GELEEBOHNE
	4.2	13. November 2012	17	JELLY_BEAN_MR1
	4.3	24. Juli 2013	18	JELLY_BEAN_MR2
KitKat	4.4	31. Oktober 2013	19	KITKAT

Name	Android-Version	Veröffentlichungsdatum	API-Ebene	Build.VERSION_CODES
		25. Juli 2014	20	KITKAT_WATCH
Lutscher	5,0	17. Oktober 2014	21	LUTSCHER
	5.1	9. März 2015	22	LOLLIPOP_MR1
Marshmallow	6,0	5. Oktober 2015	23	M
Nougat	7,0	22. August 2016	24	N
	7.1.1	5. Dezember 2016	25	N_MR1

Examples

Überprüfen der Android-Version zur Laufzeit auf dem Gerät

`Build.VERSION_CODES` ist eine Aufzählung der derzeit bekannten SDK-Versionscodes.

Verwenden `TargetApi` zur bedingten Ausführung von Code, der auf der Android-Version des Geräts basiert, die `TargetApi` Anmerkung, um Lint-Fehler zu vermeiden, und überprüfen Sie die `TargetApi` , bevor Sie den für die API-Ebene spezifischen Code ausführen.

Hier ein Beispiel für die Verwendung einer in API-23 eingeführten Klasse in einem Projekt, das API-Level niedriger als 23 unterstützt:

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

Android-Versionen online lesen: https://riptutorial.com/de/android/topic/3264/android-versionen

Kapitel 27: Android-x86 in VirtualBox

Einführung

In diesem Abschnitt wird die Installation und Verwendung der VirtualBox mit Android-x86 für Debugging-Zwecke beschrieben. Dies ist eine schwierige Aufgabe, da Unterschiede zwischen den Versionen bestehen. Im Moment werde ich 6.0 behandeln, mit der ich arbeiten musste und dann müssen wir Ähnlichkeiten finden.

Es behandelt VirtualBox oder Linux nicht im Detail, zeigt aber die Befehle, die ich verwendet habe, damit es funktioniert.

Examples

Einrichten der virtuellen Maschine

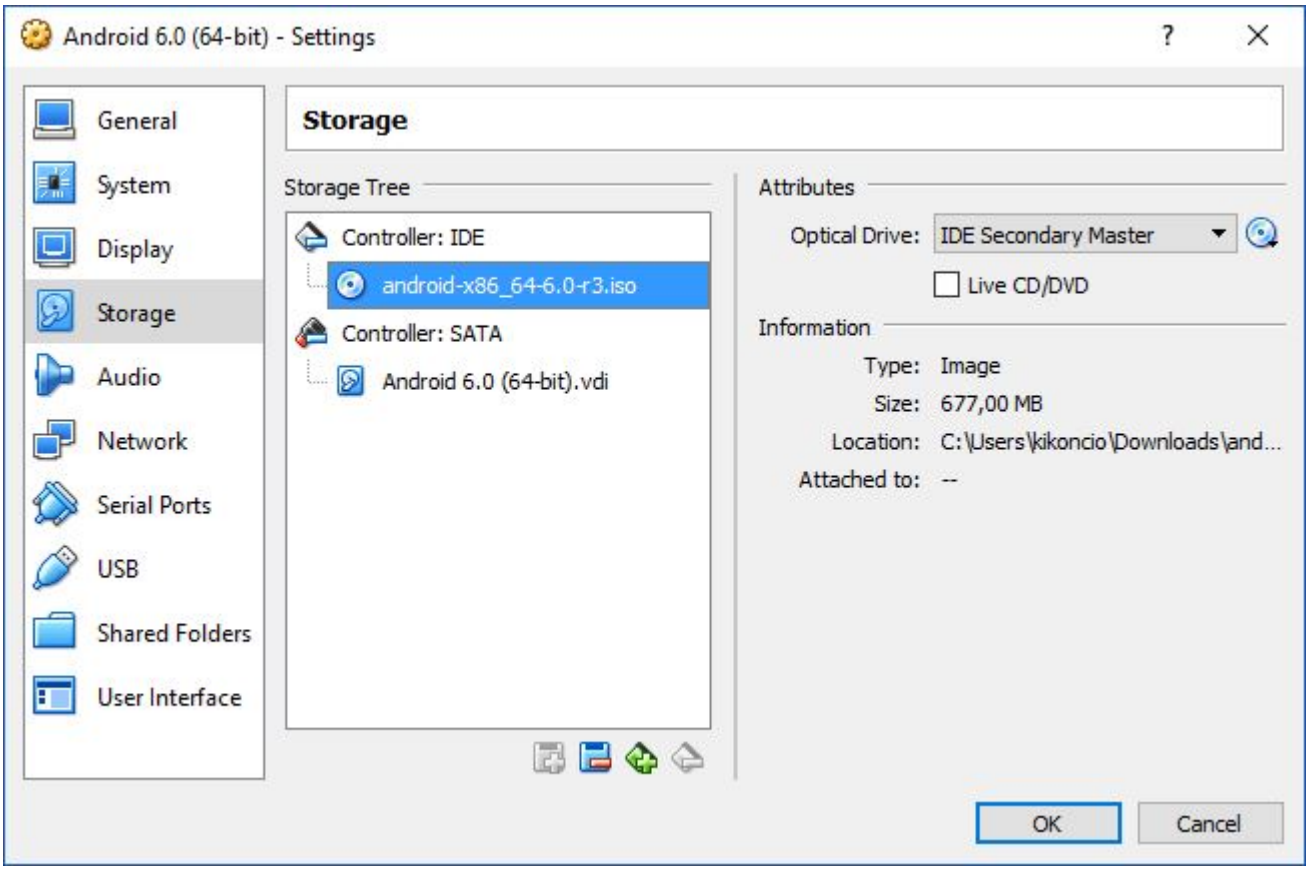
Dies sind meine VirtualBox-Einstellungen:

- Betriebssystemtyp: Linux 2.6 (Ich habe Benutzer 64bit, da mein Computer dies unterstützen kann)
- Größe der virtuellen Festplatte: 4 GB
- RAM-Speicher: 2048
- Videospeicher: 8M
- Soundgerät: Sound Blaster 16.
- Netzwerkgerät: PCnet-Fast III, an NAT angeschlossen. Sie können auch überbrückte Adapter verwenden, aber Sie benötigen einen DHCP-Server in Ihrer Umgebung.

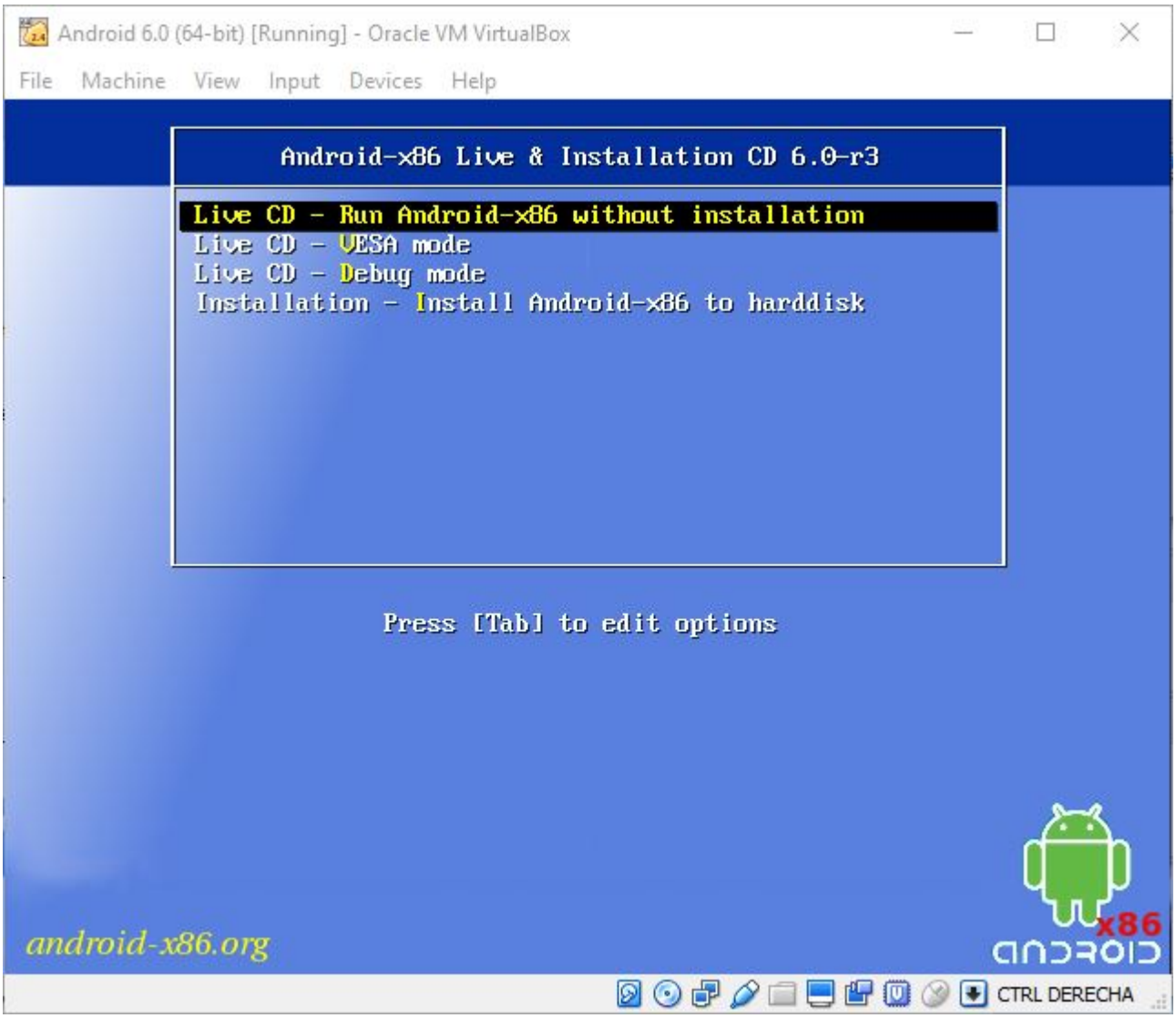
Das mit dieser Konfiguration verwendete Bild wurde android-x86_64-6.0-r3.iso (64 Bit) von http://www.android-x86.org/download heruntergeladen. Ich nehme an, dass es auch mit der 32bit-Version funktioniert.

Einrichten der virtuellen Festplatte für die SDCARD-Unterstützung

Starten Sie mit der soeben erstellten virtuellen Festplatte die virtuelle Maschine mit dem Android-x86-Image im optischen Laufwerk.



Nach dem Booten sehen Sie das Grub-Menü der Live-CD



Wählen Sie die Debug-Modusoption aus, dann sollte die Shell-Eingabeaufforderung angezeigt werden. Dies ist eine busybox-Shell. Sie können mehr Shell erhalten, indem Sie zwischen der virtuellen Konsole Alt-F1 / F2 / F3 wechseln.

Erstellen Sie zwei Partitionen von fdisk (einige andere Versionen würden cfdisk verwenden). Formatieren Sie sie in ext3. Dann neustarten:

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

Dann tippe:

"n" (neue Partition)

"p" (primäre Partition)

"1" (1. Partition)

"1" (erster Zylinder)

"261" (Wählen Sie einen Zylinder, wir belassen 50% der Festplatte für eine zweite Partition)

"2" (2. Partition)

"262" (262. Zylinder)

"522" (wähle den letzten Zylinder)

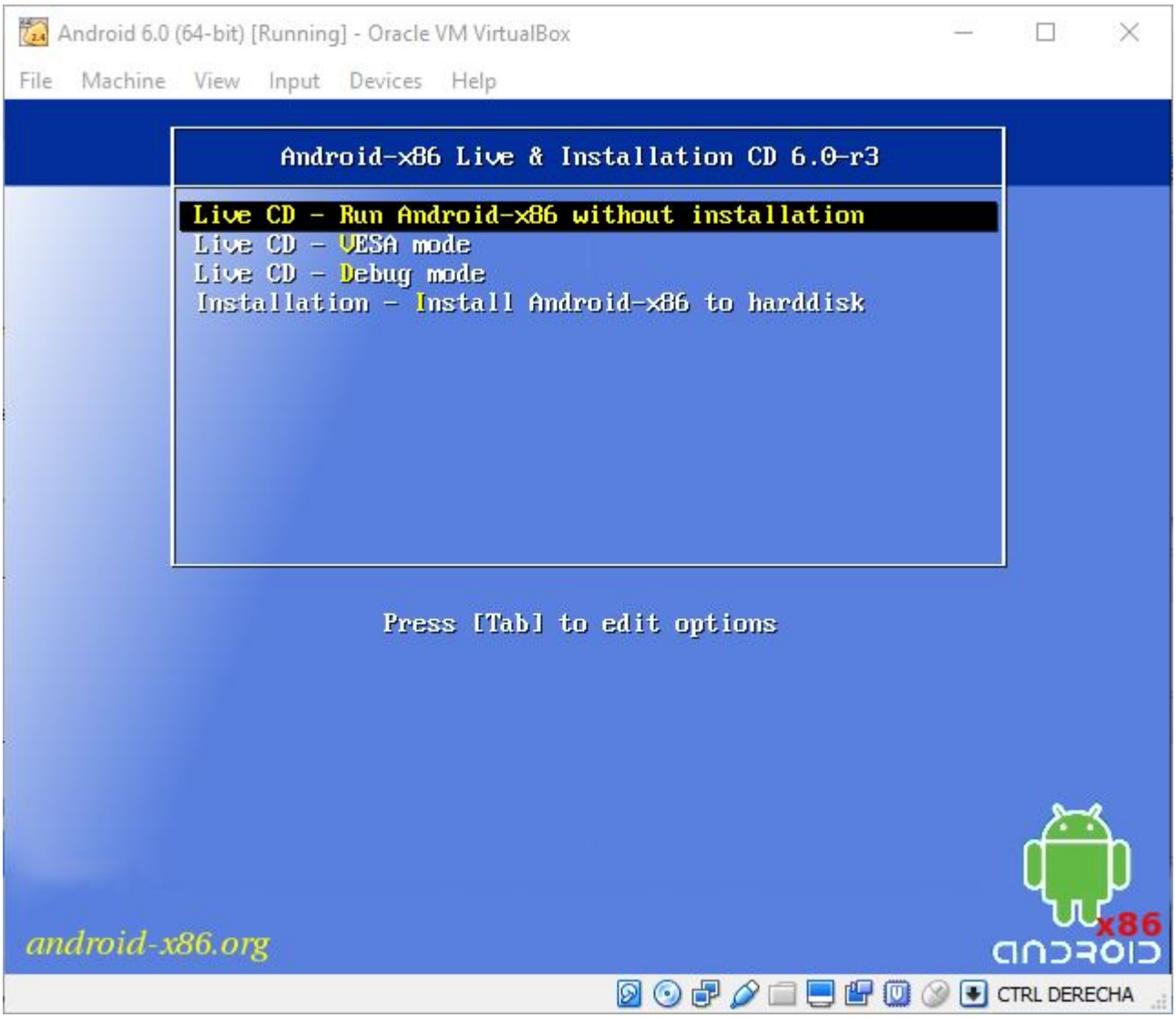
"w" (Schreiben Sie die Partition)

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

Wenn Sie die virtuelle Maschine neu starten und das Grub-Menü angezeigt wird, können Sie die Kernel- `DATA=sda1 SDCARD=sda2` bearbeiten, sodass Sie die Optionen `DATA=sda1 SDCARD=sda2` können, die auf die SD-Karte oder die Datenpartition verweisen.

Installation in der Partition

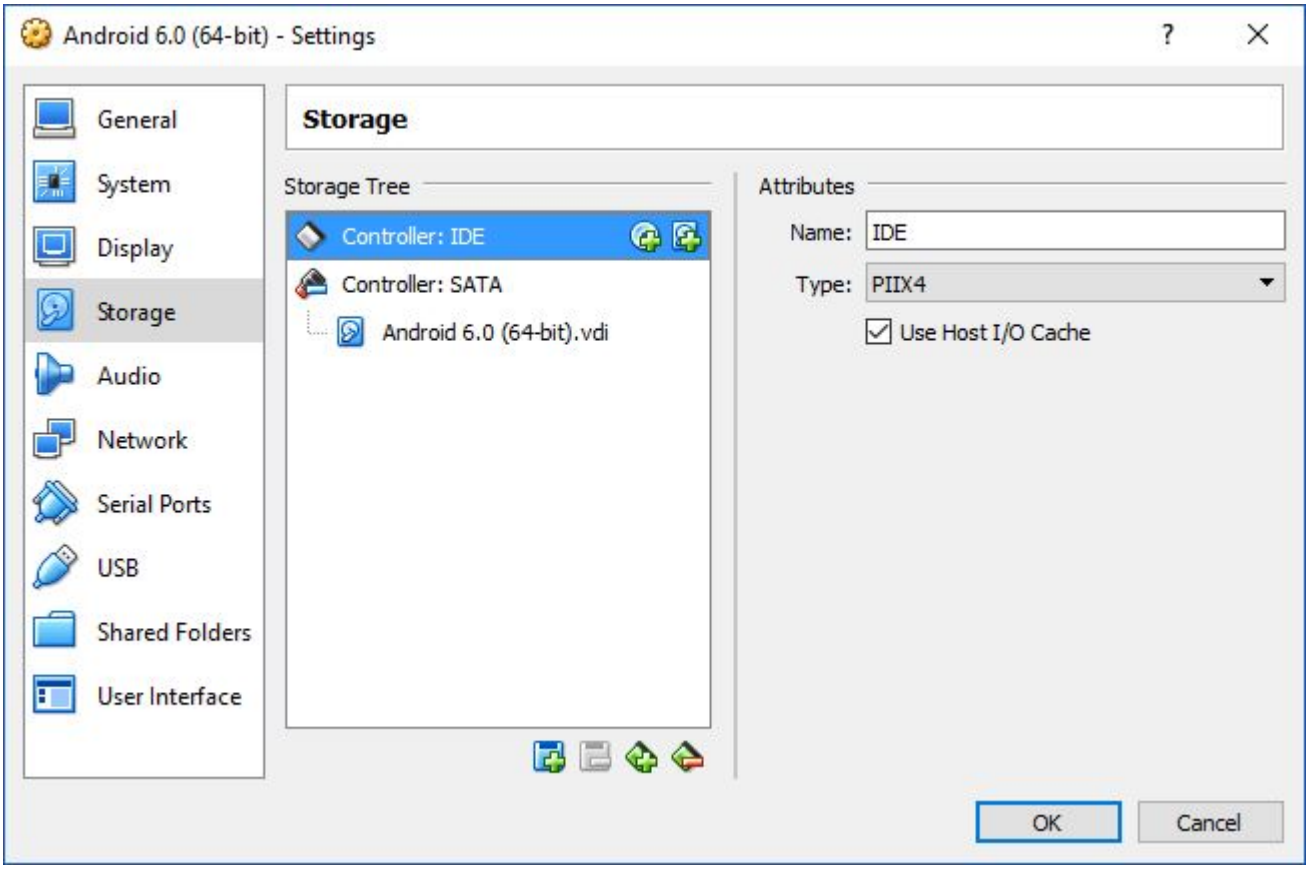
Starten Sie mit der soeben erstellten virtuellen Festplatte die virtuelle Maschine mit dem Android-x86-Image als optisches Laufwerk.



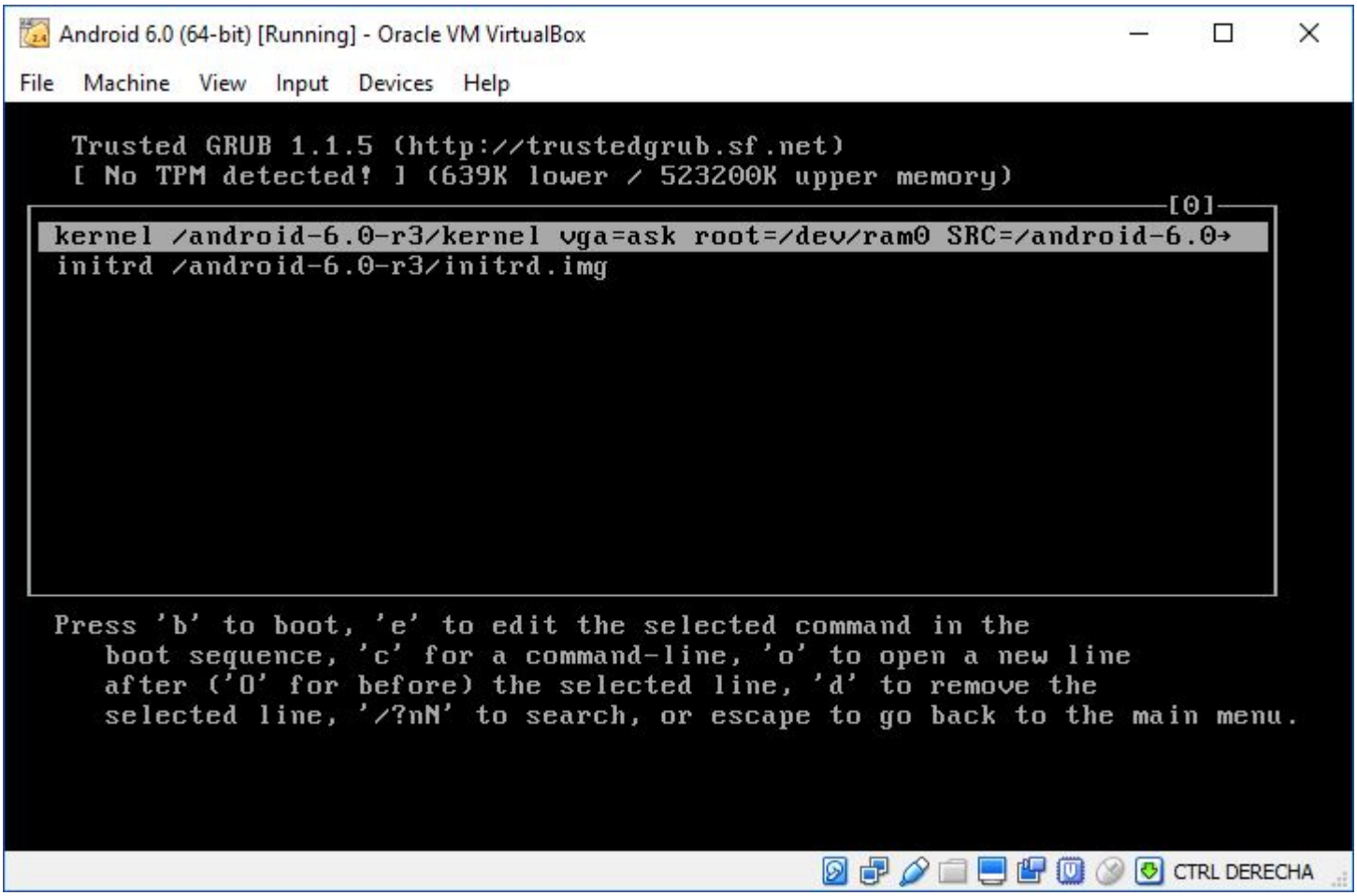
Wählen Sie in den Boot-Optionen der Live-CD "Installation - Android auf Festplatte installieren".

Wähle die sda1-Partition und installiere android. Wir installieren grub.

Starten Sie die virtuelle Maschine neu, stellen Sie jedoch sicher, dass sich das Image nicht im optischen Laufwerk befindet, damit es von der virtuellen Festplatte aus gestartet werden kann.



Im Grub-Menü müssen wir den Kernel wie in der Option "Android-x86 6.0-r3" bearbeiten. Drücken Sie also die Taste e.

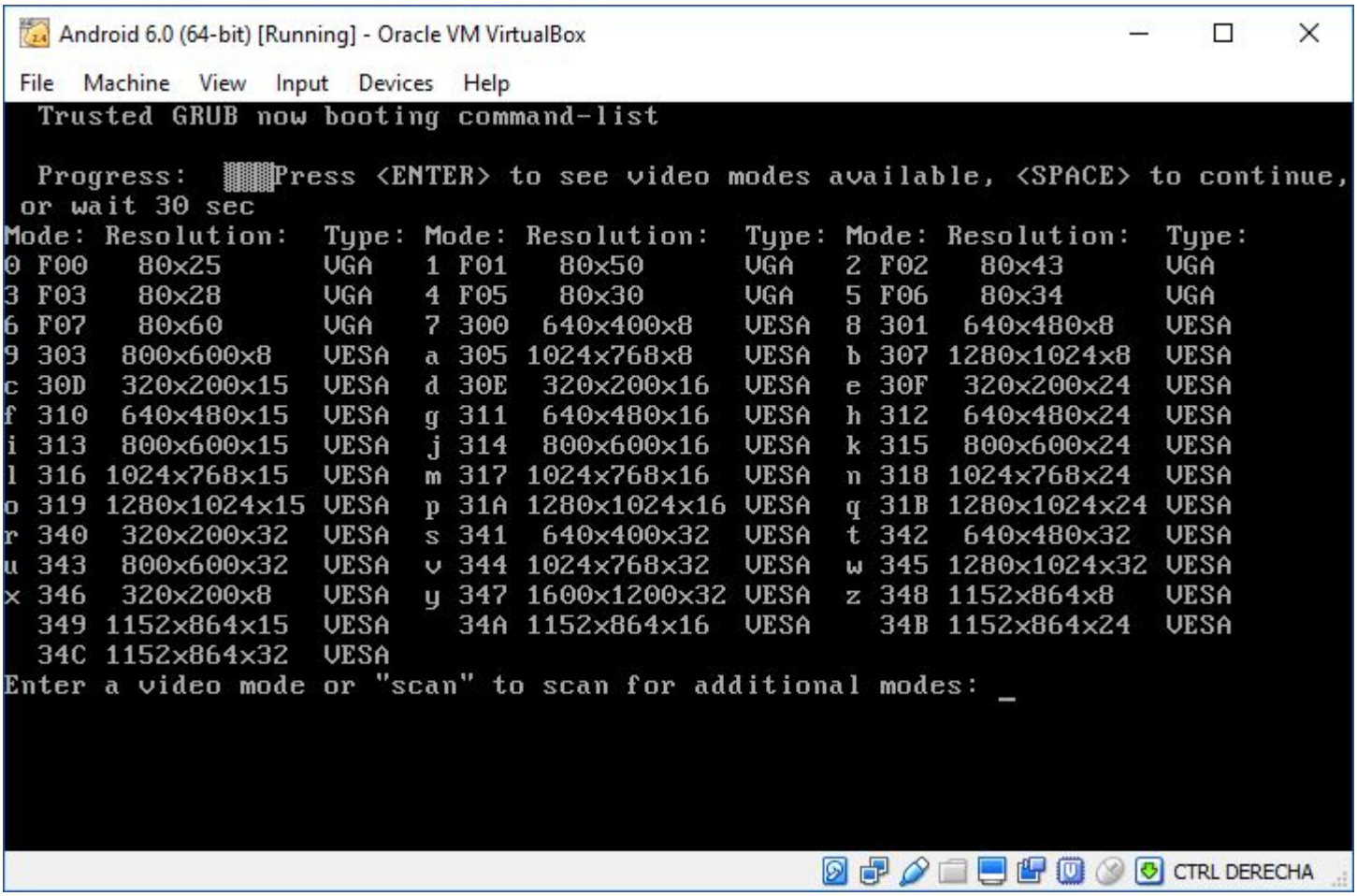


Dann ersetzen wir "leise" durch "vga = ask" und fügen die Option "SDCARD = sda2" hinzu.

In meinem Fall sieht die Kernlinie nach der Änderung folgendermaßen aus:

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

Drücken Sie b, um zu booten. Dann können Sie die Bildschirmgröße mit ENTER (`vga=ask`)
`vga=ask`



Nachdem der Installationsassistent gestartet wurde, wählen Sie die Sprache aus. Ich konnte Englisch (USA) und Spanisch (USA) wählen und hatte Schwierigkeiten, andere zu wählen.

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

Kapitel 28: Animatoren

Examples

Animation einer ImageView schütteln

Erstellen Sie im Ordner res einen neuen Ordner mit dem Namen "anim", um Ihre Animationsressourcen zu speichern, und legen Sie diesen in diesem Ordner ab.

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" />
```

Erstellen Sie eine leere Aktivität namens Landing

activity_landing.xml

```
  <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ImageView
            android:id="@+id/imgBell"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:src="@mipmap/ic_notifications_white_48dp"/>

    </RelativeLayout>
```

Und die Methode zum Animieren der Bildansicht auf Landing.java

```
Context mContext;


 @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            mContext=this;
                setContentView(R.layout.activity_landing);

                AnimateBell();
        }

public void AnimateBell() {
```

```
        Animation shake = AnimationUtils.loadAnimation(mContext, R.anim.shakeanimation);
        ImageView imgBell= (ImageView) findViewById(R.id.imgBell);
        imgBell.setImageResource(R.mipmap.ic_notifications_active_white_48dp);
        imgBell.setAnimation(shake);
    }
```

Animation ein- / ausblenden

Verwenden Sie einen `ObjectAnimator` , um eine Ansicht zu erhalten, die langsam `ObjectAnimator` oder `ObjectAnimator` . Wie im folgenden Code dargestellt, legen Sie eine Dauer mit `.setDuration(millis)` wobei der `millis` Parameter die Dauer (in Millisekunden) der Animation ist. Im folgenden Code werden die Ansichten über 500 Millisekunden oder 1/2 Sekunde ein- / ausgeblendet. Um die `ObjectAnimator` -Animation zu starten, rufen Sie `.start()` . Sobald die Animation abgeschlossen ist, wird `onAnimationEnd(Animator animation)` aufgerufen. Hier können Sie die Sichtbarkeit Ihrer Ansicht auf `View.GONE` oder `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-Animation

In diesem Beispiel wird eine Transaktion für eine Bildansicht mit nur zwei Bildern angezeigt (nach

der Transaktion können nacheinander mehrere Bilder für die Positionen der ersten und zweiten Ebene als Schleife verwendet werden.)

- `res/values/arrays.xml` zu `res/values/arrays.xml`

```
<resources>

    <array
        name="splash_images">
        <item>@drawable/spash_imge_first</item>
        <item>@drawable/spash_img_second</item>
    </array>

</resources>
```

```
    private Drawable[] backgroundsDrawableArrayForTransition;
    private TransitionDrawable transitionDrawable;

private void backgroundAnimTransAction() {

        // set res image array
        Resources resources = getResources();
        TypedArray icons = resources.obtainTypedArray(R.array.splash_images);

        @SuppressWarnings("ResourceType")
        Drawable drawable = icons.getDrawable(0);    // ending image

        @SuppressWarnings("ResourceType")
        Drawable drawableTwo = icons.getDrawable(1);   // starting image


        backgroundsDrawableArrayForTransition = new Drawable[2];
        backgroundsDrawableArrayForTransition[0] = drawable;
        backgroundsDrawableArrayForTransition[1] = drawableTwo;
        transitionDrawable = new TransitionDrawable(backgroundsDrawableArrayForTransition);

        // your image view here - backgroundImageView
        backgroundImageView.setImageDrawable(transitionDrawable);
        transitionDrawable.startTransition(4000);

        transitionDrawable.setCrossFadeEnabled(false); // call public methods


    }
```

ValueAnimator

`ValueAnimator` einfache Weise einen Wert (eines bestimmten Typs, z. B. `int` , `float` usw.) animieren.

Die übliche Art, es zu benutzen, ist:

1. Erstellen Sie einen `ValueAnimator` , der einen Wert von `min` bis `max` animiert
2. Fügen Sie einen `UpdateListener` in dem Sie den berechneten animierten Wert verwenden (den Sie mit `getAnimatedValue()`).

Es gibt zwei Möglichkeiten, den `ValueAnimator` erstellen:

(das Beispiel Code animiert einen `float` von `20f` bis `40f` in `250ms`)

1. Von `xml` (füge es in `/res/animator/`):

```
<animator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="250"
    android:valueFrom="20"
    android:valueTo="40"
    android:valueType="floatType"/>
```

```
ValueAnimator animator = (ValueAnimator) AnimatorInflater.loadAnimator(context,
        R.animator.example_animator);
animator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator anim) {
        // ... use the anim.getAnimatedValue()
    }
});
// set all the other animation-related stuff you want (interpolator etc.)
animator.start();
```

2. Aus dem Code:

```
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` ist eine Unterklasse von `ValueAnimator` mit der zusätzlichen Möglichkeit , den berechneten Wert auf die Eigenschaft eines festlegen `target` .

Genau wie beim `ValueAnimator` gibt es zwei Möglichkeiten, den `ObjectAnimator` erstellen:

(Der Beispielcode animiert ein `alpha` einer `View` von `0.4f` bis `0.2f` in `250ms`)

1. Von `xml` (in den `/res/animator`)

```
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
                android:duration="250"
                android:propertyName="alpha"
                android:valueFrom="0.4"
                android:valueTo="0.2"
                android:valueType="floatType"/>
```

```
ObjectAnimator animator = (ObjectAnimator) AnimatorInflater.loadAnimator(context,
        R.animator.example_animator);
animator.setTarget(exampleView);
// set all the animation-related stuff you want (interpolator etc.)
animator.start();
```

2. Vom Code:

```
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` ist eine vereinfachte und optimierte Methode zum Animieren von Eigenschaften einer `View` .

Für jede einzelne `View` ist ein `ViewPropertyAnimator` Objekt über die `animate()` -Methode verfügbar. Sie können damit mehrere Eigenschaften gleichzeitig mit einem einfachen Aufruf animieren. Jede einzelne Methode eines `ViewPropertyAnimator` gibt den **Zielwert** eines bestimmten Parameters an, den der `ViewPropertyAnimator` animieren soll.

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

Hinweis: Der Aufruf von `start()` für ein `ViewPropertyAnimator` Objekt ist **NICHT** obligatorisch. Wenn Sie dies nicht tun, lassen Sie die Plattform den Start der Animation zum richtigen Zeitpunkt erledigen (nächster Animations-Durchlauf). Wenn Sie das tatsächlich tun (call `start()`), stellen Sie sicher, dass die Animation sofort gestartet wird.

Erweitern und Reduzieren Sie die Animation der Ansicht

```
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);
        }
    }
```

Animatoren online lesen: https://riptutorial.com/de/android/topic/1829/animatoren

Kapitel 29: Animiertes AlertDialogfeld

Einführung

Animiertes Dialogfeld "Warnungen" Welche Anzeige mit einigen Animationseffekten angezeigt wird. Sie können Animationen für Dialogfelder erhalten, wie etwa Fadein, Slideleft, Slidetop, SlideBottom, Sliderright, Fall, Newspager, Fliph, Flipv, RotateBottom, RotateLeft, Slit, Shake, Sidefill Anwendung attraktiv ..

Examples

Fügen Sie den folgenden Code für den animierten Dialog ein ...

```
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();
    }
}
```

Fügen Sie die folgenden Zeilen in Ihr build.gradle ein, um den NifyBuilder (CustomView) aufzunehmen

build.gradle

```
 dependencies {

     compile 'com.nineoldandroids:library:2.4.0'

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

 }
```

Referenzlink: https://github.com/sd6352051/NiftyDialogEffects

Animiertes AlertDialogfeld online lesen: https://riptutorial.com/de/android/topic/10654/animiertes-alertdialogfeld

Kapitel 30: Anmerkungsprozessor

Einführung

Anmerkungsprozessor ist ein in javac eingebautes Werkzeug zum Scannen und Bearbeiten von Anmerkungen zur Kompilierzeit.

Annotationen sind eine Klasse von Metadaten, die Klassen, Methoden, Feldern und sogar anderen Annotationen zugeordnet werden können. Es gibt zwei Möglichkeiten, auf diese Annotationen zur Laufzeit über Reflection und zur Kompilierungszeit über Annotationsprozessoren zuzugreifen.

Examples

@NonNull-Anmerkung

```
public class Foo {
    private String name;
    public Foo(@NonNull String name){...};
    ...
}
```

Hier ist @NonNull eine Anmerkung, die vom Android-Studio zur Kompilierzeit verarbeitet wird, um Sie darauf hinzuweisen, dass die jeweilige Funktion einen Nicht-Null-Parameter benötigt.

Arten von Anmerkungen

Es gibt drei Arten von Anmerkungen.

1. **Markierungsanmerkung** - Anmerkung, die keine Methode hat

   ```
   @interface CustomAnnotation {}
   ```

2. **Einzelwert-Annotation** - Annotation mit einer Methode

   ```
   @interface CustomAnnotation {
       int value();
   }
   ```

3. **Multi-Value-Annotation** - Annotation mit mehr als einer Methode

   ```
   @interface CustomAnnotation{
       int value1();
       String value2();
       String value3();
   }
   ```

Erstellen und Verwenden von benutzerdefinierten Anmerkungen

Um benutzerdefinierte Anmerkungen zu erstellen, müssen wir entscheiden

- Ziel - für das diese Anmerkungen auf Feldebene, Methodenebene, Typebene usw. funktionieren.
- Aufbewahrung - auf welcher Ebene wird die Annotation verfügbar sein.

Dafür haben wir benutzerdefinierte Annotationen eingebaut. Schauen Sie sich diese meist verwendeten an:

@Ziel

Element Types	Where the a
TYPE	class, interfac
FIELD	fields
METHOD	methods
CONSTRUCTOR	constructors
LOCAL_VARIABLE	local variable
ANNOTATION_TYPE	annotation ty
PARAMETER	parameter

@Retention

RetentionPolicy	Availability
RetentionPolicy.SOURCE	refers to the source code, d class.
RetentionPolicy.CLASS	refers to the .class file, ava file.
RetentionPolicy.RUNTIME	refers to the runtime, availa

Benutzerdefinierte Annotation erstellen

```
@Retention(RetentionPolicy.SOURCE) // will not be available in compiled class
@Target(ElementType.METHOD) // can be applied to methods only
@interface CustomAnnotation{
      int value();
}
```

Benutzerdefinierte Annotation verwenden

```
class Foo{
  @CustomAnnotation(value = 1)  // will be used by an annotation processor
  public void foo(){..}
}
```

Der in `@CustomAnnotation` bereitgestellte `@CustomAnnotation` wird von einem Anmerkungsprozessor verwendet, um Code zur Kompilierungszeit usw. zu generieren.

Anmerkungsprozessor online lesen:
https://riptutorial.com/de/android/topic/10726/anmerkungsprozessor

Kapitel 31: Anzeigen von Google-Anzeigen

Examples

Grundlegendes Anzeigen-Setup

Sie müssen den Abhängigkeiten Folgendes hinzufügen:

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

und dann in die gleiche Datei.

```
apply plugin: 'com.google.gms.google-services'
```

Als Nächstes müssen Sie relevante Informationen in Ihre strings.xml einfügen.

```
<string name="banner_ad_unit_id">ca-app-pub-####/####</string>
```

Platzieren Sie als Nächstes eine Vorschau, wo immer Sie möchten, und gestalten Sie sie wie jede andere Ansicht.

```
    <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>
```

Und last but not least, werfen Sie dies in Ihren onCreate.

```
MobileAds.initialize(getApplicationContext(), "ca-app-pub-YOUR_ID");
AdView mAdView = (AdView) findViewById(R.id.adView);
    AdRequest adRequest = new AdRequest.Builder().build();
    mAdView.loadAd(adRequest);
```

Wenn Sie genau kopiert haben, sollten Sie jetzt eine kleine Banneranzeige haben. Platzieren Sie einfach mehr AdViews dort, wo Sie sie benötigen.

Hinzufügen von Interstitial-Anzeigen

Interstitial-Anzeigen sind Vollbild-Anzeigen, die die Benutzeroberfläche ihrer Host-App abdecken. Sie werden normalerweise an natürlichen Übergangspunkten im Ablauf einer App angezeigt, z. B. zwischen Aktivitäten oder während der Pause zwischen den Ebenen eines Spiels.

Stellen Sie sicher, dass Sie über die erforderlichen Berechtigungen in Ihrer `Manifest` Datei verfügen:

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

1. Wechseln Sie zu Ihrem AdMob- Konto.
2. Klicken Sie auf **die** Registerkarte **Geld verdienen** .
3. Wählen Sie oder Erstellen Sie die App und wählen Sie die Plattform aus.
4. Wählen Sie Interstitial aus und geben Sie einen Anzeigenblocknamen an.
5. Nachdem der Anzeigenblock erstellt wurde, können Sie die Anzeigenblock-ID im Dashboard feststellen. Zum Beispiel: ca-app-pub-00000000000/000000000
6. Abhängigkeiten hinzufügen

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

Dieser sollte unten sein.

```
apply plugin: 'com.google.gms.google-services'
```

Fügen Sie die **ID** Ihres **Anzeigenblocks** zu Ihrer Datei `strings.xml`

```
<string name="interstitial_full_screen">ca-app-pub-00000000/00000000</string>
```

Fügen Sie Ihrem Manifest ConfigChanges und Metadaten hinzu:

```
<activity
            android:name="com.google.android.gms.ads.AdActivity"

android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScree

            android:theme="@android:style/Theme.Translucent" />
```

und

```
<meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
```

Aktivität:

```
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();
        }
    }

}
```

Diese AdActivity zeigt jetzt eine Vollbildanzeige.

Anzeigen von Google-Anzeigen online lesen:
https://riptutorial.com/de/android/topic/5984/anzeigen-von-google-anzeigen

Kapitel 32: Apps mit ADB installieren

Examples

App installieren

Schreiben Sie den folgenden Befehl in Ihr Terminal:

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

Beachten Sie, dass Sie eine Datei übergeben müssen, die sich auf Ihrem Computer und nicht auf Ihrem Gerät befindet.

Wenn Sie am Ende `-r` anhängen, werden alle vorhandenen widersprüchlichen Apks überschrieben. Andernfalls wird der Befehl mit einem Fehler beendet.

`-g` gewährt sofort alle Laufzeitberechtigungen.

`-d` erlaubt das Herabstufen des Versionscodes (nur anwendbar bei debugierbaren Paketen).

Verwenden Sie `-s` , um die Anwendung auf der externen SD-Karte zu installieren.

`-t` erlaubt die Verwendung von Testanwendungen.

App deinstallieren

Schreiben Sie den folgenden Befehl in Ihr Terminal, um eine App mit dem angegebenen Paketnamen zu deinstallieren:

```
adb uninstall <packagename>
```

Installieren Sie alle APK-Dateien im Verzeichnis

Windows:

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

Linux:

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

Apps mit ADB installieren online lesen: https://riptutorial.com/de/android/topic/5301/apps-mit-adb-installieren

Kapitel 33: Arbeit planen

Bemerkungen

Hüten Sie sich vor dem Ausführen von viel Code oder dem Ausführen schwerer Arbeit in Ihrem `JobService` , beispielsweise in `onStartJob()` . Der Code wird auf dem **Haupt- / UI-** Thread ausgeführt und kann daher zu einer blockierten Benutzeroberfläche, nicht mehr reagierenden Apps oder sogar zum Absturz Ihrer App führen!

Aus diesem Grund müssen Sie die Arbeit `AsyncTask` , z. B. mithilfe eines `Thread` oder einer `AsyncTask` .

Examples

Grundlegende Verwendung

Erstellen Sie einen neuen JobService

Dies erfolgt durch Erweitern der `JobService` Klasse und Implementieren / Überschreiben der erforderlichen Methoden `onStartJob()` und `onStopJob()` .

```
public class MyJobService extends JobService
{
    final String TAG = getClass().getSimpleName();

    @Override
    public boolean onStartJob(JobParameters jobParameters) {
        Log.i(TAG, "Job started");

        // ... your code here ...

        jobFinished(jobParameters, false);  // signal that we're done and don't want to
reschedule the job
        return false;                       // finished: no more work to be done
    }

    @Override
    public boolean onStopJob(JobParameters jobParameters) {
        Log.w(TAG, "Job stopped");
        return false;
    }
}
```

Fügen Sie den neuen JobService Ihrer AndroidManifest.xml hinzu

Der folgende Schritt ist *obligatorisch* , andernfalls können Sie Ihren Job nicht ausführen:

Deklarieren Sie Ihre `MyJobService` Klasse als neues `<service>` -Element zwischen `<application>` `</application>` in Ihrer *AndroidManifest.xml* .

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service
            android:name=".MyJobService"
            android:permission="android.permission.BIND_JOB_SERVICE" />
    </application>
</manifest>
```

Richten Sie den Job ein und führen Sie ihn aus

Nachdem Sie einen neuen JobService implementiert und zu Ihrer *AndroidManifest.xml* hinzugefügt *haben* , können Sie mit den letzten Schritten fortfahren.

- `onButtonClick_startJob()` bereitet einen periodischen Job vor und führt ihn aus. Mit `JobInfo.Builder` können neben periodischen Jobs viele andere Einstellungen und Einschränkungen festgelegt werden. Sie können beispielsweise festlegen, dass ein angeschlossenes *Ladegerät* oder eine *Netzwerkverbindung* erforderlich ist, um den Job auszuführen.
- `onButtonClick_stopJob()` bricht alle laufenden Jobs ab

```
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
    }
}
```

Nach dem Aufruf von `onButtonClick_startJob()` wird der Job ungefähr in Intervallen von 10 Sekunden ausgeführt, auch wenn sich die App im *angehaltenen* Status befindet (der Benutzer hat die Home-Taste gedrückt und die App ist nicht mehr sichtbar).

Anstatt alle laufenden Jobs in `onButtonClick_stopJob()` , können Sie auch `jobScheduler.cancel()` aufrufen, um einen bestimmten Job basierend auf seiner Job-ID abzubrechen.

Arbeit planen online lesen: https://riptutorial.com/de/android/topic/6907/arbeit-planen

Kapitel 34: AsyncTask

Parameter

Parameter	Einzelheiten
Params	die Art der Parameter, die bei der Ausführung an die Task gesendet werden.
Fortschritt	die Art der Fortschrittseinheiten, die während der Hintergrundberechnung veröffentlicht wurden
Ergebnis	die Art des Ergebnisses der Hintergrundberechnung.

Examples

Grundlegende Verwendung

In Android Aktivitäten und Dienstleistungen sind die meisten Rückrufe auf der Flucht Hauptthread . Dies macht es einfach, die Benutzeroberfläche zu aktualisieren, aber das Ausführen von prozessor- oder E / A-schweren Tasks im Haupt-Thread kann dazu führen, dass Ihre Benutzeroberfläche angehalten wird und nicht mehr reagiert (offizielle Dokumentation dazu, was dann passiert).

Sie können dies beheben, indem Sie diese schwereren Aufgaben in einem Hintergrundthread ablegen.

Eine Möglichkeit hierzu ist die Verwendung einer AsyncTask , die ein Framework bereitstellt , das die einfache Verwendung eines Hintergrund-Threads erleichtert und UI-Thread-Tasks vor, während und nach Abschluss der Arbeit des Hintergrund-Threads ausführt.

Methoden, die beim Erweitern von `AsyncTask` überschrieben werden `AsyncTask` :

- `onPreExecute()` : `onPreExecute()` im **UI-Thread** aufgerufen, bevor die Task ausgeführt wird
- `doInBackground()` : auf **dem Hintergrund** - **Thread** aufgerufen unmittelbar nach `onPreExecute()` beendet die Ausführung.
- `onProgressUpdate()` : `onProgressUpdate()` nach einem Aufruf von `publishProgress(Progress...)` im **UI-Thread** aufgerufen.
- `onPostExecute()` : `onPostExecute()` im **UI-Thread** aufgerufen, nachdem die Hintergrundberechnung abgeschlossen ist

Beispiel

```
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.");
        }
    }

}
```

Verwendungszweck:

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

oder einfach:

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

Hinweis

Bei der Definition einer `AsyncTask` wir drei Typen zwischen `< >` Klammern übergeben.
Definiert als `<Params, Progress, Result>` (siehe **Abschnitt Parameter**)

Im vorherigen Beispiel haben wir die Typen `<File, Void, String>` :

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

`Void` wird verwendet, wenn Sie einen Typ als nicht verwendet markieren möchten.

Beachten Sie, dass Sie keine primitiven Typen (dh `int` , `float` und 6 andere) als Parameter übergeben können. In solchen Fällen sollten Sie ihre Wrapper-Klassen übergeben , z. B. `Integer` statt `int` oder `Float` statt `float` .

Der AsyncTask- und Activity-Lebenszyklus

AsyncTasks folgen nicht dem Lebenszyklus der Aktivitätsinstanzen. Wenn Sie eine AsyncTask in einer Aktivität starten und das Gerät drehen, wird die Aktivität zerstört und eine neue Instanz erstellt. Aber die AsyncTask wird nicht sterben. Es wird weiterleben, bis es fertig ist.

Lösung: AsyncTaskLoader

Eine Unterklasse von Loadern ist der AsyncTaskLoader. Diese Klasse hat die gleiche Funktion wie die AsyncTask, jedoch viel besser. Es kann Aktivitätskonfigurationsänderungen einfacher handhaben und verhält sich innerhalb der Lebenszyklen von Fragmenten und Aktivitäten. Das Schöne ist, dass der AsyncTaskLoader in jeder Situation verwendet werden kann, in der die AsyncTask verwendet wird. Immer wenn Daten in den Speicher geladen werden müssen, damit die Aktivität / das Fragment verarbeitet werden kann, kann der AsyncTaskLoader die Aufgabe besser erfüllen.

AsyncTask abbrechen

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

Dies stoppt Ihre Aufgabe nicht, wenn sie gerade ausgeführt wird. Sie setzt lediglich das Flag " `isCancelled()` das überprüft werden kann, indem der Rückgabewert von `isCancelled()` überprüft wird (vorausgesetzt, Ihr Code läuft derzeit).

```
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;
            }
        }
    }
}
```

Hinweis

Wenn eine AsyncTask abgebrochen wird, während noch `doInBackground(Params... params)` wird, wird die Methode `onPostExecute(Result result)` **NICHT** nach `doInBackground(Params... params)` . Die AsyncTask ruft stattdessen `onCancelled(Result result)` auf, um anzuzeigen, dass die Aufgabe während der Ausführung abgebrochen wurde.

Veröffentlichungsfortschritt

Manchmal müssen wir den Fortschritt der von einer `AsyncTask` Berechnung `AsyncTask` . Dieser Fortschritt kann durch eine Zeichenfolge, eine Ganzzahl usw. dargestellt werden. Dazu müssen wir zwei Funktionen verwenden. Zuerst müssen wir die `onProgressUpdate` Funktion `onProgressUpdate` , deren Parametertyp dem zweiten Typparameter unserer `AsyncTask` .

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

Zweitens müssen wir die Funktion `publishProgress` notwendigerweise auf der `doInBackground` Funktion, und das ist alles, tun die vorherige Methode alle den Job.

```
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;
 }
```

Laden Sie das Bild mit AsyncTask in Android herunter

In diesem Lernprogramm wird erläutert, wie Sie Image mit AsyncTask in Android herunterladen. Das Beispiel unten zeigt ein Bild zum Herunterladen, während der Download während des Fortschritts angezeigt wird.

Grundlegendes zu Android AsyncTask

Mit der asynchronen Task können Sie MultiThreading implementieren, ohne die Hände in Threads zu verschmutzen. AsyncTask ermöglicht die ordnungsgemäße und einfache Verwendung des UI-Threads. Es ermöglicht Hintergrundoperationen und die Weitergabe der Ergebnisse an den UI-Thread. Wenn Sie etwas Isoliertes in Bezug auf die Benutzeroberfläche tun, beispielsweise Daten herunterladen, um sie in einer Liste anzuzeigen, fahren Sie fort und verwenden Sie AsyncTask.

- AsyncTasks sollten idealerweise für kurze Operationen (höchstens einige Sekunden) verwendet werden.
- Eine asynchrone Task wird durch 3 generische Typen definiert, die als Params, Progress und Result bezeichnet werden, sowie 4 Schritte, die als `onPreExecute()` , `doInBackground()` , `onProgressUpdate()` und `onPostExecute()` .
- In `onPreExecute()` Sie Code definieren, der ausgeführt werden muss, bevor die Hintergrundverarbeitung beginnt.
- doInBackground enthält Code, der im Hintergrund ausgeführt werden muss. Hier in `doInBackground()` wir die Ergebnisse mit der Methode publishProgress () mehrmals an den Ereignisthread senden. Wenn die Hintergrundverarbeitung abgeschlossen ist, können wir die Ergebnisse einfach zurückgeben.
- `onProgressUpdate()` Methode empfängt Fortschrittsaktualisierungen von der Methode `doInBackground()` , die über die `publishProgress()` Methode veröffentlicht wird. Diese Methode kann diese Fortschrittsaktualisierung zum Aktualisieren des Ereignisthreads verwenden
- `onPostExecute()` -Methode verarbeitet Ergebnisse, die von der `doInBackground()` -Methode zurückgegeben werden.
- Die generischen Typen sind
    - Params, der Typ der Parameter, die bei der Ausführung an die Task gesendet werden
    - Fortschritt, der Typ der Fortschrittseinheiten, die während der Hintergrundberechnung veröffentlicht wurden.
    - Result ist die Art des Ergebnisses der Hintergrundberechnung.
- Wenn ein asynchroner Task keine Typen verwendet, kann er als ungültiger Typ markiert werden.
- Eine laufende async-Task kann abgebrochen werden, indem `cancel(boolean)` Methode `cancel(boolean)` wird.

Bild mit Android AsyncTask herunterladen

Ihr XML-Layout

```
<?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-Klasse

```
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;
        }
    }
}
```

Da es derzeit kein Kommentarfeld für Beispiele gibt (oder ich habe es nicht gefunden oder ich habe keine Erlaubnis dafür), hier ein Kommentar dazu:

Dies ist ein gutes Beispiel für die Verwendung von AsyncTask.

Allerdings hat das Beispiel derzeit Probleme mit

- mögliche Speicherlecks
- App stürzt ab, wenn kurz vor Beendigung der asynchronen Task eine Bildschirmrotation stattgefunden hat.

Für Details siehe:

- Übergeben Sie die Aktivität als WeakReference, um Speicherverluste zu vermeiden
- http://stackoverflow.com/documentation/android/117/asynctask/5377/possible-problems-mit-inner-async-tasks
- Vermeiden Sie undichte Aktivitäten mit AsyncTask

Übergeben Sie die Aktivität als WeakReference, um Speicherverluste zu vermeiden

Es ist üblich, dass eine AsyncTask einen Verweis auf die Aktivität erfordert, die sie aufgerufen hat.

Wenn die AsyncTask eine innere Klasse der Aktivität ist, können Sie direkt auf sie und alle Member-Variablen / -Methoden verweisen.

Wenn die AsyncTask jedoch keine innere Klasse der Aktivität ist, müssen Sie eine Aktivitätsreferenz an die AsyncTask übergeben. Wenn Sie dies tun, besteht ein mögliches Problem möglicherweise darin, dass die AsyncTask den Verweis der Aktivität behält, bis die AsyncTask ihre Arbeit in ihrem Hintergrundthread abgeschlossen hat. Wenn die Aktivität beendet oder beendet wird, bevor die Hintergrundarbeit des AsyncTask-Threads abgeschlossen ist, hat die AsyncTask immer noch ihren Verweis auf die Aktivität. Daher kann keine Sammlung von Speicherbereinigungen erfolgen.

Dies führt zu einem Speicherverlust.

Um dies zu verhindern, verwenden Sie eine WeakReference in der AsyncTask, anstatt einen

direkten Verweis auf die Aktivität zu haben.

Hier ist ein Beispiel für AsyncTask, das eine WeakReference verwendet:

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

AsyncTask aus einer Aktivität aufrufen:

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

AsyncTask aus einem Fragment aufrufen:

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

Reihenfolge der Ausführung

Bei der ersten Einführung wurden `AsyncTasks` seriell in einem einzigen Hintergrundthread ausgeführt. Beginnend mit `DONUT` wurde dies in einen Pool von Threads geändert, sodass mehrere Aufgaben parallel ausgeführt werden können. Beginnend mit `HONEYCOMB` werden Aufgaben in einem einzelnen Thread ausgeführt, um häufige Anwendungsfehler zu vermeiden, die durch die parallele Ausführung verursacht werden.

Wenn Sie wirklich eine parallele Ausführung wünschen, können Sie den `executeOnExecutor(java.util.concurrent.Executor, Object[])`

mit `THREAD_POOL_EXECUTOR` .

> SERIAL_EXECUTOR -> Ein Executor, der Aufgaben nacheinander in serieller Reihenfolge ausführt.
>
> THREAD_POOL_EXECUTOR -> Ein Executor, mit dem Aufgaben parallel ausgeführt werden können.

Probe :

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

AsyncTask: Serienausführung und parallele Ausführung der Aufgabe

AsyncTask ist eine abstrakte Klasse und erbt nicht die `Thread` Klasse. Es verfügt über eine **abstrakte** Methode `doInBackground(Params... params)` , die zur Ausführung der Aufgabe überschrieben wird. Diese Methode wird aus `AsyncTask.call()` aufgerufen.

Executor sind Teil des Pakets `java.util.concurrent` .

Außerdem enthält AsyncTask 2 `Executor` Dateien

`THREAD_POOL_EXECUTOR`

Es verwendet Worker-Threads, um die Aufgaben parallel auszuführen.

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

`SERIAL_EXECUTOR`

Er führt die Aufgabe seriell aus, dh eine nach der anderen.

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

Beide `Executor` `THREAD_POOL_EXECUTOR` sind **statisch** , daher gibt es nur ein `THREAD_POOL_EXECUTOR` und ein `SerialExecutor` Objekt. Sie können jedoch mehrere `AsyncTask` Objekte erstellen.

Wenn Sie versuchen, mehrere Hintergrundaufgaben mit dem Standard-Executor (`SerialExecutor`) `SerialExecutor` , werden diese Aufgaben in die Warteschlange gestellt und seriell ausgeführt.

Wenn Sie versuchen, mehrere Hintergrundaufgaben mit `THREAD_POOL_EXECUTOR` , werden diese parallel ausgeführt.

Beispiel:

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

Ausführen Klicken Sie mehrmals auf die Schaltfläche, um eine Aufgabe zu starten und das Ergebnis anzuzeigen.

Im Thread-Pool ausgeführte Aufgabe (1)

Jede Aufgabe dauert 1000 ms.

Bei t = 36s werden die Aufgaben 2, 3 und 4 in die Warteschlange gestellt und gestartet, auch weil sie parallel ausgeführt werden.

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

Kommentar- `Task Executed in thread pool` (1) ausgeführt wird, und uncomment `Task executed Serially` (2).

Ausführen Klicken Sie mehrmals auf die Schaltfläche, um eine Aufgabe zu starten und das Ergebnis anzuzeigen.

Die Aufgabe wird seriell ausgeführt, daher wird jede Aufgabe gestartet, nachdem die Ausführung der aktuellen Aufgabe abgeschlossen ist. Wenn die Ausführung von Task 1 abgeschlossen ist, wird daher nur Task 2 im Hintergrund ausgeführt. Und umgekehrt.

```
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
```

AsyncTask online lesen: https://riptutorial.com/de/android/topic/117/asynctask

Kapitel 35: AudioManager

Examples

Transienten-Audio-Fokus anfordern

```
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);
        }
    }
}
```

Anfordern des Audiofokus

```
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.
        }
    }
}
```

AudioManager online lesen: https://riptutorial.com/de/android/topic/6798/audiomanager

Kapitel 36: AudioTrack

Examples

Ton einer bestimmten Frequenz erzeugen

Um einen Ton mit einem bestimmten Ton abzuspielen, müssen Sie zunächst einen Sinus-Klang erzeugen. Dies geschieht auf folgende Weise.

```
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
}
```

Jetzt müssen wir AudioTrack so konfigurieren, dass es dem generierten Puffer entspricht. Dies geschieht auf folgende Weise

```
AudioTrack audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC,
                    sampleRate, AudioFormat.CHANNEL_OUT_MONO,
                    AudioFormat.ENCODING_PCM_16BIT, buffer.length,
                    AudioTrack.MODE_STATIC);
```

Schreiben Sie den generierten Puffer und spielen Sie den Titel

```
audioTrack.write(buffer, 0, buffer.length);
audioTrack.play();
```

Hoffe das hilft :)

AudioTrack online lesen: https://riptutorial.com/de/android/topic/9155/audiotrack

Kapitel 37: Ausnahmen

Examples

NetworkOnMainThreadException

Aus der Dokumentation :

> Die Ausnahme, die ausgelöst wird, wenn eine Anwendung versucht, einen Netzwerkvorgang in ihrem Haupt-Thread auszuführen.
>
> Dies wird nur für Anwendungen ausgelöst, die auf das Honeycomb SDK oder höher abzielen. Anwendungen, die auf frühere SDK-Versionen abzielen, dürfen in ihren Hauptereignisschleifen-Threads vernetzen, es wird jedoch dringend davon abgeraten.

Hier ist ein Beispiel eines Codefragments, das diese Ausnahme verursachen kann:

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

Mit dem `NetworkOnMainThreadException` Code wird `NetworkOnMainThreadException` für Anwendungen `NetworkOnMainThreadException` auf Honeycomb SDK (Android v3.0) oder höher abzielen, da die Anwendung versucht, eine Netzwerkoperation im Hauptthread auszuführen.

Um diese Ausnahme zu vermeiden, müssen Ihre Netzwerkvorgänge immer in einer Hintergrundtask über eine `AsyncTask` , einen `Thread` , einen `IntentService` usw. ausgeführt werden.

```
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

Dies ist eine sehr häufige `Exception` . Dadurch wird Ihre Anwendung während des Starts oder der Ausführung Ihrer App angehalten. Im `LogCat` Sie die Meldung:

```
android.content.ActivityNotFoundException : Unable to find explicit activity class;
have you declared this activity in your AndroidManifest.xml?
```

Überprüfen Sie in diesem Fall, ob Sie Ihre Aktivität in der Datei `AndroidManifest.xml` .

Die einfachste Möglichkeit, Ihre `Activity` in `AndroidManifest.xml` zu deklarieren, ist:

```
<activity  android:name="com.yourdomain.YourStoppedActivity" />
```

Außerhalb des Speicherfehler

Dies ist ein Laufzeitfehler, der auftritt, wenn Sie eine große Menge Speicher auf dem Heap anfordern. Dies ist üblich, wenn eine Bitmap in eine ImageView geladen wird.

Sie haben einige Möglichkeiten:

1. Verwenden Sie einen großen Anwendungshaufen

Fügen Sie dem Anwendungstag in Ihrer AndroidManifest.xml die Option "largeHeap" hinzu. Dadurch wird Ihrer App mehr Speicher zur Verfügung gestellt, das Root-Problem wird jedoch wahrscheinlich nicht behoben.

```
<application largeHeap="true" ... >
```

2. Recyceln Sie Ihre Bitmaps

Stellen Sie nach dem Laden einer Bitmap sicher, dass Sie sie recyceln und Speicherplatz freigeben:

```
    if (bitmap != null && !bitmap.isRecycled())
       bitmap.recycle();
```

3. Laden Sie abgetastete Bitmaps in den Speicher

Vermeiden Sie das gleichzeitige Laden der gesamten Bitmap in den Arbeitsspeicher, indem Sie eine reduzierte Größe mit BitmapOptions und inSampleSize abfragen.

Siehe beispielsweise die Android-Dokumentation

DexException

```
com.android.dex.DexException: Multiple dex files define Lcom/example/lib/Class;
```

Dieser Fehler tritt auf, weil die App beim `.dex` zwei `.dex` Dateien findet, die dieselbe Gruppe von Methoden definieren.

Normalerweise geschieht dies, weil die App versehentlich zwei separate Abhängigkeiten von derselben Bibliothek erworben hat.

Angenommen, Sie haben ein Projekt und möchten sich auf zwei Bibliotheken verlassen: `A` und `B` , die jeweils eigene Abhängigkeiten haben. Wenn Bibliothek `B` bereits eine Abhängigkeit von Bibliothek `A` , wird dieser Fehler ausgegeben, wenn Bibliothek `A` von sich aus zum Projekt hinzugefügt wird. Beim Kompilieren der Bibliothek `B` bereits der Code von `A` . Wenn der Compiler die Bibliothek `A` bündelt, werden die Methoden der Bibliothek `A` bereits gepackt.

Stellen Sie zum Lösen sicher, dass keine Ihrer Abhängigkeiten auf diese Weise zweimal zufällig hinzugefügt werden konnte

Unbekannte Ausnahme

Wenn Sie nicht erfasste Ausnahmen behandeln möchten, versuchen Sie, sie alle in der onCreate-Methode Ihrer Application-Klasse abzufangen:

```
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();
    }
}
```

Registrieren eines eigenen Handlers für unerwartete Ausnahmen

So können Sie auf Ausnahmen reagieren, die nicht erfasst wurden, ähnlich dem Standard des Systems "Anwendung XYZ ist abgestürzt".

```
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
    }
}
```

Dann registrieren Sie diese Anwendungsklasse in Ihrer AndroidManifest.xml:

```
<application android:name="de.ioxp.arkmobile.MyApplication" >
```

Ausnahmen online lesen: https://riptutorial.com/de/android/topic/112/ausnahmen

Kapitel 38: AutoCompleteTextView

Bemerkungen

Wenn Sie dem Benutzer beim `AutoCompleteTextView` in ein bearbeitbares Textfeld Vorschläge unterbreiten möchten, können Sie eine `AutoCompleteTextView` . Es gibt automatisch Vorschläge, wenn der Benutzer schreibt. Die Liste der Vorschläge wird in einem Dropdown-Menü angezeigt, aus dem der Benutzer einen auswählen kann, um den Inhalt des Bearbeitungsfelds zu ersetzen.

Examples

Einfache, hart codierte AutoCompleteTextView

Design (Layout XML):

```
<AutoCompleteTextView
   android:id="@+id/autoCompleteTextView1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentTop="true"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="65dp"
   android:ems="10" />
```

Suchen Sie die Ansicht im Code nach `setContentView()` (oder dessen äquivalentem Fragment oder benutzerdefinierten Ansicht):

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

Festcodierte Daten über einen Adapter bereitstellen:

```
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);
```

Tipp: Die bevorzugte Methode wäre jedoch die Bereitstellung von Daten über einen beliebigen `Loader` anstelle einer hart codierten Liste wie dieser.

AutoComplete mit CustomAdapter, ClickListener und Filter

Hauptlayout: 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>
```

Zeilenlayout 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;
    }
}
```

Modellklasse: 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;
    }
```

```
}
```

Adapterklasse: 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();
                }
            }
        }
    };
}
```

AutoCompleteTextView online lesen:
https://riptutorial.com/de/android/topic/5300/autocompletetextview

Kapitel 39: Barcode- und QR-Code lesen

Bemerkungen

QRCodeReaderView

Zxing

Examples

Verwendung von QRCodeReaderView (basierend auf Zxing)

QRCodeReaderView implementiert eine Android-Ansicht, die die Kamera anzeigt und benachrichtigt, wenn sich in der Vorschau ein QR-Code befindet.

Es verwendet die Open-Source- Zxing -Bibliothek für die Verarbeitung von 1D / 2D-Barcodes mit mehreren Formaten.

Hinzufügen der Bibliothek zu Ihrem Projekt

Fügen Sie Ihrem build.gradle die QRCodeReaderView-Abhängigkeit hinzu

```
dependencies{
    compile 'com.dlazaro66.qrcodereaderview:qrcodereaderview:2.0.0'
}
```

Erste Benutzung

- Fügen Sie Ihrem Layout eine `QRCodeReaderView`

```
<com.dlazaro66.qrcodereaderview.QRCodeReaderView
       android:id="@+id/qrdecoderview"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />
```

- Erstellen Sie eine Aktivität, die `onQRCodeReadListener` implementiert, und verwenden Sie sie als Listener für `QrCodeReaderView` .
- Stellen Sie sicher, dass Sie über Kameraberechtigungen verfügen, um die Bibliothek verwenden zu können. (https://developer.android.com/training/permissions/requesting.html)

Dann können Sie es in Ihrer Aktivität wie folgt verwenden:

```
   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();
    }
}
```

Barcode- und QR-Code lesen online lesen: https://riptutorial.com/de/android/topic/6067/barcode--und-qr-code-lesen

Kapitel 40: Bedienung

Einführung

Ein Dienst wird im **Hintergrund ausgeführt** , um lange andauernde Vorgänge auszuführen oder um Arbeit für Remote-Prozesse auszuführen. Ein Dienst bietet keine Benutzeroberfläche, die nur im Hintergrund mit Benutzereingaben ausgeführt wird. Beispielsweise kann ein Dienst Musik im Hintergrund abspielen, während sich der Benutzer in einer anderen App befindet, oder er kann Daten aus dem Internet herunterladen, ohne die Interaktion des Benutzers mit dem Android-Gerät zu blockieren.

Bemerkungen

Wenn Sie Ihren Dienst nicht in Ihrer AndroidManifest.xml definiert haben, erhalten Sie beim Starten des Diensts eine `ServiceNotFoundException` .

Hinweis:

Informationen zu `IntentService` finden Sie hier: IntentService-Beispiel

Examples

Service starten

Das Starten eines Dienstes ist sehr einfach. Rufen `startService` einfach `startService` mit einer Absicht aus einer Aktivität heraus auf:

```
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.
```

Lebenszyklus eines Dienstes

Der Servicelebenszyklus hat die folgenden Rückrufe

- `onCreate()` :

Wird ausgeführt, wenn der Service zum ersten Mal erstellt wird, um die erforderlichen Anfangskonfigurationen einzurichten. Diese Methode wird nur ausgeführt, wenn der Dienst noch nicht ausgeführt wird.

- `onStartCommand()` :

`startService()` jedes Mal ausgeführt, wenn `startService()` von einer anderen Komponente wie

einer Aktivität oder einem BroadcastReceiver aufgerufen wird. Wenn Sie diese Methode verwenden, wird der Dienst ausgeführt, bis Sie `stopSelf()` oder `stopService()` . Beachten Sie, dass die Methoden `stopSelf()` und `stopService()` nur einmal aufgerufen werden müssen, unabhängig davon, wie oft Sie `onStartCommand()` `stopSelf()` , um den Dienst zu stoppen.

- `onBind()` :

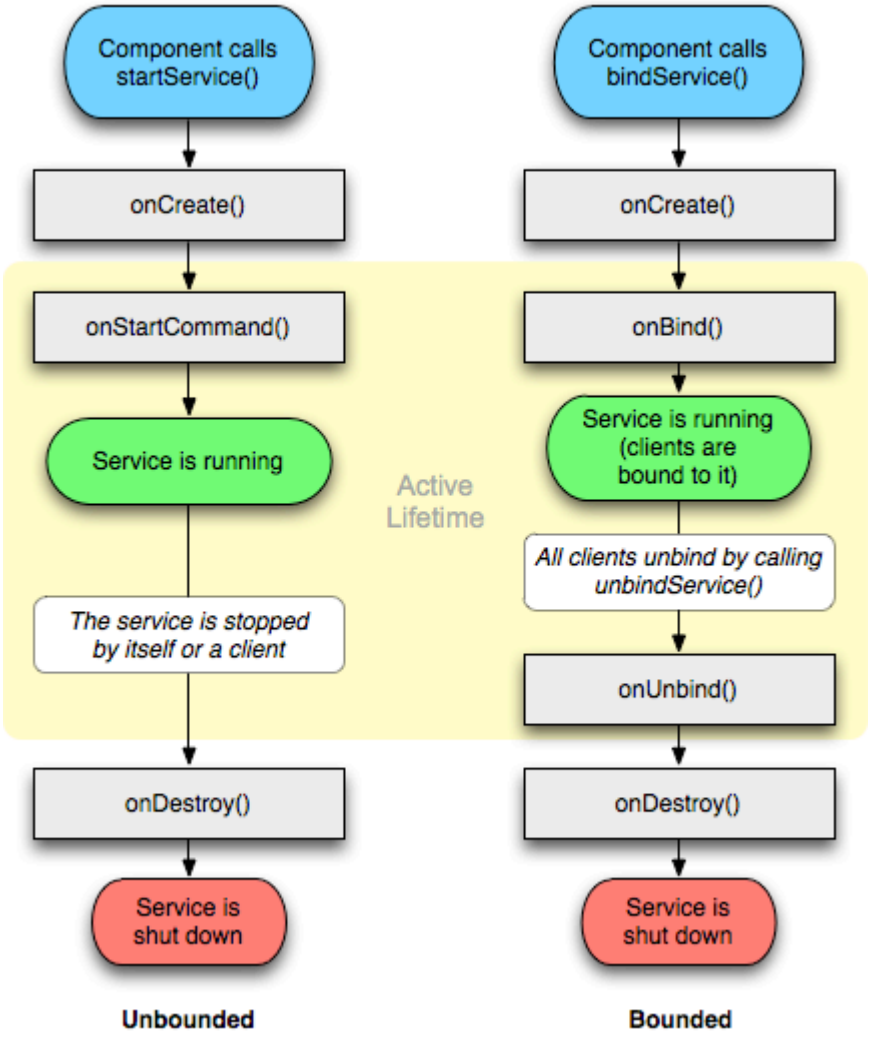
`bindService()` ausgeführt, wenn eine Komponente `bindService()` und eine Instanz von IBInder zurückgibt, die einen Kommunikationskanal für den Service bereitstellt. Ein Aufruf von `bindService()` hält den Dienst so lange an, wie Clients daran gebunden sind.

- `onDestroy()` :

Wird ausgeführt, wenn der Dienst nicht mehr verwendet wird, und ermöglicht die Entsorgung der zugewiesenen Ressourcen.

Es ist wichtig zu beachten, dass während des Lebenszyklus eines Diensts andere Callbacks aufgerufen werden können, z. B. `onConfigurationChanged()` und `onLowMemory()`

https://developer.android.com/guide/components/services.html



Definieren des Prozesses eines Services

Das `android:process` Feld definiert den Namen des Prozesses, in dem der Dienst ausgeführt werden soll. Normalerweise werden alle Komponenten einer Anwendung in dem Standardprozess ausgeführt, der für die Anwendung erstellt wurde. Eine Komponente kann jedoch den Standardwert mit einem eigenen Prozessattribut überschreiben, sodass Sie Ihre Anwendung auf mehrere Prozesse verteilen können.

Wenn der diesem Attribut zugewiesene Name mit einem Doppelpunkt (':') beginnt, wird der Dienst in einem separaten Prozess ausgeführt.

```
<service
  android:name="com.example.appName"
  android:process=":externalProcess" />
```

Wenn der Prozessname mit einem Kleinbuchstaben beginnt, wird der Dienst in einem globalen Prozess mit diesem Namen ausgeführt, sofern er dazu berechtigt ist. Dadurch können Komponenten in verschiedenen Anwendungen einen Prozess gemeinsam nutzen, wodurch der Ressourcenverbrauch reduziert wird.

Bound Service mit Hilfe von Binder erstellen

Erstellen Sie eine Klasse, die die `Service` Klasse erweitert und in der überschriebenen Methode `onBind` Ihre lokale `onBind` zurückgeben:

```
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;
    }
}
```

Dann binden Sie in Ihrer Aktivität an den Service in `onStart` callback, verwenden die `ServiceConnection` Instanz und binden die Bindung von ihr in `onStop`

```
public class BindingActivity extends Activity {
    LocalService mService;
    boolean mBound = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    protected void onStart() {
        super.onStart();
        // Bind to LocalService
        Intent intent = new Intent(this, LocalService.class);
        bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
    }

    @Override
    protected void onStop() {
        super.onStop();
        // Unbind from the service
        if (mBound) {
            unbindService(mConnection);
            mBound = false;
        }
    }

    /** Defines callbacks for service binding, passed to bindService() */
    private ServiceConnection mConnection = new ServiceConnection() {

        @Override
        public void onServiceConnected(ComponentName className,
                IBinder service) {
            // We've bound to LocalService, cast the IBinder and get LocalService instance
            LocalBinder binder = (LocalBinder) service;
            mService = binder.getService();
            mBound = true;
        }

        @Override
        public void onServiceDisconnected(ComponentName arg0) {
            mBound = false;
        }
    };
}
```

Remote-Service erstellen (über AIDL)

Beschreiben Sie Ihre Service-Zugriffsschnittstelle über die `.aidl` Datei:

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

Nach dem Erstellen der Anwendung generieren die sdk-Tools die entsprechende `.java` Datei. Diese Datei enthält die `Stub` Klasse, die unsere Hilfsschnittstelle implementiert und die wir

erweitern müssen:

```
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;
    }
}
```

Dann in Tätigkeit:

```
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;
    }
}
```

Erstellen eines ungebundenen Dienstes

Als Erstes fügen Sie den Service zu `AndroidManifest.xml` im `<application>` -Tag hinzu:

```
<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>
```

Wenn Sie Ihre Serviceklasse in einem separaten Paket verwalten möchten (z. B. .AllServices.RecordingService), müssen Sie angeben, wo sich Ihr Service befindet. In diesem Fall werden wir Folgendes ändern:

```
android:name=".RecordingService"
```

zu

```
android:name=".AllServices.RecordingService"
```

oder am einfachsten ist es, den vollständigen Paketnamen anzugeben.

Dann erstellen wir die eigentliche Serviceklasse:

```
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();
    }

}
```

Dieser Dienst zeigt lediglich eine Benachrichtigung an, wenn er ausgeführt wird, und er kann

Toasts anzeigen, wenn seine `doSomething()` -Methode aufgerufen wird.

Wie Sie feststellen werden, ist es als Singleton implementiert und verfolgt die eigene Instanz - aber ohne die übliche statische Singleton Factory-Methode, da Services von Natur aus Singletons sind und durch Absichten erstellt werden. Die Instanz ist nach außen nützlich, um den Dienst bei seiner Ausführung "in den Griff zu bekommen".

Zuletzt müssen wir den Dienst aus einer Aktivität heraus starten und stoppen:

```
public void startOrStopService(){
    if( RecordingService.isRunning ){
        // Stop service
        Intent intent = new Intent(this, RecordingService.class);
        stopService(intent);
    }
    else {
        // Start service
        Intent intent = new Intent(this, RecordingService.class);
        startService(intent);
    }
}
```

In diesem Beispiel wird der Dienst abhängig vom aktuellen Status auf dieselbe Weise gestartet und gestoppt.

Wir können auch die `doSomething()` -Methode aus unserer Aktivität aufrufen:

```
public void makeServiceDoSomething(){
    if( RecordingService.isRunning )
        RecordingService.instance.doSomething();
}
```

Bedienung online lesen: https://riptutorial.com/de/android/topic/137/bedienung

Kapitel 41: Benachrichtigungen

Examples

Erstellen einer einfachen Benachrichtigung

In diesem Beispiel wird veranschaulicht, wie eine einfache Benachrichtigung erstellt wird, mit der eine Anwendung gestartet wird, wenn der Benutzer darauf klickt.

Geben Sie den Inhalt der Benachrichtigung an:

```
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
```

Erstellen Sie die Absicht, mit einem Klick abzufeuern:

```
Intent intent = new Intent(this, MainActivity.class);
PendingIntent pi = PendingIntent.getActivity(this, 0, intent, Intent.FLAG_ACTIVITY_NEW_TASK);
mBuilder.setContentIntent(pi);
```

Erstellen Sie schließlich die Benachrichtigung und zeigen Sie sie

```
NotificationManager mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(0, mBuilder.build());
```

Heads Up-Benachrichtigung mit Ticker für ältere Geräte

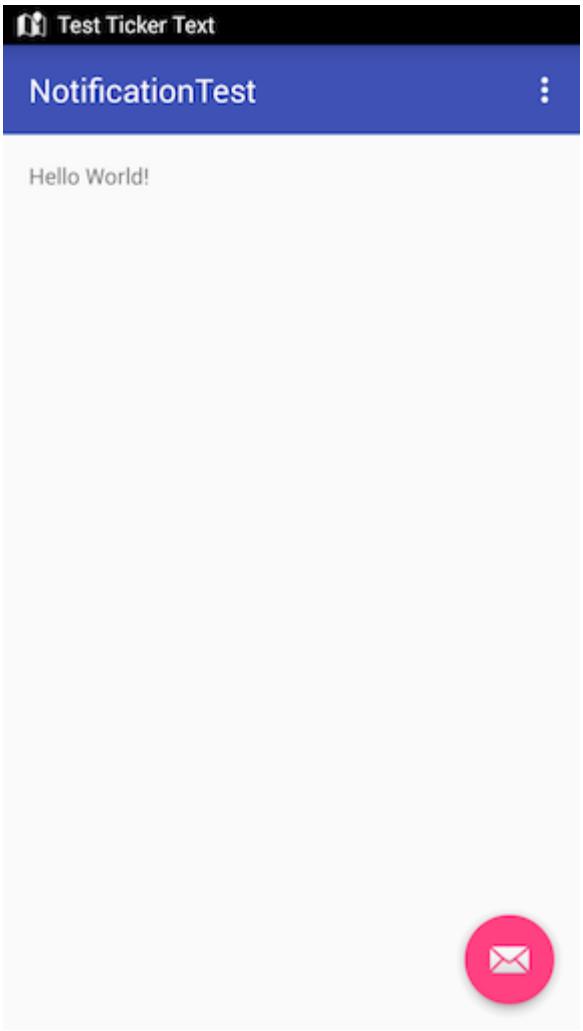
So erstellen Sie eine Heads-Up-Benachrichtigung für fähige Geräte und einen Ticker für ältere Geräte.

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

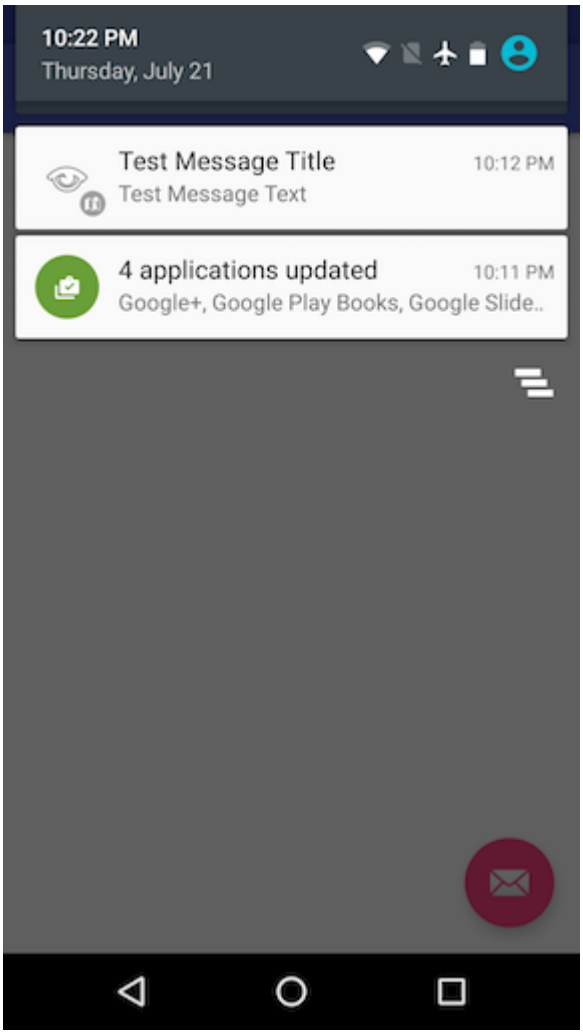
So sieht es auf Android Marshmallow mit der Heads Up-Benachrichtigung aus:

So sieht es auf Android KitKat mit dem Ticker aus:

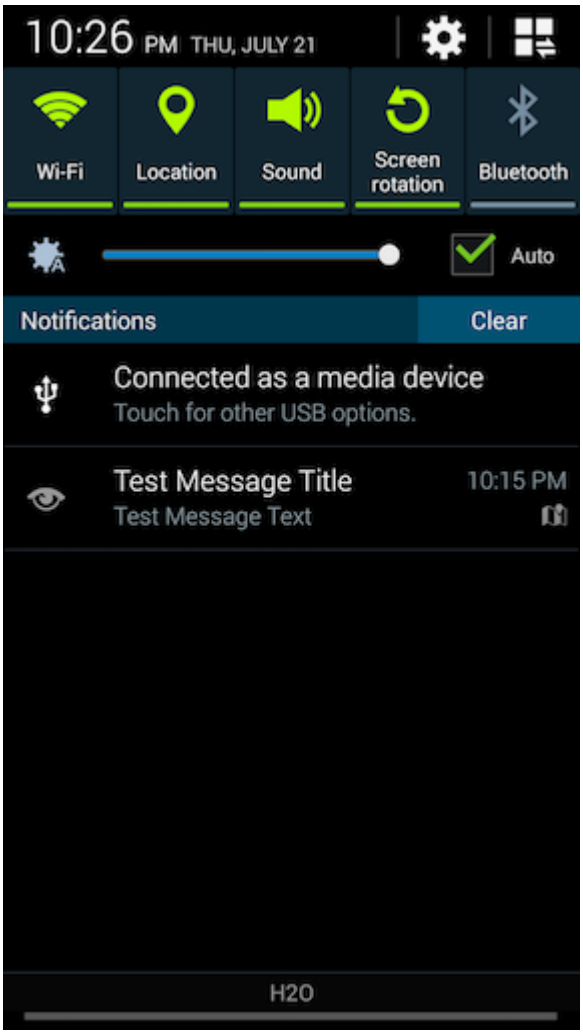


Bei allen Android-Versionen wird die `Notification` im `Notification` angezeigt.

Android 6.0 Eibisch:



Android 4.4.x KitKat:



Einstellen verschiedener Prioritäten in der Benachrichtigung

```
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);
```

> Wenn die Benachrichtigung ein Bild enthält und Sie das Bild automatisch erweitern möchten, wenn die Benachrichtigung "PRIORITY_MAX" erhalten hat, können Sie je nach Anforderung andere Prioritätsstufen verwenden

Verschiedene Prioritätsstufen Info:

PRIORITY_MAX - Verwenden Sie diese **Option** für kritische und dringende Benachrichtigungen, die den Benutzer auf einen zeitkritischen Zustand aufmerksam machen oder aufgelöst werden müssen, bevor er mit einer bestimmten Aufgabe fortfahren kann.

PRIORITY_HIGH - **Wird** hauptsächlich für wichtige Kommunikation verwendet, z. B. Nachrichten- oder Chat-Ereignisse mit Inhalten, die für den Benutzer besonders interessant sind. Benachrichtigungen mit hoher Priorität lösen die Anzeige der Heads-Up-Benachrichtigungen aus.

PRIORITY_DEFAULT - Verwenden Sie diese **Option** für alle Benachrichtigungen, die nicht zu den anderen hier beschriebenen Prioritäten gehören.

PRIORITY_LOW - Verwenden Sie diese **Option** für Benachrichtigungen, über die der Benutzer informiert werden soll, die jedoch weniger dringend sind. Benachrichtigungen mit niedriger Priorität werden in der Regel am Ende der Liste angezeigt. Dies macht sie zu einer guten Wahl für öffentliche oder ungerichtete soziale Updates: Der Benutzer hat darum gebeten, über sie benachrichtigt zu werden. Diese Benachrichtigungen sollten jedoch niemals Vorrang vor dringenden oder dringenden Aktionen haben direkte Kommunikation.

PRIORITY_MIN - **Wird** für Kontext- oder Hintergrundinformationen wie Wetterinformationen oder Kontextinformationen verwendet. Benachrichtigungen mit der niedrigsten Priorität werden nicht in der Statusleiste angezeigt. Der Benutzer entdeckt sie beim Erweitern des Benachrichtigungsbereichs.

Referenzen: Richtlinien für Materialdesign - Benachrichtigungen

Benachrichtigungen planen

Manchmal ist es erforderlich, eine Benachrichtigung zu einem bestimmten Zeitpunkt anzuzeigen, eine Aufgabe, die auf dem Android-System leider nicht trivial ist, da es keine Methode `setTime()` oder ähnliches für Benachrichtigungen gibt. In diesem Beispiel werden die Schritte beschrieben, die zum `AlarmManager` Benachrichtigungen mit dem `AlarmManager` :

1. **Fügen** Sie **eine** **`BroadcastReceiver`** , die zuhört `Intent` s vom Android ausgestrahlt `AlarmManager` .

Dies ist der Ort, an dem Sie Ihre Benachrichtigung auf der Grundlage der mit dem `Intent` bereitgestellten Extras erstellen:

```
   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. **Registrieren** Sie **die** **`BroadcastReceiver`** in Ihrer `AndroidManifest.xml` Datei (sonst wird der Empfänger nicht erhalten keine `Intent` s aus dem `AlarmManager`):

```
<receiver
    android:name=".NotificationReceiver"
```

```
    android:enabled="true" />
```

3. `PendingIntent` Sie **eine Benachrichtigung,** indem Sie einen `PendingIntent` für Ihren `BroadcastReceiver` mit den erforderlichen `Intent` Extras an den `AlarmManager` des Systems `AlarmManager` . Ihr `BroadcastReceiver` empfängt die `Intent` sobald die angegebene Zeit eingetroffen ist, und zeigt die Benachrichtigung an. Die folgende Methode plant eine Benachrichtigung:

```
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);
}
```

Bitte beachten Sie, dass die `42` oben für jede geplante Benachrichtigung eindeutig sein müssen. Andernfalls werden sich die `PendingIntent` gegenseitig ersetzen, was zu unerwünschten Effekten führt!

4. **Brechen Sie eine Benachrichtigung ab,** indem Sie den zugehörigen `PendingIntent` und ihn im System `AlarmManager` . Die folgende Methode bricht eine Benachrichtigung ab:

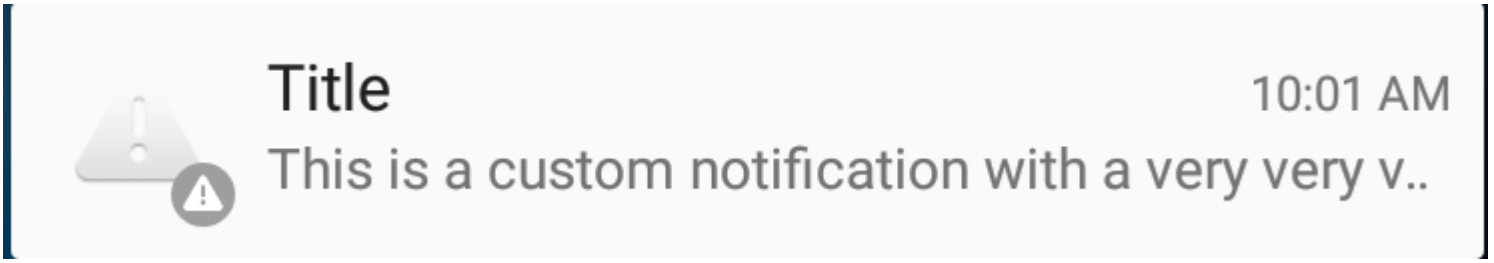
```
public static void cancelNotification(Context context, String title, String text) {
    Intent intent = new Intent(context, NotificationReceiver.class);
    intent.putExtra("title", title);
    intent.putExtra("text", text);
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
    // Cancel notification
    AlarmManager manager = (AlarmManager)
context.getSystemService(Context.ALARM_SERVICE);
    manager.cancel(pending);
}
```

Beachten Sie, dass die `42` oben mit der Nummer aus Schritt 3 übereinstimmen muss!

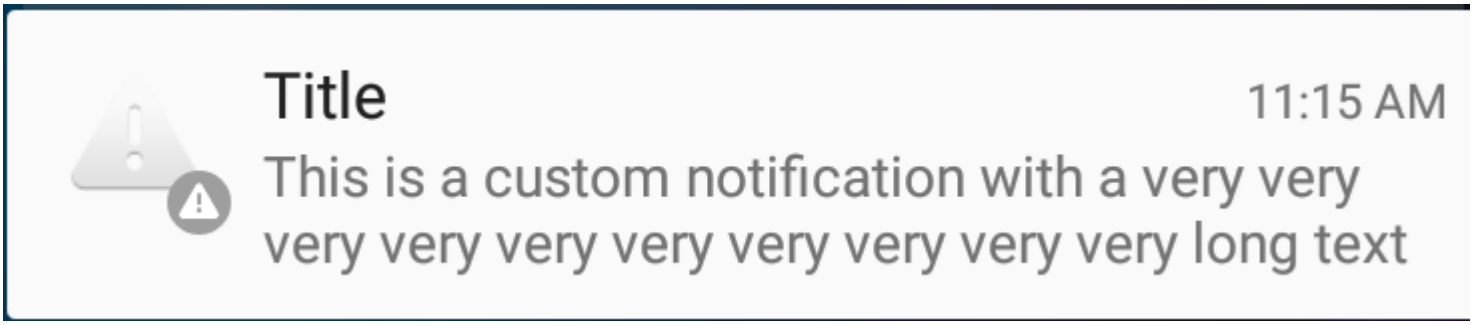
Benutzerdefinierte Benachrichtigung festlegen - Den gesamten Inhaltstext anzeigen

Wenn Sie möchten, dass ein langer Text im Kontext angezeigt wird, müssen Sie einen benutzerdefinierten Inhalt festlegen.

Zum Beispiel haben Sie Folgendes:



Sie wünschen jedoch, dass Ihr Text vollständig angezeigt wird:



Alles, was Sie tun müssen, ist, Ihren Inhalten **einen Stil** wie folgt **hinzuzufügen** :

```
  private void generateNotification(Context context) {
        String message = "This is a custom notification with a very very very very very very
very very very very long text";
        Bitmap largeIcon = BitmapFactory.decodeResource(getResources(),
android.R.drawable.ic_dialog_alert);

        NotificationCompat.Builder builder = new NotificationCompat.Builder(context);

        builder.setContentTitle("Title").setContentText(message)
                .setSmallIcon(android.R.drawable.ic_dialog_alert)
                .setLargeIcon(largeIcon)
                .setAutoCancel(true)
                .setWhen(System.currentTimeMillis())
                .setStyle(new NotificationCompat.BigTextStyle().bigText(message));

        Notification notification = builder.build();
        NotificationManagerCompat notificationManager =
NotificationManagerCompat.from(context);
        notificationManager.notify(101, notification);
    }
```

Benutzerdefiniertes Benachrichtigungssymbol mithilfe der `Picasso`-Bibliothek einstellen

```
    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);
```

Und definieren Sie dann ein Layout in Ihrem Layoutordner:

```
<?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>
```

Dynamisches Abrufen der korrekten Pixelgröße für das große Symbol

Wenn Sie ein Bild erstellen, ein Bild dekodieren oder die Größe eines Bilds an den großen Benachrichtigungsbildbereich anpassen, können Sie die korrekten Pixelabmessungen wie folgt ermitteln:

```
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);
```

Laufende Benachrichtigung mit Aktionstaste

```
// 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);
```

Registrieren Sie einen Broadcast-Empfänger für dieselbe Aktion, um das Klickereignis für die Aktionsschaltflächen zu bearbeiten.

Benachrichtigungen online lesen: https://riptutorial.com/de/android/topic/1347/benachrichtigungen

Kapitel 42: Benachrichtigungskanal Android O

Einführung

Benachrichtigungskanäle ermöglichen es App-Entwicklern, unsere Benachrichtigungen in Gruppen (Channels) zu gruppieren, wobei der Benutzer die Benachrichtigungseinstellungen für den gesamten Kanal gleichzeitig ändern kann. In Android O wird diese Funktion eingeführt. Jetzt steht eine Vorschau für Entwickler zur Verfügung.

Syntax

1. class NotificationUtils {} // Zum Erstellen eines Benachrichtigungskanals
2. createChannel () // Generische Methode zum Erstellen von Benachrichtigungen

Parameter

Methode	Beschreibung
IMPORTANCE_MAX	ungebraucht
WICHTIGKEIT: HOCH	zeigt überall, macht Lärm und Blicke
IMPORTANCE_DEFAULT	zeigt überall, macht Lärm, stört aber nicht
IMPORTANCE_LOW	zeigt überall, ist aber nicht aufdringlich
IMPORTANCE_MIN	nur im Schatten, unterhalb der Falte
IMPORTANCE_NONE	eine Benachrichtigung ohne Bedeutung; zeigt nicht im Schatten

Examples

Benachrichtigungskanal

Was sind Benachrichtigungskanäle?

Benachrichtigungskanäle ermöglichen es App-Entwicklern, unsere Benachrichtigungen in Gruppen (Channels) zu gruppieren, wobei der Benutzer die Benachrichtigungseinstellungen für den gesamten Kanal gleichzeitig ändern kann. Zum Beispiel können Benutzer für jeden Kanal alle Benachrichtigungen vollständig blockieren, die Wichtigkeitsstufe überschreiben oder die Anzeige eines Benachrichtigungsausweises zulassen. Diese neue Funktion trägt dazu bei, die Benutzererfahrung einer App erheblich zu verbessern.

Erstellen Sie Benachrichtigungskanäle

```
 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;
}}
```

Im obigen Code haben wir zwei Instanzen des NotificationChannel erstellt, wobei uniqueid einen

Kanalnamen und eine Wichtigkeitsstufe in seinem Konstruktor übergeben. Für jeden Benachrichtigungskanal haben wir die folgenden Merkmale angewendet.

1. Klingen
2. Beleuchtung
3. Vibration
4. Benachrichtigung, die auf dem Sperrbildschirm angezeigt wird.

Schließlich haben wir den NotificationManager vom System erhalten und dann den Kanal registriert, indem wir die Methode createNotificationChannel () aufgerufen haben und den von uns erstellten Kanal übergeben.

Mit createNotificationChannels () können wir mehrere Benachrichtigungskanäle auf einmal erstellen und eine Java-Liste mit NotificationChannel-Instanzen übergeben. Sie können alle Benachrichtigungskanäle für eine App mit getNotificationChannels () und einen bestimmten Kanal mit getNotificationChannel () abrufen, wobei nur die Kanal-ID als Argument übergeben wird.

Wichtigkeitsgrad in Benachrichtigungskanälen

Methode	Beschreibung
IMPORTANCE_MAX	ungebraucht
WICHTIGKEIT: HOCH	zeigt überall, macht Lärm und Blicke
IMPORTANCE_DEFAULT	zeigt überall, macht Lärm, stört aber nicht
IMPORTANCE_LOW	zeigt überall, ist aber nicht aufdringlich, der Wert ist 0
IMPORTANCE_MIN	nur im Schatten, unterhalb der Falte
IMPORTANCE_NONE	eine Benachrichtigung ohne Bedeutung; zeigt nicht im Schatten

Erstellen Sie eine Benachrichtigung und veröffentlichen Sie sie in den Kanal

Wir haben zwei Benachrichtigungen erstellt, die NotificationUtils verwenden, und NotificationBuilder.

```
public Notification.Builder getAndroidChannelNotification(String title, String body) {
    return new Notification.Builder(getApplicationContext(), ANDROID_CHANNEL_ID)
            .setContentTitle(title)
            .setContentText(body)
            .setSmallIcon(android.R.drawable.stat_notify_more)
            .setAutoCancel(true);
}

public Notification.Builder getIosChannelNotification(String title, String body) {
    return new Notification.Builder(getApplicationContext(), IOS_CHANNEL_ID)
            .setContentTitle(title)
            .setContentText(body)
            .setSmallIcon(android.R.drawable.stat_notify_more)
            .setAutoCancel(true);
```

```
}
```

Wir können NotificationChannel auch mit Notification.Builder () ***einstellen*** , dazu können wir ***setChannel (String channelId) verwenden*** .

Benachrichtigungskanaleinstellungen aktualisieren

Nachdem Sie einen Benachrichtigungskanal erstellt haben, ist der Benutzer für die Einstellungen und das Verhalten verantwortlich. Sie können createNotificationChannel () erneut aufrufen, um einen vorhandenen Benachrichtigungskanal umzubenennen oder die Beschreibung zu aktualisieren. Der folgende Beispielcode beschreibt, wie Sie einen Benutzer zu den Einstellungen für einen Benachrichtigungskanal umleiten können, indem Sie eine Absicht erstellen, um eine Aktivität zu starten. In diesem Fall erfordert die Absicht erweiterte Daten, einschließlich der ID des Benachrichtigungskanals und des Paketnamens Ihrer App.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    //...
    Button buttonAndroidNotifSettings = (Button)
findViewById(R.id.btn_android_notif_settings);
    buttonAndroidNotifSettings.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View view) {
            Intent i = new Intent(Settings.ACTION_CHANNEL_NOTIFICATION_SETTINGS);
            i.putExtra(Settings.EXTRA_APP_PACKAGE, getPackageName());
            i.putExtra(Settings.EXTRA_CHANNEL_ID, NotificationUtils.ANDROID_CHANNEL_ID);
            startActivity(i);
        }
    });
}
```

XML-Datei:

```
<!--...-->
<Button
    android:id="@+id/btn_android_notif_settings"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Notification Settings"/>
<!--...-->
```

Benachrichtigungskanal löschen

Sie können Benachrichtigungskanäle löschen, indem Sie deleteNotificationChannel () aufrufen.

```
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);
```

Erstellen Sie jetzt MainActivity und XML

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/activity_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:layout_margin="16dp"
        tools:context="com.chikeandroid.tutsplusalerts.MainActivity">

    <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Tuts+ Android Channel"
                android:layout_gravity="center_horizontal"
                android:textAppearance="@style/TextAppearance.AppCompat.Title"/>

        <EditText
                android:id="@+id/et_android_title"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Title"/>

        <EditText
                android:id="@+id/et_android_author"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Author"/>

        <Button
                android:id="@+id/btn_send_android"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Send"/>
    </LinearLayout>

    <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_marginTop="20dp">

        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Tuts+ IOS Channel"
                android:layout_gravity="center_horizontal"
                android:textAppearance="@style/TextAppearance.AppCompat.Title"/>
```

```
        <EditText
                android:id="@+id/et_ios_title"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Title"
                />

        <EditText
                android:id="@+id/et_ios_author"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Author"/>
        <Button
                android:id="@+id/btn_send_ios"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Send"/>
    </LinearLayout>

</LinearLayout>
```

MainActivity.java

Wir werden unsere MainActivity so bearbeiten, dass Titel und Autor von den EditText-Komponenten abgerufen und an den Android-Kanal gesendet werden können. Wir erhalten den Notification.Builder für den Android-Kanal, den wir in unseren NotificationUtils erstellt haben, und benachrichtigen dann den NotificationManager.

import android.app.Notification; import android.os.Bundle; import android.support.v7.app.AppCompatActivity; import android.text.TextUtils; 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());
                }
            }
        });
    }
}
```

Benachrichtigungskanal Android O online lesen:
https://riptutorial.com/de/android/topic/10018/benachrichtigungskanal-android-o

Kapitel 43: Benutzerdefinierte Ansichten erstellen

Examples

Benutzerdefinierte Ansichten erstellen

Wenn Sie eine vollständig angepasste Ansicht benötigen, müssen Sie die Ansicht " `View` (die Oberklasse aller Android-Ansichten) unterteilen und Ihre benutzerdefinierten `onMeasure(...)`) und Zeichnungsmethoden (`onDraw(...)`) `onDraw(...)` :

1. **Erstellen Sie Ihr benutzerdefiniertes Ansichtskelett:** Dies ist grundsätzlich für jede benutzerdefinierte Ansicht gleich. Hier erstellen wir das Skelett für eine benutzerdefinierte Ansicht, die einen Smiley namens `SmileyView` :

```
public class SmileyView extends View {
    private Paint mCirclePaint;
    private Paint mEyeAndMouthPaint;

    private float mCenterX;
    private float mCenterY;
    private float mRadius;
    private RectF mArcBounds = new RectF();

    public SmileyView(Context context) {
        this(context, null, 0);
    }

    public SmileyView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        initPaints();
    }

    private void initPaints() {/* ... */}

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {/* ... */}

    @Override
    protected void onDraw(Canvas canvas) {/* ... */}
}
```

2. **Initialisieren Sie Ihre Farben:** Die `Paint` Objekte sind die Pinsel Ihrer virtuellen Leinwand, die festlegen, wie Ihre geometrischen Objekte (z. B. Farbe, Füll- und Konturenstil usw.) gerendert werden. Hier erstellen wir zwei `Paint` , eine gelbe Farbe für den Kreis und eine schwarze Strichfarbe für Augen und Mund:

```
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. **Implementieren Sie Ihre eigene `onMeasure(...)` -Methode:** Dies ist erforderlich, damit die übergeordneten Layouts (z. B. `FrameLayout`) Ihre benutzerdefinierte Ansicht richtig ausrichten können. Sie enthält eine Reihe von `measureSpecs` , mit deren Hilfe Sie Höhe und Breite Ihrer Ansicht bestimmen können. Hier erstellen wir ein Quadrat, indem wir sicherstellen, dass Höhe und Breite gleich sind:

```
@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);
}
```

   Beachten Sie, dass `onMeasure(...)` mindestens einen Aufruf von `setMeasuredDimension(..)` enthalten muss. `setMeasuredDimension(..)` Ihre benutzerdefinierte Ansicht mit einer `IllegalStateException` .

4. **Implementieren Sie Ihre eigene `onSizeChanged(...)` -Methode:** Damit können Sie die aktuelle Höhe und Breite Ihrer benutzerdefinierten Ansicht abrufen, um Ihren Rendering-Code richtig anzupassen. Hier berechnen wir nur unser Zentrum und unseren Radius:

```
@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. **Implementieren Sie Ihre eigene `onDraw(...)` -Methode:** Hier implementieren Sie das tatsächliche Rendern Ihrer Ansicht. Es stellt ein `Canvas` Objekt bereit, auf das Sie `Canvas` können (alle offiziellen Zeichenmethoden finden Sie in der offiziellen `Canvas` Dokumentation).

```
@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. **Fügen Sie Ihre benutzerdefinierte Ansicht einem Layout hinzu:** Die benutzerdefinierte Ansicht kann jetzt in alle vorhandenen Layoutdateien eingefügt werden. Hier wickeln wir es einfach in ein `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>
```

Beachten Sie, dass es empfohlen wird, Ihr Projekt zu erstellen, nachdem der Ansichtscode abgeschlossen ist. Ohne sie zu erstellen, können Sie die Ansicht auf einem Vorschaubildschirm in Android Studio nicht sehen.

Nachdem Sie alles zusammengestellt haben, sollten Sie nach dem Starten der Aktivität, die das obige Layout enthält, mit dem folgenden Bildschirm begrüßt werden:

Attribute zu Ansichten hinzufügen

Benutzerdefinierte Ansichten können auch benutzerdefinierte Attribute enthalten, die in Android-Layout-Ressourcendateien verwendet werden können. Um Ihrer benutzerdefinierten Ansicht Attribute hinzuzufügen, müssen Sie Folgendes tun:

1. **Definieren Sie den Namen und den Typ Ihrer Attribute:** Dies geschieht in `res/values/attrs.xml` (`res/values/attrs.xml` erstellen). Die folgende Datei definiert ein Farbattribut für die Gesichtsfarbe unseres Smileys und ein Aufzählungsattribut für den Ausdruck des Smileys:

```
<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. **Verwenden Sie Ihre Attribute in Ihrem Layout:** Dies kann in allen Layoutdateien erfolgen, die Ihre benutzerdefinierte Ansicht verwenden. Die folgende Layoutdatei erstellt einen Bildschirm mit einem fröhlichen gelben Smiley:

```
<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>
```

   Tipp: Benutzerdefinierte Attribute funktionieren nicht mit den `tools:` Präfix in Android Studio 2.1 und älter (und möglicherweise in zukünftigen Versionen). In diesem Beispiel wird `app:smileyColor` durch `tools:smileyColor` , `smileyColor` weder zur Laufzeit noch zur Entwurfszeit festgelegt wird.

3. **Lesen Sie Ihre Attribute:** Dies geschieht in Ihrem benutzerdefinierten Ansichts-Quellcode. Der folgende Ausschnitt von `SmileyView` zeigt, wie die Attribute extrahiert werden können:

```
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. **(Optional) Standardstil** hinzufügen **:** Dazu fügen Sie einen Stil mit den Standardwerten hinzu und laden ihn in Ihre benutzerdefinierte Ansicht. Der folgende Standard-Smiley-Stil steht für einen glücklichen gelben Stil:

```
<!-- styles.xml -->
<style name="DefaultSmileyStyle">
    <item name="smileyColor">#ffff00</item>
    <item name="smileyExpression">happy</item>
</style>
```

   Welches wird in unserem `SmileyView` angewendet, indem es als letzter Parameter des Aufrufs von `obtainStyledAttributes` (siehe Code in Schritt 3):

```
TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView,
defStyleAttr, R.style.DefaultSmileyViewStyle);
```

   Beachten Sie, dass bei Attributwerten, die in der aufgeblähten Layoutdatei festgelegt wurden (siehe Code in Schritt 2), die entsprechenden Werte des Standardstils überschrieben werden.

5. **(Optional) Bereitstellen von Stilen in Designs:** Dazu fügen Sie ein neues Stilreferenzattribut hinzu, das in Ihren Designs verwendet werden kann, und stellt einen Stil für dieses Attribut bereit. Hier nennen wir einfach unser Referenzattribut `smileyStyle` :

```
<!-- attrs.xml -->
<attr name="smileyStyle" format="reference" />
```

   Für das wir dann einen Stil in unserem App-Theme bereitstellen (hier verwenden wir einfach den Standardstil aus Schritt 4):

```
<!-- themes.xml -->
<style name="AppTheme" parent="AppBaseTheme">
    <item name="smileyStyle">@style/DefaultSmileyStyle</item>
```

```
</style>
```

Zusammengesetzte Ansicht erstellen

Eine **zusammengesetzte Ansicht** ist eine benutzerdefinierte `ViewGroup` , die vom umgebenden Programmcode als einzelne Ansicht behandelt wird. Eine solche ViewGroup kann im DDD-ähnlichen Design sehr nützlich sein, da sie einem Aggregat, in diesem Beispiel einem Kontakt, entsprechen kann. Sie kann überall dort verwendet werden, wo der Kontakt angezeigt wird.

Dies bedeutet, dass der umgebende Controller-Code, eine Aktivität, ein Fragment oder ein Adapter, das Datenobjekt einfach an die Ansicht übergeben kann, ohne es in einer Reihe verschiedener UI-Widgets aufzuteilen.

Dies erleichtert die Wiederverwendung von Code und sorgt für ein besseres Design gemäß den SOLID-Grundsätzen .

Das Layout XML

Hier fängt man normalerweise an. Sie haben ein vorhandenes Stück XML, das Sie wiederverwenden können, möglicherweise als `<include/>` . Extrahieren Sie es in eine separate XML-Datei und umschließen Sie das Root-Tag in einem `<merge>` -Element:

```
<?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>
```

Diese XML-Datei funktioniert weiterhin einwandfrei im Layout-Editor von Android Studio. Sie können es wie jedes andere Layout behandeln.

Die zusammengesetzte ViewGroup

Wenn Sie die XML-Datei haben, erstellen Sie die benutzerdefinierte Ansichtsgruppe.

```
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);
        }
    }
}
```

Bei der `init(Context, AttributeSet)` Methode `init(Context, AttributeSet)` lesen Sie benutzerdefinierte XML-Attribute, wie unter Hinzufügen von Attributen zu Ansichten beschrieben .

Mit diesen Stücken können Sie sie in Ihrer App verwenden.

Verwendung in XML

Hier ein Beispiel für `fragment_contact_info.xml` , das veranschaulicht, wie Sie eine einzelne ContactView über eine Liste von Nachrichten setzen:

```
<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>
```

Verwendung im Code

Hier ist ein Beispiel für `RecyclerView.Adapter` , das eine Liste von Kontakten enthält. Dieses Beispiel zeigt, wie viel sauberer der Controller-Code wird, wenn er völlig frei von View-Manipulationen ist.

```
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-Leistungstipps

Ordnen Sie keine neuen Objekte in **onDraw zu**

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = new Paint(); //Do not allocate here
}
```

Anstatt Drawables auf Leinwand zu zeichnen ...

```
drawable.setBounds(boundsRect);

drawable.draw(canvas);
```

Verwenden Sie eine Bitmap zum schnelleren Zeichnen:

```
canvas.drawBitmap(bitmap, srcRect, boundsRect, paint);
```

Zeichnen Sie nicht die gesamte Ansicht neu, um nur einen kleinen Teil der Ansicht zu aktualisieren. Zeichnen Sie stattdessen den bestimmten Teil der Ansicht neu.

```
invalidate(boundToBeRefreshed);
```

Wenn Ihre Ansicht eine kontinuierliche Animation tut, zum Beispiel eine Uhr-Gesicht jede Sekunde zeigt, zumindest stoppen Sie die Animation bei `onStop()` der Aktivität und starten Sie ihn wieder

auf `onStart()` der Aktivität.

`onDraw` Sie keine Berechnungen innerhalb der `onDraw` Methode einer Ansicht durch. Stattdessen sollten Sie die Zeichnung beenden, bevor Sie `onDraw` `invalidate()` aufrufen. Mit dieser Technik können Sie verhindern, dass Bilder in Ihrer Ansicht verschoben werden.

Drehungen

Die grundlegenden Vorgänge einer Ansicht sind Übersetzen, Drehen usw. ... Fast jeder Entwickler ist mit diesem Problem konfrontiert, wenn er Bitmap oder Farbverläufe in seiner benutzerdefinierten Ansicht verwendet. Wenn in der Ansicht eine gedrehte Ansicht angezeigt wird und die Bitmap in dieser benutzerdefinierten Ansicht gedreht werden muss, glauben viele von uns, dass dies teuer wird. Viele denken, dass das Drehen einer Bitmap sehr teuer ist, weil dafür die Pixelmatrix der Bitmap übersetzt werden muss. Aber die Wahrheit ist, dass es nicht so schwer ist! Anstatt die Bitmap zu drehen, drehen Sie einfach die Leinwand selbst!

```
// 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.
```

Zusammengesetzte Ansicht für SVG / VectorDrawable als drawableRight

Hauptmotiv für die Entwicklung dieser zusammengesetzten Ansicht ist, dass unter 5.0 Geräte svg nicht in TextView / EditText drawable unterstützt. Ein weiteres `drawableRight` ist, wir können die `height` und `width` von `drawableRight` in `EditText` . Ich habe es von meinem Projekt getrennt und in einem separaten Modul erstellt.

Modulname: custom_edit_drawable (Kurzname für Präfix c_d_e)

Präfix "c_d_e_", damit die Ressourcen des App-Moduls nicht versehentlich überschrieben werden. Beispiel: Das Präfix "abc" wird von Google in der Support-Bibliothek verwendet.

build.gradle

```
dependencies {
   compile 'com.android.support:appcompat-v7:25.3.1'
}
```

Verwenden Sie AppCompat> = 23

Layoutdatei: 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>
```

Benutzerdefinierte Attribute: 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>
```

Code: 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();
        }
    }
}
```

Beispiel: Verwendung der obigen Ansicht

Layout: activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
```

```
    <com.customeditdrawable.AppEditTextWithDrawable
        android:id="@+id/edt_search_emp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:c_e_d_drawableRightSVG="@drawable/ic_svg_search"
        app:c_e_d_hint="@string/hint_search_here"
        app:c_e_d_textColor="@color/text_color_dark_on_light_bg"
        app:c_e_d_textSize="@dimen/text_size_small" />
</LinearLayout>
```

Tätigkeit: 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);
    }
}
```

Reagieren auf Berührungsereignisse

Viele benutzerdefinierte Ansichten müssen Benutzerinteraktionen in Form von Berührungsereignissen akzeptieren. Sie können Zugriff auf Berührungsereignisse erhalten, indem Sie `onTouchEvent` überschreiben. Es gibt eine Reihe von Aktionen, die Sie herausfiltern können. Die wichtigsten sind

- `ACTION_DOWN` : Dies wird einmal ausgelöst, wenn Ihr Finger die Ansicht zum ersten Mal berührt.
- `ACTION_MOVE` : Dies wird jedes Mal aufgerufen, wenn sich Ihr Finger ein wenig über die Ansicht bewegt. Es wird oft aufgerufen.
- `ACTION_UP` : Dies ist die letzte Aktion, die aufgerufen wird, wenn Sie Ihren Finger vom Bildschirm nehmen.

Sie können Ihrer Ansicht die folgende Methode hinzufügen und dann die Protokollausgabe beobachten, wenn Sie Ihren Finger berühren und um Ihre Ansicht bewegen.

```
@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;
}
```

Lesen Sie weiter:

- Offizielle Dokumentation für Android: Antworten auf Berührungsereignisse

Benutzerdefinierte Ansichten erstellen online lesen:
https://riptutorial.com/de/android/topic/1446/benutzerdefinierte-ansichten-erstellen

Kapitel 44: Benutzerdefinierte Schriftarten

Examples

Eine benutzerdefinierte Schriftart in Ihre App einfügen

1. Gehe zum (Projektordner)
2. Dann app -> src -> main.
3. Legen Sie den Ordner 'Assets -> Schriftarten' im Hauptordner an.
4. Legen Sie Ihre 'fontfile.ttf' in den fonts-Ordner ab.

Eine Schriftart wird initialisiert

```
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");
}
```

Verwenden einer benutzerdefinierten Schriftart in einer TextView

```
public void setFont(TextView textView) {
    textView.setTypeface(myFont);
}
```

Anwenden von Schrift auf TextView durch XML (nicht erforderlicher Java-Code)

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: (Wo werden **res / values platziert**)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="TextViewPlus">
        <attr name="customFont" format="string"/>
    </declare-styleable>
</resources>
```

Wie benutzt man:

```
<?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>
```

Benutzerdefinierte Schriftart im Canvas-Text

Zeichnen von Text in Leinwand mit Ihrer Schriftart aus Assets.

```
Typeface typeface = Typeface.createFromAsset(getAssets(), "fonts/SomeFont.ttf");
Paint textPaint = new Paint();
textPaint.setTypeface(typeface);
canvas.drawText("Your text here", x, y, textPaint);
```

Effizientes Laden von Schriftarten

Das Laden von benutzerdefinierten Schriftarten kann zu einer schlechten Leistung führen. Ich empfehle dringend, diesen kleinen Helfer zu verwenden, der Ihre bereits verwendeten Schriften in einer Hashtabelle speichert / lädt.

```
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;
}
```

}

Verwendungszweck:

```
Typeface typeface = TypefaceUtils.getTypeface(context, "RobotoSlab-Bold.ttf");
setTypeface(typeface);
```

Benutzerdefinierte Schriftart für die gesamte Aktivität

```
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();
    }
}
```

Dann in Ihrer Aktivität in der `onCreate()` -Methode:

```
// Put your font to assets folder...

ReplaceFont.changeDefaultFont(getApplication(), "DEFAULT", "LinLibertine.ttf");
```

Arbeiten mit Schriftarten in Android O

Android O ändert die Arbeitsweise mit Zeichensätzen.

Mit Android O wird eine neue Funktion namens *Fonts in XML eingeführt* , mit der Sie Fonts als Ressourcen verwenden können. Dies bedeutet, dass keine Schriften als Assets gebündelt werden müssen. Schriftarten werden jetzt in einer *R*- Datei kompiliert und stehen automatisch im System als Ressource zur Verfügung.

Um eine neue **Schriftart** hinzuzufügen, müssen Sie Folgendes tun:

- Erstellen Sie ein neues Ressourcenverzeichnis: `res/font` .
- Fügen Sie Ihre Schriftdateien in diesen Ordner ein. Durch das Hinzufügen von `myfont.ttf` können Sie diese Schriftart beispielsweise über `R.font.myfont` .

Sie können auch eine eigene **Schriftfamilie** erstellen, indem Sie die folgende XML-Datei im Verzeichnis `res/font` hinzufügen:

```
<?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>
```

Sie können sowohl die **Schriftartdatei** als auch die **Schriftartfamiliendatei** auf dieselbe Weise verwenden:

- Verwenden Sie **in einer XML-Datei** das Attribut `android:fontFamily` , beispielsweise wie `android:fontFamily` :

  ```
  <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:fontFamily="@font/myfont"/>
  ```

  Oder so:

  ```
  <style name="customfontstyle" parent="@android:style/TextAppearance.Small">
      <item name="android:fontFamily">@font/myfont</item>
  ```

```
</style>
```

- Verwenden Sie **in Ihrem Code** die folgenden Codezeilen:

```
Typeface typeface = getResources().getFont(R.font.myfont);
textView.setTypeface(typeface);
```

Benutzerdefinierte Schriftarten online lesen:
https://riptutorial.com/de/android/topic/3358/benutzerdefinierte-schriftarten

Kapitel 45: Berechnete Ansichtsmaße erhalten

Bemerkungen

Beachten Sie, dass eine `ViewTreeObserver` Instanz, die einer `View` Instanz zugeordnet ist, ungültig werden kann, solange diese `View` noch aktiv ist. In den `View.getViewTreeObserver` Javadocs:

```
// 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()}.
```

Wenn Sie also zuvor einer `ViewTreeObserver` Instanz einen Listener hinzugefügt haben und nun entfernen möchten, können Sie `getViewTreeObserver` für die entsprechende `View` Instanz erneut `ViewTreeObserver` , um eine neue `ViewTreeObserver` Instanz zu erhalten. (Das Überprüfen von `isAlive` für eine vorhandene Instanz ist mehr Arbeit für wenig Nutzen; wenn der `ViewTreeObserver` nicht mehr aktiv ist, erhalten Sie diese neue Referenz trotzdem!)

Examples

Berechnung der anfänglichen Bemaßungsansicht in einer Aktivität

```
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);
    }

}
```

Berechnete Ansichtsmaße erhalten online lesen:
https://riptutorial.com/de/android/topic/115/berechnete-ansichtsma-e-erhalten

Kapitel 46: Berühren Sie Ereignisse

Examples

Wie unterscheidet man zwischen untergeordneten und übergeordneten Ansichtsgruppenereignissen?

1. `onTouchEvents()` für verschachtelte Ansichtsgruppen kann mit dem `boolean` onInterceptTouchEvent verwaltet werden .

Der Standardwert für `OnInterceptTouchEvent` ist false.

Das `onTouchEvent` des übergeordneten `onTouchEvent` wird vor dem des Kindes empfangen. Wenn `OnInterceptTouchEvent` den `OnInterceptTouchEvent` false zurückgibt, sendet es das Bewegungsereignis entlang der Kette an den `OnTouchEvent` Handler des `OnTouchEvent` . Wenn der Wert true zurückgegeben wird, behandelt das übergeordnete Element das Berührungsereignis.

Es kann jedoch Fälle geben, in denen wir möchten, dass einige `OnTouchEvent` Elemente `OnTouchEvent` verwalten und andere von der übergeordneten Ansicht (oder möglicherweise dem übergeordneten Element des `OnTouchEvent`) verwaltet werden.

Dies kann auf mehrere Arten verwaltet werden.

2. Ein `OnInterceptTouchEvent` Element kann vor dem `OnInterceptTouchEvent` des übergeordneten `OnInterceptTouchEvent` geschützt werden, indem das requestDisallowInterceptTouchEvent implementiert wird .

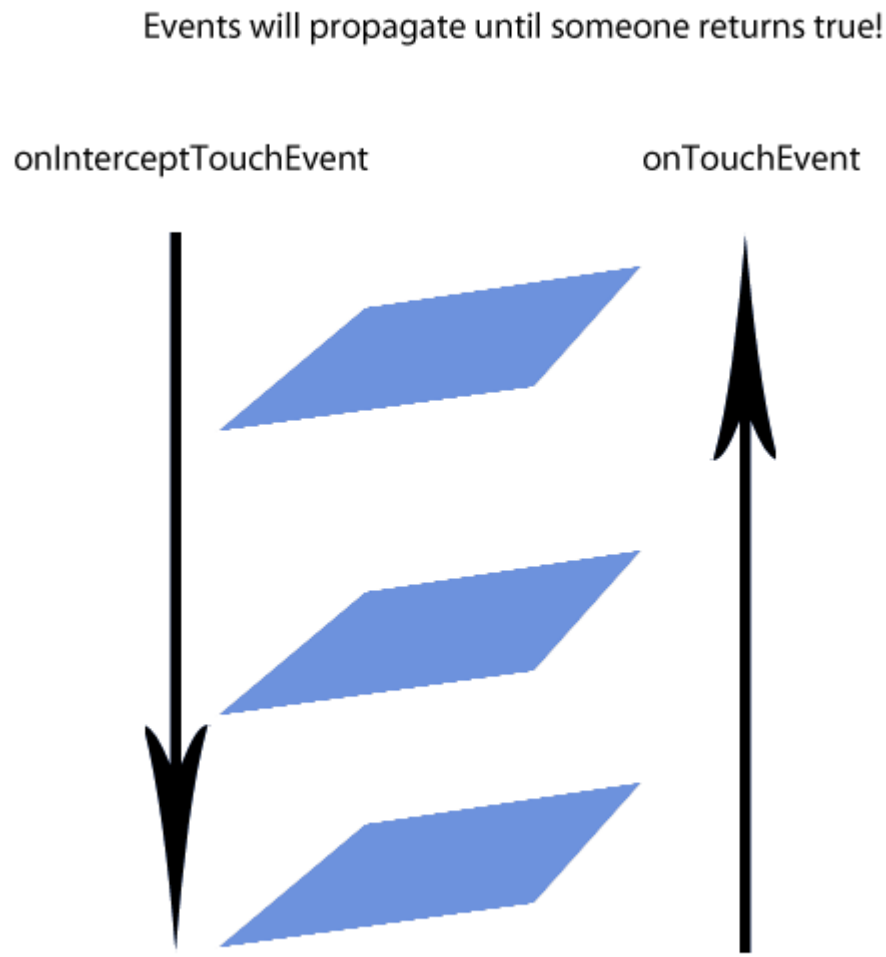
   public void requestDisallowInterceptTouchEvent (boolean disallowIntercept)

Dadurch wird verhindert, dass eine der übergeordneten Ansichten das `OnTouchEvent` für dieses Element verwaltet, wenn für das Element Ereignisprozeduren aktiviert sind.

3.

Wenn `OnInterceptTouchEvent` den `OnInterceptTouchEvent` false hat, wird das `OnTouchEvent` des `OnTouchEvent` Elements ausgewertet. Wenn in den untergeordneten Elementen Methoden vorhanden sind, die die verschiedenen Berührungsereignisse verarbeiten, geben alle zugehörigen deaktivierten Ereignishandler das OnTouchEvent an das übergeordnete Element zurück.

Diese Antwort:
Eine Visualisierung, wie die Ausbreitung von Berührungsereignissen durchläuft:
`parent -> child|parent -> child|parent -> child views.`



Höflichkeit von hier

4. Eine andere Möglichkeit ist, unterschiedliche Werte aus dem `OnInterceptTouchEvent` für das übergeordnete `OnInterceptTouchEvent` .

In diesem Beispiel aus dem Verwalten von Touch-Ereignissen in einer ViewGroup wird veranschaulicht , wie das `OnTouchEvent` des Kindes `OnTouchEvent` wenn der Benutzer einen Bildlauf durchführt.

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;
}
```

Dies ist ein Code aus demselben Link, der zeigt, wie Sie die Parameter des Rechtecks um Ihr Element erstellen:

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);
}
```

Berühren Sie Ereignisse online lesen: https://riptutorial.com/de/android/topic/7167/beruhren-sie-ereignisse

Kapitel 47: Bibliotheksdolch 2: Abhängigkeitsinjektion in Anwendungen

Einführung

Dagger 2 ist, wie auf GitHub erläutert , ein Entwicklungsansatz für die Abhängigkeitsinjektion während der Kompilierung. Um den in Dagger 1.x begonnenen Ansatz zu einem endgültigen Ergebnis zu bringen, beseitigt Dagger 2.x alle Reflexionen und verbessert die Code-Klarheit, indem der traditionelle `ObjectGraph` / `Injector` zugunsten der vom Benutzer angegebenen `@Component` Schnittstellen entfernt wird.

Bemerkungen

1. Bibliotheksaufbau in Anwendung (für Maven-, Gradle- und Java-Projekte)
2. Vorteile des Dragger-Einsatzes
3. Wichtige Links (für Dokumentation und Demos)
4. So integrieren und verwenden Sie Dragger-Komponenten

Dolch 2 API:

Dolch 2 macht einige spezielle Anmerkungen verfügbar:

@Module für die Klassen, deren Methoden Abhängigkeiten bereitstellen

@Liefert die Methoden innerhalb von @Module-Klassen

@Injekt zum Anfordern einer Abhängigkeit (Konstruktor, Feld oder Methode)

@Component ist eine Brückenschnittstelle zwischen Modulen und Injektion

Wichtige Links:

GitHub: https://github.com/google/dagger

UserGuide (Google): https://google.github.io/dagger/users-guide.html

Videos: https://google.github.io/dagger/resources.html

Vogella Tutorial: http://www.vogella.com/tutorials/Dagger/article.html

Codepath-Tutorial: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Examples

Erstellen Sie @Module-Klasse und @Singleton-Annotation für Object

```
import javax.inject.Singleton;
import dagger.Module;
import dagger.Provides;

@Module
public class VehicleModule {

    @Provides @Singleton
    Motor provideMotor(){
        return new Motor();
    }

    @Provides @Singleton
    Vehicle provideVehicle(){
        return new Vehicle(new Motor());
    }
}
```

Jeder Anbieter (oder jede Methode) muss die Annotation `@Provides` und die Klasse die Annotation `@Module` . Die Annotation `@Singleton` zeigt an, dass es nur eine Instanz des Objekts gibt.

Anfordern von Abhängigkeiten in abhängigen Objekten

Nun, da Sie die Anbieter für Ihre verschiedenen Modelle haben, müssen Sie diese anfordern. So wie `Vehicle` `Motor` benötigt, müssen Sie die `@Inject` Annotation im `Vehicle` wie folgt hinzufügen:

```
@Inject
public Vehicle(Motor motor){
    this.motor = motor;
}
```

`@Inject` Annotation `@Inject` können Sie Abhängigkeiten im Konstruktor, in Feldern oder in Methoden anfordern. In diesem Beispiel halte ich die Injektion im Konstruktor.

@Module mit @Inject verbinden

Die Verbindung zwischen dem Anbieter von Abhängigkeiten, `@Module` , und den Klassen, die sie über `@Inject` anfordern, wird mithilfe von `@Component` , einer Schnittstelle:

```
import javax.inject.Singleton;
import dagger.Component;

@Singleton
@Component(modules = {VehicleModule.class})
public interface VehicleComponent {
    Vehicle provideVehicle();
}
```

Für die Annotation `@Component` müssen Sie angeben, welche Module verwendet werden sollen. In diesem Beispiel wird `VehicleModule` verwendet, das in diesem Beispiel definiert ist. Wenn Sie mehrere Module verwenden müssen, fügen Sie diese einfach mit einem Komma als Trennzeichen hinzu.

Verwenden der @Component-Schnittstelle zum Abrufen von Objekten

Da Sie nun jede Verbindung bereit haben, müssen Sie eine Instanz dieser Schnittstelle abrufen und ihre Methoden aufrufen, um das gewünschte Objekt abzurufen:

```
VehicleComponent component = Dagger_VehicleComponent.builder().vehicleModule(new
VehicleModule()).build();
vehicle = component.provideVehicle();
Toast.makeText(this, String.valueOf(vehicle.getSpeed()), Toast.LENGTH_SHORT).show();
```

Wenn Sie versuchen, ein neues Objekt der Schnittstelle mit der `@Component` Annotation zu erstellen, müssen Sie dies mit dem Präfix `Dagger_<NameOfTheComponentInterface>` , in diesem Fall `Dagger_VehicleComponent` , `Dagger_VehicleComponent` und dann die Builder-Methode verwenden, um jedes Modul darin aufzurufen.

Bibliotheksdolch 2: Abhängigkeitsinjektion in Anwendungen online lesen: https://riptutorial.com/de/android/topic/9079/bibliotheksdolch-2--abhangigkeitsinjektion-in-anwendungen

Kapitel 48: Bildansicht

Einführung

ImageView (`android.widget.ImageView`) ist eine Ansicht zum Anzeigen und Bearbeiten von Bildressourcen wie Drawables und Bitmaps.

Einige Effekte, die in diesem Thema behandelt werden, können auf das Bild angewendet werden. Die Bildquelle kann in XML - Datei (eingestellt wird `layout` - Ordner) oder programmatisch in Java - Code.

Syntax

- Die Methode `setImageResource(int resId)` setzt als Inhalt dieser `ImageView` .
- **Verwendung:** `imageView.setImageResource(R.drawable.anyImage)`

Parameter

Parameter	Beschreibung
`resId`	Name Ihrer Image-Datei im Ordner `res` (normalerweise im `drawable`)

Examples

Bildressource einstellen

```
<ImageView
 android:id="@+id/imgExample"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 ...
 />
```

Legen Sie eine Zeichnung als Inhalt von `ImageView` mithilfe des XML-Attributs fest:

```
android:src="@drawable/android2"
```

programmgesteuert einen Zeichnungssatz festlegen:

```
 ImageView imgExample = (ImageView) findViewById(R.id.imgExample);
 imgExample.setImageResource(R.drawable.android2);
```

Alpha einstellen

"alpha" wird verwendet, um die Deckkraft für ein Bild festzulegen.

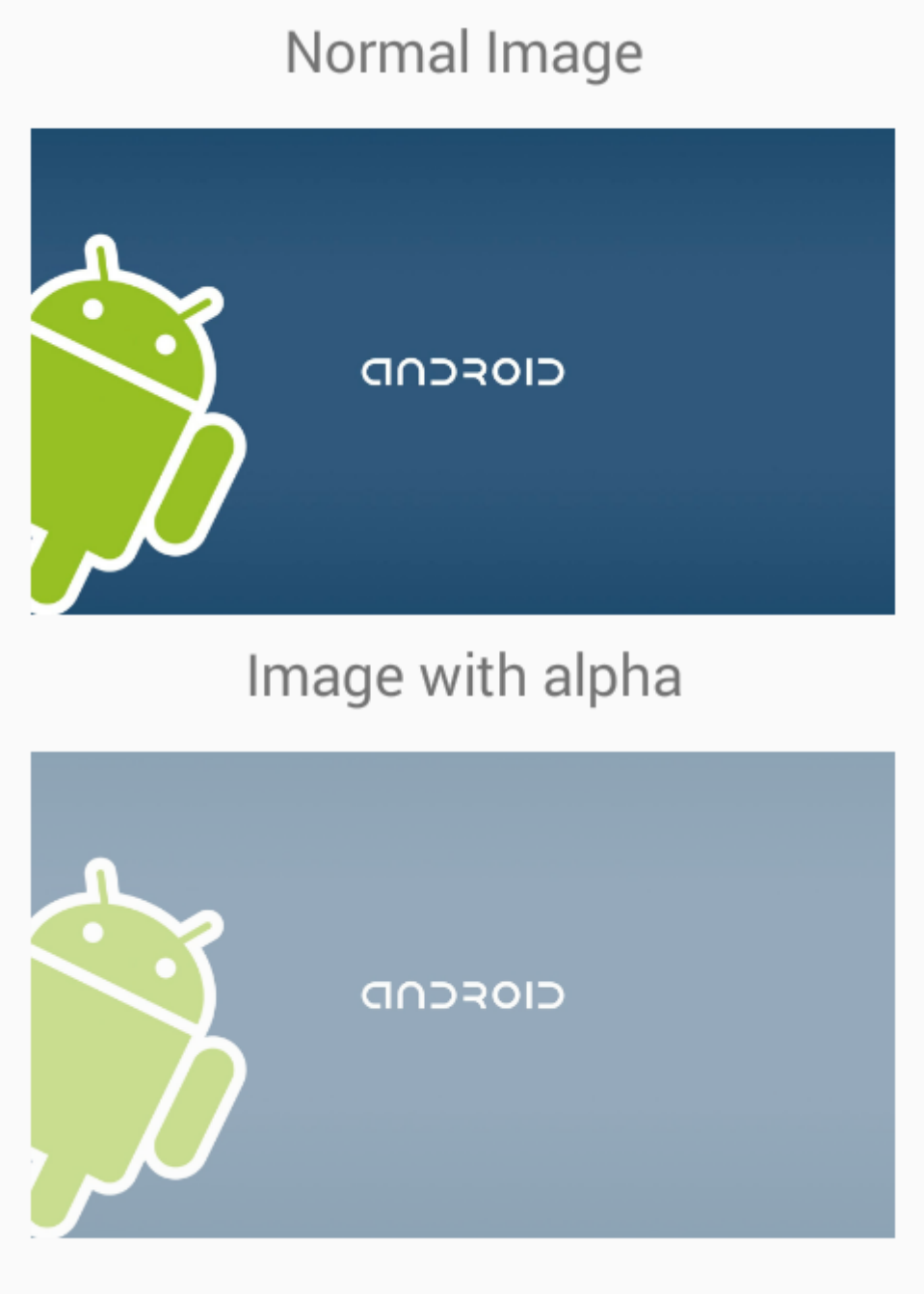
Alpha mit XML-Attribut setzen:

```
android:alpha="0.5"
```

Hinweis: nimmt den Float-Wert von 0 (transparent) bis 1 (vollständig sichtbar).

Alpha programmgesteuert einstellen:

```
imgExample.setAlpha(0.5f);
```



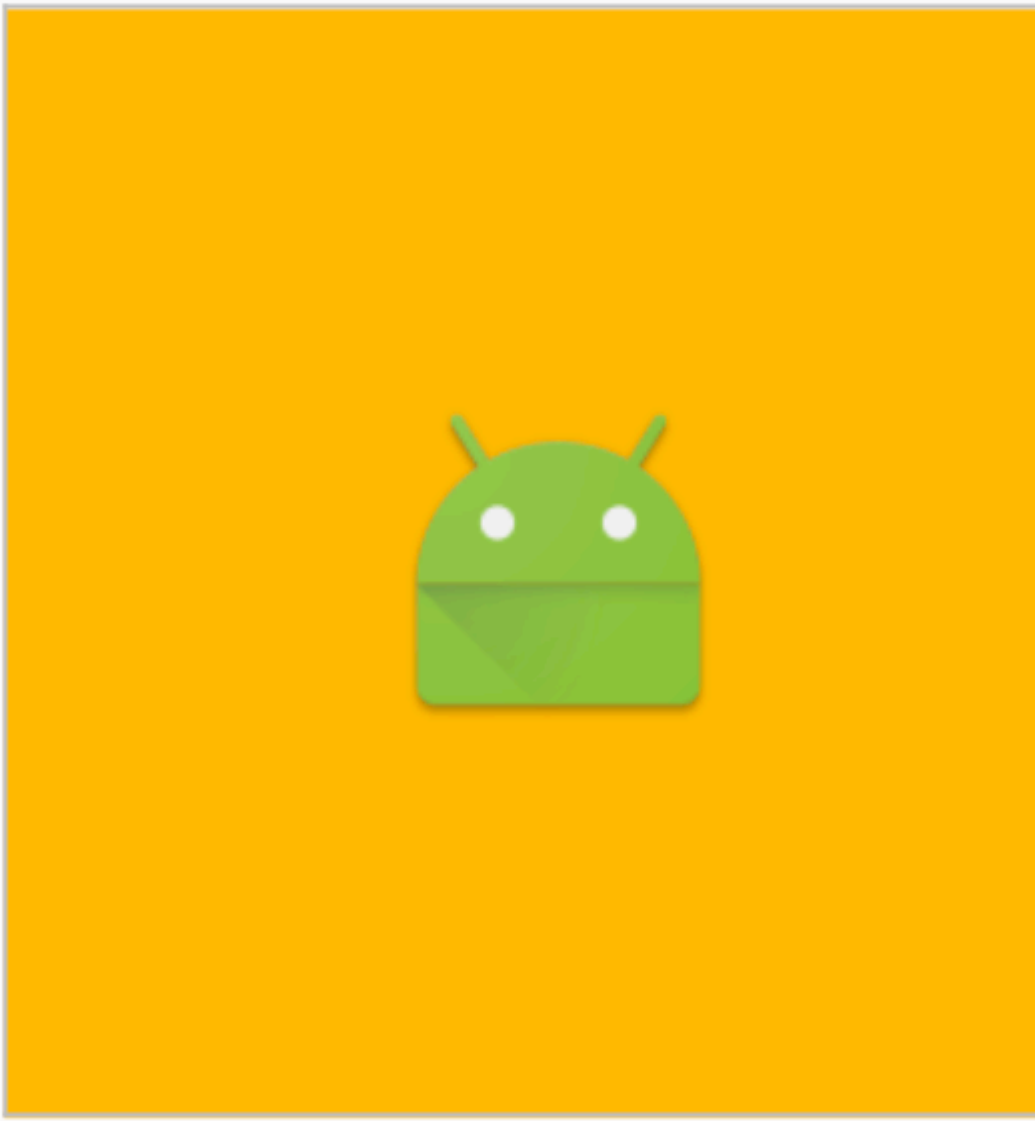
ImageView ScaleType - Center

Das in der ImageView enthaltene Bild stimmt möglicherweise nicht genau mit der für den Container angegebenen Größe überein. In diesem Fall können Sie die Größe des Bildes auf verschiedene Weise ändern.

Center

```
    <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"/>
```

Dadurch wird das Bild nicht verkleinert und innerhalb des Containers zentriert *(Orange = container).*



`ImageView` .

XML-Attribut:

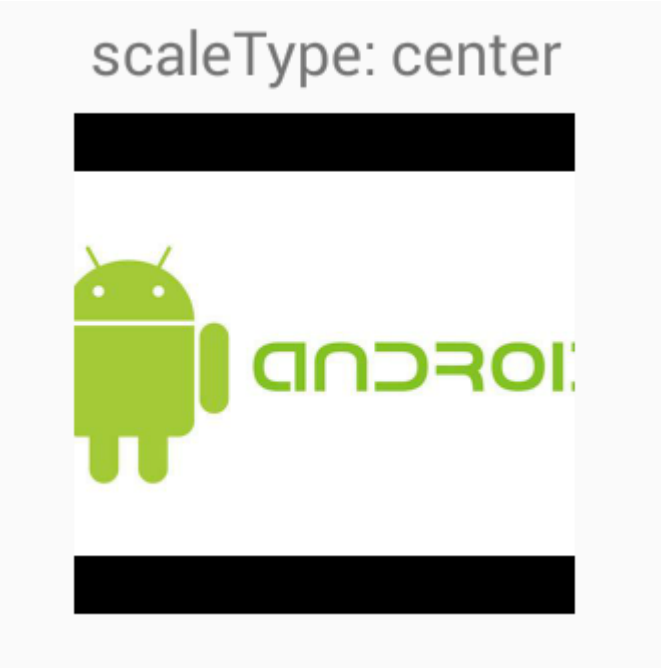
```
android:scaleType="..."
```

Ich werde verschiedene `ImageView` mit einem quadratischen `ImageView` mit schwarzem Hintergrund `ImageView` . In `ImageView` möchten wir einen rechteckigen, weißen Hintergrund `ImageView` .

```
<ImageView
 android:id="@+id/imgExample"
 android:layout_width="200dp"
 android:layout_height="200dp"
 android:background="#000"
 android:src="@drawable/android2"
 android:scaleType="..."/>
```

scaleType muss einer der folgenden Werte sein:

1. `center` : Zentriert das Bild in der Ansicht, führt jedoch keine Skalierung durch.



2. `centerCrop` : `centerCrop` das Bild gleichmäßig (behält das Seitenverhältnis des Bildes bei), so dass beide Abmessungen (Breite und Höhe) des Bildes gleich oder größer als die entsprechende Abmessung der Ansicht sind (minus Auffüllen). Das Bild wird dann in der Ansicht zentriert.

3. `centerInside` : `centerInside` das Bild gleichmäßig (behalten Sie das Seitenverhältnis des Bildes bei), so dass beide Abmessungen (Breite und Höhe) des Bildes gleich oder kleiner als die entsprechende Größe der Ansicht sind (minus Auffüllen). Das Bild wird dann in der Ansicht zentriert.

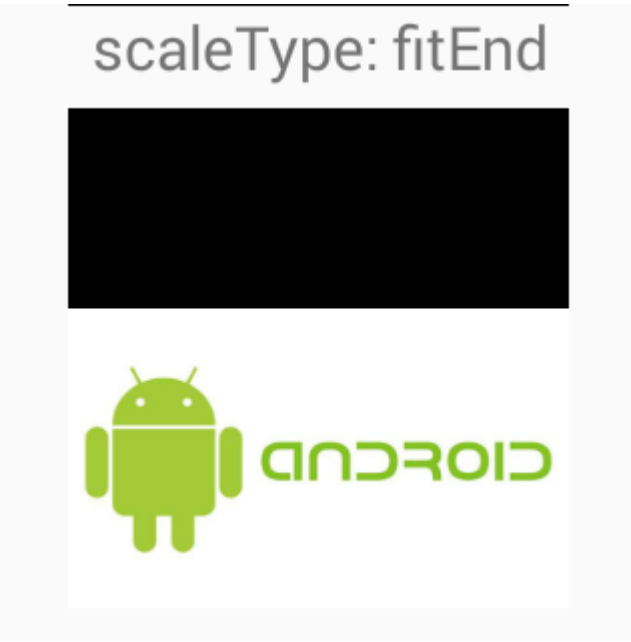
4. `matrix` : Beim Zeichnen mit der Bildmatrix skalieren.

5. `fitXY` : `fitXY` das Bild mit FILL .

6. `fitStart` : `fitStart` das Bild mit START .

7. `fitCenter` : `fitCenter` das Bild mit CENTER .

8. `fitEnd` : `fitEnd` das Bild mit END .



Farbton einstellen

Legen Sie eine Tönungsfarbe für das Bild fest. Standardmäßig wird der Farbton im `SRC_ATOP` Modus `SRC_ATOP` .

Farbton mithilfe des XML-Attributs festlegen:

```
android:tint="#009c38"
```

Hinweis: Muss ein Farbwert sein, in Form von `"#rgb"` , `"#argb"` , `"#rrggbb"` oder `"#aarrggbb"` .

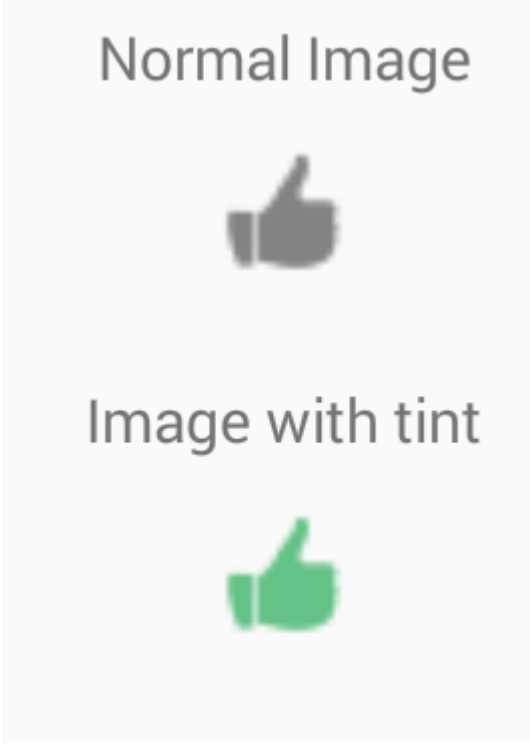
Farbton programmatisch einstellen:

```
imgExample.setColorFilter(Color.argb(255, 0, 156, 38));
```

und Sie können diesen Farbfilter löschen:

```
imgExample.clearColorFilter();
```

Beispiel:



MLRoundedImageView.java

Kopieren und fügen Sie folgende Klasse in Ihr Paket ein:

```
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;
    }
}
```

Verwenden Sie diese Klasse in XML mit dem Paketnamen anstelle von `ImageView`

```
<com.androidbuts.example.MLRoundedImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />
```

Bildansicht online lesen: https://riptutorial.com/de/android/topic/4709/bildansicht

Kapitel 49: Bildkompression

Examples

So komprimieren Sie das Bild ohne Größenänderung

Komprimiertes Bitmap aus der Singleton-Klasse abrufen:

```
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;
    }
}
```

Die Abmessungen sind nach dem Komprimieren der Bitmap gleich .

Wie habe ich überprüft?

```
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());
```

Ausgabe:

```
Before Compress : Dimension: 1080-1452
After Compress : Dimension: 1080-1452
```

Bildkompression online lesen: https://riptutorial.com/de/android/topic/5588/bildkompression

Kapitel 50: Bitmap-Cache

Einführung

Speichereffizientes Bitmap-Caching: Dies ist besonders wichtig, wenn Ihre Anwendung Animationen verwendet, da diese während der GC-Bereinigung angehalten werden und Ihre Anwendung für den Benutzer träge erscheint. Ein Cache ermöglicht die Wiederverwendung von Objekten, deren Erstellung teuer ist. Wenn Sie ein Objekt in den Speicher laden, können Sie sich das als Cache für das Objekt vorstellen. Die Arbeit mit Bitmap in Android ist schwierig. Es ist wichtiger, die Bimap zwischenzuspeichern, wenn Sie sie wiederholt verwenden.

Syntax

- `LruCache<String, Bitmap> mMemoryCache;//declaration of LruCache object.`
- void addBitmapToMemoryCache (String-Schlüssel, Bitmap-Bitmap) {} // Deklaration einer generischen Methode, die Bitmap in den Cache-Speicher hinzufügt
- Bitmap getBitmapFromMemCache (String key) {} // Deklaration einer generischen Methode zum Abrufen von Bimap-Daten aus dem Cache.

Parameter

Parameter	Einzelheiten
Schlüssel	Schlüssel zum Speichern der Bitmap im Speicher-Cache
Bitmap	Bitmap-Wert, der im Arbeitsspeicher zwischengespeichert wird

Examples

Bitmap-Cache mit LRU-Cache

LRU-Cache

Der folgende Beispielcode veranschaulicht eine mögliche Implementierung der LruCache-Klasse zum Zwischenspeichern von Bildern.

```
private LruCache<String, Bitmap> mMemoryCache;
```

Hier ist der String-Wert der Schlüssel für den Bitmap-Wert.

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

Zum Hinzufügen einer Bitmap zum Speichercache

```
public void addBitmapToMemoryCache(String key, Bitmap bitmap) {
if (getBitmapFromMemCache(key) == null) {
        mMemoryCache.put(key, bitmap);
    }
}
```

Zum Abrufen einer Bitmap aus dem Cache-Speicher

```
public Bitmap getBitmapFromMemCache(String key) {
    return mMemoryCache.get(key);
}
```

Um Bitmap in imageview zu laden, verwenden Sie einfach **getBitmapFromMemCache ("Pass key").**

Bitmap-Cache online lesen: https://riptutorial.com/de/android/topic/9901/bitmap-cache

Kapitel 51: Bitmaps effektiv laden

Einführung

Dieses Thema konzentriert sich hauptsächlich auf das effektive Laden der Bitmaps in Android-Geräten.

Beim Laden einer Bitmap stellt sich die Frage, woher sie geladen wird. Hier besprechen wir, wie Sie die Bitmap von Resource in das Android-Gerät laden. zB aus der Galerie.

Wir werden dies anhand von Beispielen durchgehen, die im Folgenden erläutert werden.

Syntax

- `<uses-permission>` **->** Tag, das für die Berechtigung verwendet wird.
- `android:name` **->** Ein Attribut, das verwendet wird, um der Berechtigung, die wir anfordern `android:name` , einen Namen zu geben.
- `android.permission.READ_EXTERNAL_STORAGE` **->** `android.permission.READ_EXTERNAL_STORAGE`
- Beispiel "android.permission.CAMERA" oder "android.permission.READ_CONTACTS"

Examples

Laden Sie das Bild von der Ressource vom Android-Gerät. Absichten verwenden.

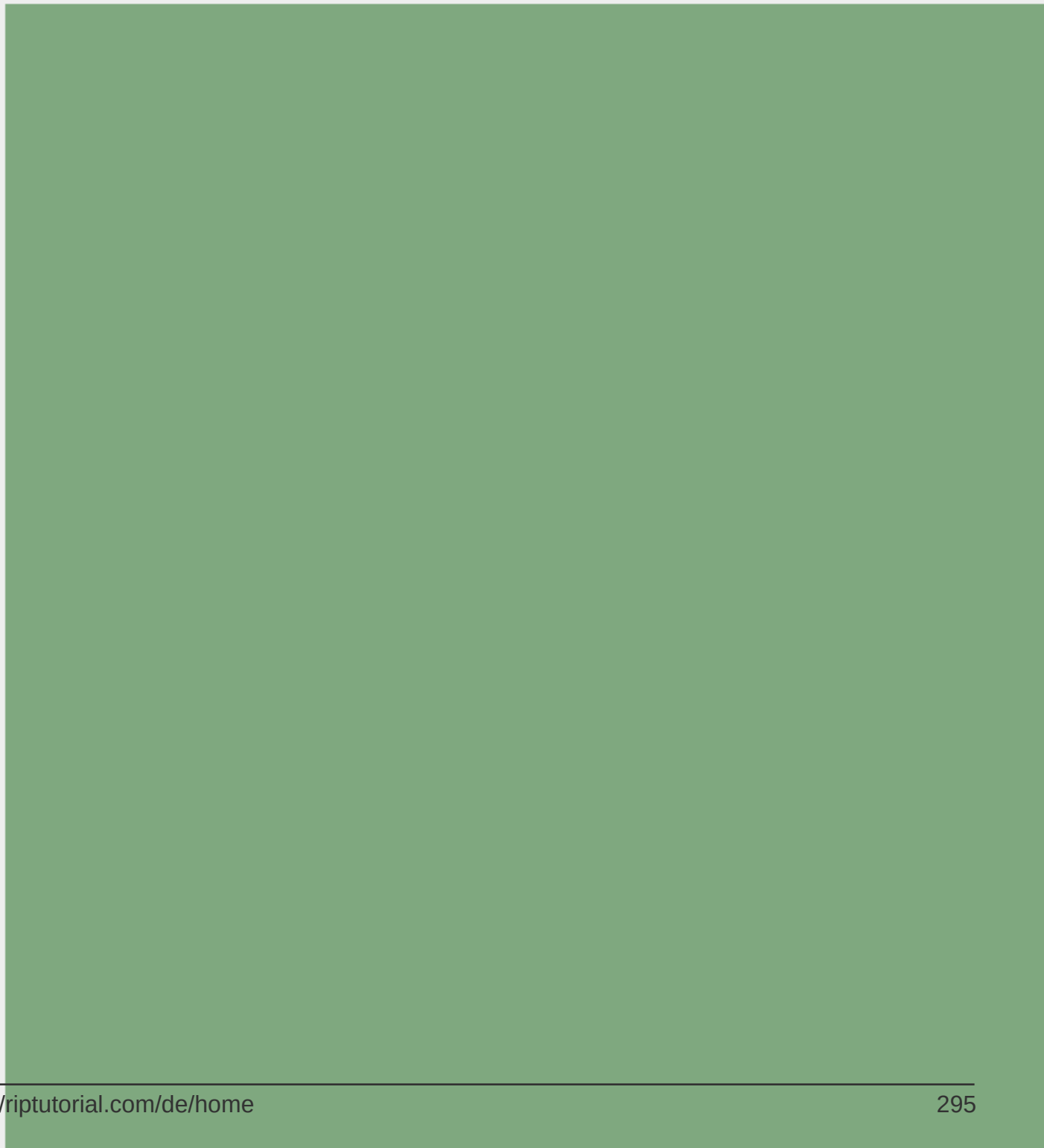
Verwenden von Absichten zum Laden des Bildes aus der Galerie.

1. Zunächst benötigen Sie die Erlaubnis

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

2. Verwenden Sie den folgenden Code, um das Layout wie folgt festzulegen.

LoadImageFrmGallery



https://riptutorial.com/de/android/topic/10902/bitmaps-effektiv-laden

Kapitel 52: Bluetooth Low Energy

Einführung

Diese Dokumentation ist als Erweiterung der Originaldokumentation gedacht und konzentriert sich auf die neueste Bluetooth LE-API, die in Android 5.0 (API 21) eingeführt wurde. Es werden sowohl die zentralen als auch die peripheren Rollen behandelt sowie der Start des Scan- und Werbevorgangs.

Examples

BLE-Geräte suchen

Die folgenden Berechtigungen sind erforderlich, um die Bluetooth-APIs zu verwenden:

```
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
```

Wenn Sie auf Geräte mit Android 6.0 (**API Level 23**) oder höher zielen und Scan- / Werbevorgänge durchführen möchten, benötigen Sie eine Standortberechtigung:

```
android.permission.ACCESS_FINE_LOCATION
```

oder

```
android.permission.ACCESS_COARSE_LOCATION
```

Hinweis.- Bei Geräten mit Android 6.0 (API Level 23) oder höher müssen auch die Ortungsdienste aktiviert sein.

Zum Starten von Scan- / Werbevorgängen ist ein BluetoothAdapter-Objekt erforderlich:

```
BluetoothManager bluetoothManager = (BluetoothManager)
context.getSystemService(Context.BLUETOOTH_SERVICE);
bluetoothAdapter = bluetoothManager.getAdapter();
```

Die `startScan (ScanCallback callback)` Methode `startScan (ScanCallback callback)` der BluetoothLeScanner-Klasse ist die einfachste Methode, um einen Scanvorgang zu starten. Ein `ScanCallback` Objekt ist erforderlich, um Ergebnisse zu erhalten:

```
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());
       }
```

```
    });
```

Verbindung zu einem GATT-Server

Wenn Sie ein gewünschtes BluetoothDevice-Objekt erkannt haben, können Sie eine Verbindung mit ihm herstellen, indem Sie die Methode `connectGatt()` verwenden, die als Parameter ein Context-Objekt verwendet. Ein boolescher `connectGatt()` , der angibt, ob automatisch eine Verbindung zum BLE-Gerät hergestellt werden soll, und eine BluetoothGattCallback-Referenz, bei der Verbindungsereignisse und Clientvorgänge ausgeführt werden Ergebnisse werden geliefert:

```
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        device.connectGatt(context, false, bluetoothGattCallback,
BluetoothDevice.TRANSPORT_AUTO);
    } else {
        device.connectGatt(context, false, bluetoothGattCallback);
    }
```

`onConnectionStateChange` in BluetoothGattCallback, um Verbindungsereignisse und Verbindungsereignisse zu empfangen:

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

Schreiben und Lesen aus Merkmalen

Sobald Sie mit einem Gatt-Server verbunden sind, werden Sie mit ihm interagieren, indem Sie von den Eigenschaften des Servers aus schreiben und lesen. Dazu müssen Sie zunächst herausfinden, welche Dienste auf diesem Server verfügbar sind und welche Merkmale in den einzelnen Diensten verfügbar sind:

```
 @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
                    }
                }
            }
          }
```

Ein grundlegender Schreibvorgang lautet wie folgt:

```
characteristic.setValue(newValue);
characteristic.setWriteType(BluetoothGattCharacteristic.WRITE_TYPE_DEFAULT);
gatt.writeCharacteristic(characteristic);
```

Wenn der Schreibvorgang abgeschlossen ist, wird die `onCharacteristicWrite` Methode Ihres `BluetoothGattCallback` aufgerufen:

```
    @Override
    public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
        super.onCharacteristicWrite(gatt, characteristic, status);
        Log.d(TAG, "Characteristic " + characteristic.getUuid() + " written);
    }
```

Ein grundlegender Schreibvorgang lautet wie folgt:

```
gatt.readCharacteristic(characteristic);
```

Wenn der Schreibvorgang abgeschlossen ist, wird die `onCharacteristicRead` Methode Ihres `BluetoothGattCallback` aufgerufen:

```
@Override
public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
  super.onCharacteristicRead(gatt, characteristic, status);
  byte[] value = characteristic.getValue();
  }
```

Abonnieren von Benachrichtigungen vom Gatt-Server

Sie können anfordern, vom Gatt-Server benachrichtigt zu werden, wenn der Wert eines Merkmals geändert wurde:

```
gatt.setCharacteristicNotification(characteristic, true);
BluetoothGattDescriptor descriptor = characteristic.getDescriptor(
    UUID.fromString("00002902-0000-1000-8000-00805f9b34fb");
descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
```

```
mBluetoothGatt.writeDescriptor(descriptor);
```

Alle Benachrichtigungen vom Server werden in der `onCharacteristicChanged` Methode Ihres BluetoothGattCallbacks empfangen:

```
    @Override
    public void onCharacteristicChanged(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic) {
        super.onCharacteristicChanged(gatt, characteristic);
        byte[] newValue = characteristic.getValue();
}
```

Werbung für ein BLE-Gerät

Sie können Bluetooth LE Advertising verwenden, um Datenpakete an alle in der Nähe befindlichen Geräte zu senden, ohne zuerst eine Verbindung herstellen zu müssen. Beachten Sie, dass die Ankündigungsdaten auf 31 Bytes begrenzt sind. Werbung für Ihr Gerät ist auch der erste Schritt, um anderen Benutzern die Verbindung zu Ihnen zu ermöglichen.

Da nicht alle Geräte Bluetooth LE Advertising unterstützen, müssen Sie zunächst prüfen, ob Ihr Gerät über alle erforderlichen Voraussetzungen verfügt, um dies zu unterstützen. Anschließend können Sie ein `BluetoothLeAdvertiser` Objekt initialisieren und damit den Werbevorgang starten:

```
        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);
        }
```

Verwendung eines Gatt-Servers

Damit Ihr Gerät als Peripheriegerät fungieren kann, müssen Sie zuerst einen `BluetoothGattServer` öffnen und mit mindestens einem `BluetoothGattService` und einer `BluetoothGattCharacteristic` .

```
BluetoothGattServer server=bluetoothManager.openGattServer(context,
bluetoothGattServerCallback);

BluetoothGattService service = new BluetoothGattService(SERVICE_UUID,
BluetoothGattService.SERVICE_TYPE_PRIMARY);
```

Dies ist ein Beispiel für eine BluetoothGattCharacteristic mit vollständigen Schreib-, Lese- und Benachrichtigungsberechtigungen. Je nach Ihren Anforderungen möchten Sie möglicherweise die Berechtigungen, die Sie dieses Merkmal gewähren, genau einstellen:

```
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);
```

Der `BluetoothGattServerCallback` ist für den Empfang aller Ereignisse im Zusammenhang mit Ihrem `BluetoothGattServer` :

```
BluetoothGattServerCallback bluetoothGattServerCallback= new BluetoothGattServerCallback() {
                @Override
                public void onConnectionStateChange(BluetoothDevice device, int status, int
newState) {
                    super.onConnectionStateChange(device, status, newState);
                }

                @Override
                public void onCharacteristicReadRequest(BluetoothDevice device, int requestId,
int offset, BluetoothGattCharacteristic characteristic) {
                    super.onCharacteristicReadRequest(device, requestId, offset,
characteristic);
                }

                @Override
                public void onCharacteristicWriteRequest(BluetoothDevice device, int
requestId, BluetoothGattCharacteristic characteristic, boolean preparedWrite, boolean
responseNeeded, int offset, byte[] value) {
                    super.onCharacteristicWriteRequest(device, requestId, characteristic,
```

```
preparedWrite, responseNeeded, offset, value);
                }

                @Override
                public void onDescriptorReadRequest(BluetoothDevice device, int requestId, int
offset, BluetoothGattDescriptor descriptor) {
                    super.onDescriptorReadRequest(device, requestId, offset, descriptor);
                }

                @Override
                public void onDescriptorWriteRequest(BluetoothDevice device, int requestId,
BluetoothGattDescriptor descriptor, boolean preparedWrite, boolean responseNeeded, int offset,
byte[] value) {
                    super.onDescriptorWriteRequest(device, requestId, descriptor,
preparedWrite, responseNeeded, offset, value);
                }
```

Wenn Sie eine Anforderung zum Schreiben / Lesen eines Merkmals oder eines Deskriptors erhalten, müssen Sie eine Antwort senden, damit die Anforderung erfolgreich abgeschlossen werden kann:

```
@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);
}
```

Bluetooth Low Energy online lesen: https://riptutorial.com/de/android/topic/10020/bluetooth-low-energy

Kapitel 53: Bluetooth und Bluetooth LE API

Bemerkungen

Bluetooth Classic ist ab Android 2.0 (API Level 5) und höher verfügbar. Bluetooth LE ist ab Android 4.3 (API Level 18) und höher verfügbar.

Examples

Berechtigungen

Fügen Sie der Manifestdatei diese Berechtigung hinzu, um die Bluetooth-Funktionen in Ihrer Anwendung zu verwenden:

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

Wenn Sie die Geräteerkennung starten oder die Bluetooth-Einstellungen bearbeiten müssen, müssen Sie auch diese Berechtigung hinzufügen:

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

Für das Android-API-Level 23 und höher ist ein Standortzugriff erforderlich:

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

* Siehe auch die Berechtigungen Thema , um weitere Informationen darüber , wie geeignete Berechtigungen zu verwenden.

Prüfen Sie, ob Bluetooth aktiviert ist

```
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
        }
    }
}
```

Machen Sie das Gerät auffindbar

```
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
        }
    }
}
```

Finden Sie Bluetooth-Geräte in der Nähe

Deklarieren Sie zuerst einen `BluetoothAdapter` .

```
BluetoothAdapter mBluetoothAdapter;
```

Erstellen Sie nun einen `BroadcastReceiver` für `ACTION_FOUND`

```
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();

    //Device found
    if (BluetoothDevice.ACTION_FOUND.equals(action))
    {
        // Get the BluetoothDevice object from the Intent
```

```
        BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
        // Add the name and address to an array adapter to show in a list
        mArrayAdapter.add(device.getName() + "\n" + device.getAddress());
    }
  }
};
```

Registrieren Sie den `BroadcastReceiver`

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

Starten Sie dann die `startDiscovery` den Bluetooth-Geräten in der Nähe, indem Sie `startDiscovery` aufrufen

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

Vergessen Sie nicht, den `BroadcastReceiver` in `onDestroy`

```
unregisterReceiver(mReceiver);
```

Verbinden Sie sich mit einem Bluetooth-Gerät

Nachdem Sie das Bluetooth-Gerät erworben haben, können Sie damit kommunizieren. Diese Art der Kommunikation erfolgt mithilfe von Socket-Input \ Output-Streams:

Dies sind die grundlegenden Schritte für die Einrichtung der Bluetooth-Kommunikation:

1) Socket initialisieren:

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

2) An Buchse anschließen:

```
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) Socket-Input \ Output-Streams erhalten

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

Eingangsstrom - Wird als eingehender Datenkanal verwendet (Daten vom angeschlossenen Gerät empfangen)

Ausgangsstrom - Wird als ausgehender Datenkanal verwendet (Daten an angeschlossenes Gerät senden)

Nach Abschluss des dritten Schritts können wir Daten zwischen beiden Geräten mit zuvor initialisierten Streams empfangen und senden:

1) Daten empfangen (Lesen vom Socket-Eingangsstrom)

```
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) Senden von Daten (Schreiben in den Ausgabestrom)

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

- Die Verbindungs-, Lese- und Schreibfunktionalität sollte natürlich in einem dedizierten

Thread ausgeführt werden.
- Sockets und Stream-Objekte müssen sein

Finden Sie in der Nähe befindliche Bluetooth Low Energy-Geräte

Die BluetoothLE-API wurde in API 18 eingeführt. Die Art und Weise des Scannens von Geräten hat sich jedoch in API 21 geändert. Die Suche nach Geräten muss mit der Definition der zu untersuchenden Dienst-UUID beginnen (entweder offiziell angenommene 16-Bit-UUID oder proprietäre). . Dieses Beispiel zeigt, wie Sie eine API-unabhängige Suche nach BLE-Geräten vornehmen können:

1. Erstellen Sie ein Bluetooth-Gerätemodell:

```
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. Definieren Sie die Bluetooth-Scanschnittstelle:

```
public interface ScanningAdapter {

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

3. Scanfabrikklasse erstellen:

```
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. Erstellen Sie die Factory-Implementierung für jede API:

API 18:

```
import android.annotation.TargetApi;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.os.Build;
import android.os.Parcelable;
import android.util.Log;

import bluetooth.model.BTDevice;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

@TargetApi(Build.VERSION_CODES.JELLY_BEAN_MR2)
public class JellyBeanBluetoothLEScanAdapter implements ScanningAdapter{
    BluetoothAdapter bluetoothAdapter;
    ScanCallback mCallback;

    List<BTDevice> mBluetoothDeviceList;

    public JellyBeanBluetoothLEScanAdapter() {
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        mCallback = new ScanCallback();
        mBluetoothDeviceList = new ArrayList<>();
    }

    @Override
    public void startScanning(String[] uuids) {
        if (uuids == null || uuids.length == 0) {
            return;
        }
        UUID[] uuidList = createUUIDList(uuids);
        bluetoothAdapter.startLeScan(uuidList, mCallback);
    }
```

```
    private UUID[] createUUIDList(String[] uuids) {
        UUID[] uuidList = new UUID[uuids.length];
        for (int i = 0 ; i < uuids.length ; ++i) {
            String uuid = uuids[i];
            if (uuid == null) {
                continue;
            }
            uuidList[i] = UUID.fromString(uuid);
        }
        return uuidList;
    }

    @Override
    public void stopScanning() {
        bluetoothAdapter.stopLeScan(mCallback);
    }

    @Override
    public List<BTDevice> getFoundDeviceList() {
        return mBluetoothDeviceList;
    }

    private class ScanCallback implements BluetoothAdapter.LeScanCallback {

        @Override
        public void onLeScan(BluetoothDevice device, int rssi, byte[] scanRecord) {
            if (isAlreadyAdded(device)) {
                return;
            }
            BTDevice btDevice = new BTDevice();
            btDevice.setName(new String(device.getName()));
            btDevice.setAddress(device.getAddress());
            mBluetoothDeviceList.add(btDevice);
            Log.d("Bluetooth discovery", device.getName() + " " + device.getAddress());
            Parcelable[] uuids = device.getUuids();
            String uuid = "";
            if (uuids != null) {
                for (Parcelable ep : uuids) {
                    uuid += ep + " ";
                }
                Log.d("Bluetooth discovery", device.getName() + " " + device.getAddress() + "
" + uuid);
            }
        }

        private boolean isAlreadyAdded(BluetoothDevice bluetoothDevice) {
            for (BTDevice device : mBluetoothDeviceList) {
                String alreadyAddedDeviceMACAddress = device.getAddress();
                String newDeviceMACAddress = bluetoothDevice.getAddress();
                if (alreadyAddedDeviceMACAddress.equals(newDeviceMACAddress)) {
                    return true;
                }
            }
            return false;
        }
    }
}
```

API 21:

```
import android.annotation.TargetApi;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.le.BluetoothLeScanner;
import android.bluetooth.le.ScanFilter;
import android.bluetooth.le.ScanResult;
import android.bluetooth.le.ScanSettings;
import android.os.Build;
import android.os.ParcelUuid;

import bluetooth.model.BTDevice;

import java.util.ArrayList;
import java.util.List;

@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class LollipopBluetoothLEScanAdapter implements ScanningAdapter {
    BluetoothLeScanner bluetoothLeScanner;
    ScanCallback mCallback;

    List<BTDevice> mBluetoothDeviceList;

    public LollipopBluetoothLEScanAdapter() {
        bluetoothLeScanner = BluetoothAdapter.getDefaultAdapter().getBluetoothLeScanner();
        mCallback = new ScanCallback();
        mBluetoothDeviceList = new ArrayList<>();
    }

    @Override
    public void startScanning(String[] uuids) {
        if (uuids == null || uuids.length == 0) {
            return;
        }
        List<ScanFilter> filterList = createScanFilterList(uuids);
        ScanSettings scanSettings = createScanSettings();
        bluetoothLeScanner.startScan(filterList, scanSettings, mCallback);
    }

    private List<ScanFilter> createScanFilterList(String[] uuids) {
        List<ScanFilter> filterList = new ArrayList<>();
        for (String uuid : uuids) {
            ScanFilter filter = new ScanFilter.Builder()
                    .setServiceUuid(ParcelUuid.fromString(uuid))
                    .build();
            filterList.add(filter);
        };
        return filterList;
    }

    private ScanSettings createScanSettings() {
        ScanSettings settings = new ScanSettings.Builder()
                .setScanMode(ScanSettings.SCAN_MODE_BALANCED)
                .build();
        return settings;
    }

    @Override
    public void stopScanning() {
        bluetoothLeScanner.stopScan(mCallback);
    }

    @Override
```

```
    public List<BTDevice> getFoundDeviceList() {
        return mBluetoothDeviceList;
    }

    public class ScanCallback extends android.bluetooth.le.ScanCallback {

        @Override
        public void onScanResult(int callbackType, ScanResult result) {
            super.onScanResult(callbackType, result);
            if (result == null) {
                return;
            }
            BTDevice device = new BTDevice();
            device.setAddress(result.getDevice().getAddress());
            device.setName(new
StringBuffer(result.getScanRecord().getDeviceName()).toString());
            if (device == null || device.getAddress() == null) {
                return;
            }
            if (isAlreadyAdded(device)) {
                return;
            }
            mBluetoothDeviceList.add(device);
        }

        private boolean isAlreadyAdded(BTDevice bluetoothDevice) {
            for (BTDevice device : mBluetoothDeviceList) {
                String alreadyAddedDeviceMACAddress = device.getAddress();
                String newDeviceMACAddress = bluetoothDevice.getAddress();
                if (alreadyAddedDeviceMACAddress.equals(newDeviceMACAddress)) {
                    return true;
                }
            }
            return false;
        }
    }
}
```

5. Erhalten Sie die Liste der gefundenen Geräte durch Anrufen:

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

wait few seconds...

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

Bluetooth und Bluetooth LE API online lesen:
https://riptutorial.com/de/android/topic/2462/bluetooth-und-bluetooth-le-api

Kapitel 54: BottomNavigationView

Einführung

Die untere Navigationsansicht befand sich schon seit einiger Zeit in den Richtlinien für die Materialentwicklung, aber es war nicht einfach für uns, sie in unsere Apps zu implementieren.

Einige Anwendungen haben eigene Lösungen entwickelt, während andere sich auf Open-Source-Bibliotheken von Drittanbietern verlassen haben, um die Arbeit zu erledigen.

Jetzt wird in der Design-Support-Bibliothek diese untere Navigationsleiste hinzugefügt. Lassen Sie uns einen Blick darauf werfen, wie wir sie verwenden können!

Bemerkungen

Stellt eine standardmäßige untere Navigationsleiste für die Anwendung dar. Es ist eine Implementierung der Materialnavigation.

Links:

- Offizieller Javadoc

Examples

Grundlegende Implementierung

`BottomNavigationView` folgendermaßen vor, um die `BottomNavigationView` hinzuzufügen:

1. Fügen Sie in Ihrem `build.gradle` die **Abhängigkeit hinzu** :

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

2. Fügen Sie die `BottomNavigationView` **Ihrem Layout hinzu** :

```
<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. Erstellen Sie das **Menü** , um die Ansicht aufzufüllen:

```
<!-- 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. Hängen Sie **einen Listener** für die Klickereignisse an:

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

Checkout-Demo-Code bei BottomNavigation-Demo

Anpassung von BottomNavigationView

Hinweis: Ich ***`BottomNavigationView`*** *davon aus, dass Sie wissen, wie* ***`BottomNavigationView`*** *.*

In diesem Beispiel werde ich erläutern, wie der Selektor für **`BottomNavigationView`** . Sie können also auf der Benutzeroberfläche für Symbole und Texte angeben.

Erstellen Sie eine drawable `bottom_navigation_view_selector.xml` als

```
<?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>
```

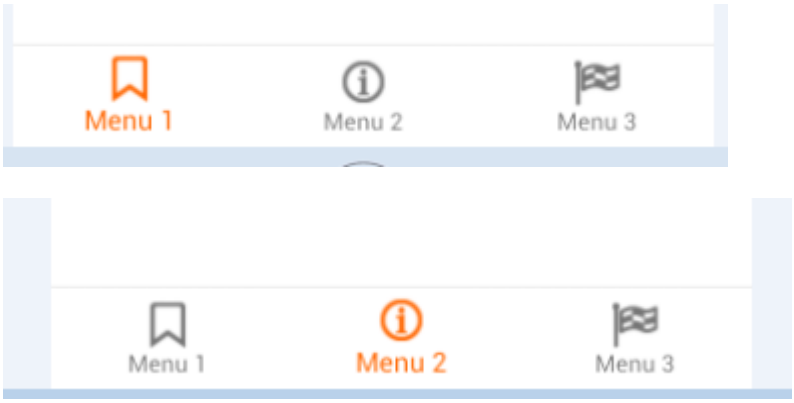
Verwenden Sie die folgenden Attribute in **`BottomNavigationView`** in der Layoutdatei

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

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

> In obigem Beispiel habe ich denselben Selektor `bottom_navigation_view_selector` für `app:itemIconTint` und `app:itemTextColor` , um die Text- und `app:itemTextColor` zu halten. Wenn Ihr Design jedoch eine andere Farbe für Text und Symbole hat, können Sie zwei verschiedene Selektoren definieren und verwenden.

Die Ausgabe ist ähnlich wie unten



Umgang mit aktivierten / deaktivierten Zuständen

Selektor für Menüelement aktivieren / deaktivieren.

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" />
```

Erlaubt mehr als 3 Menüs

Dieses Beispiel ist ausnahmslos eine Problemumgehung, da ein Verhalten, das als ShiftMode bekannt ist, derzeit nicht deaktiviert werden kann.

Erstellen Sie eine Funktion als solche.

```
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);
    }
}
```

Dies deaktiviert das Verschiebungsverhalten des Menüs, wenn die Elementanzahl 3 Nummern überschreitet.

VERWENDUNGSZWECK

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

Proguard-Problem : Fügen Sie die folgende Zeile der Proguard-Konfigurationsdatei hinzu. Andernfalls funktioniert dies nicht.

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

Alternativ können Sie eine Klasse erstellen und von dort auf diese Methode zugreifen. Siehe Originalantwort hier

HINWEIS : Dies ist ein auf Reflection basierender **HOTFIX** . Bitte aktualisieren Sie diesen, sobald die Support-Bibliothek von Google mit einem direkten Funktionsaufruf aktualisiert wird.

BottomNavigationView online lesen:
https://riptutorial.com/de/android/topic/7565/bottomnavigationview

Kapitel 55: Buttermesser

Einführung

Butterknife ist ein Ansichtsbindungswerkzeug, das Anmerkungen verwendet, um Boilerplate-Code für uns zu generieren. Dieses Tool wurde von Jake Wharton at Square entwickelt und dient im Wesentlichen zum Speichern von sich wiederholenden Codezeilen wie `findViewById(R.id.view)` beim Umgang mit Ansichten.

Um klar zu sein, Butterknife ist **keine Abhängigkeitsinjektionsbibliothek** . Butterknife injiziert Code zur Kompilierzeit. Es ist der Arbeit von Android Annotations sehr ähnlich.

Bemerkungen

Buttermesser

Feld- und Methodenbindung für Android-Ansichten, in denen Anmerkungsverarbeitung verwendet wird, um für Sie Boilerplate-Code zu generieren.

- Beseitigen Sie die Aufrufe von findViewById mithilfe von @BindView für Felder.
- Gruppieren Sie mehrere Ansichten in einer Liste oder einem Array. Sie können alle Aktionen gleichzeitig mit Aktionen, Setters oder Eigenschaften ausführen.
- Beseitigen Sie anonyme Innenklassen für Zuhörer, indem Sie Methoden mit @OnClick und anderen kommentieren.
- Beseitigen Sie Ressourcensuchen, indem Sie Ressourcennotizen für Felder verwenden.

Weitere Informationen: http://jakewharton.github.io/butterknife/

Lizenz

Examples

ButterKnife in Ihrem Projekt konfigurieren

Konfigurieren Sie Ihr `build.gradle` auf `build.gradle` , um das `build.gradle` für `android-apt` :

```
buildscript {
   repositories {
      mavenCentral()
   }

   dependencies {
      classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
   }
}
```

`build.gradle` dann das `android-apt` Plugin in Ihrem `build.gradle` auf `build.gradle` und fügen Sie die ButterKnife-Abhängigkeiten hinzu:

```
apply plugin: 'android-apt'

android {
    ...
}

dependencies {
      compile 'com.jakewharton:butterknife:8.5.1'
      annotationProcessor 'com.jakewharton:butterknife-compiler:8.5.1'
}
```

Hinweis: Wenn Sie den neuen Jack-Compiler mit Version 2.2.0 oder neuer verwenden, benötigen Sie das `android-apt` und können stattdessen apt durch `annotationProcessor` ersetzen, wenn Sie die Compilerabhängigkeit deklarieren.

Um ButterKnife-Anmerkungen zu verwenden, sollten Sie nicht vergessen, sie in `onCreate()` Ihrer Aktivitäten oder `onCreateView()` Ihrer Fragmente zu `onCreateView()` :

```
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;
  }

}
```

Snapshots der Entwicklungsversion sind im Sonatype-Repository für Snapshots verfügbar.

Nachfolgend finden Sie die zusätzlichen Schritte, die Sie ausführen müssen, um ButterKnife in einem Bibliotheksprojekt zu verwenden

Um ButterKnife in einem Bibliotheksprojekt zu verwenden, fügen Sie das Plugin zu Ihrem `build.gradle` auf Projektebene `build.gradle` :

```
buildscript {
    dependencies {
        classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
    }
}
```

… Und bewerben Sie sich dann auf Ihr Modul, indem Sie diese Zeilen oben in `build.gradle` Bibliotheksebene `build.gradle` :

```
apply plugin: 'com.android.library'
// ...
apply plugin: 'com.jakewharton.butterknife'
```

Stellen Sie jetzt sicher, dass Sie in allen ButterKnife-Anmerkungen `R2` anstelle von `R` verwenden.

```
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
  }
```

```
}
```

Bindungsansichten mit ButterKnife

Wir können Felder mit `@BindView` und einer Ansichts-ID für Butter Knife kommentieren, um die entsprechende Ansicht in unserem Layout zu suchen und automatisch `@BindView` .

Verbindliche Ansichten

Ansichten in Aktivität binden

```
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...
  }
}
```

Verbindliche Ansichten in Fragmenten

```
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();
    }
}
```

Bindungsansichten in Dialogen

Wir können `ButterKnife.findById` , um Ansichten zu einer Ansicht, Aktivität oder einem Dialog zu suchen. Es verwendet Generika, um auf den Rückgabetyp zu schließen, und führt automatisch die Umwandlung durch.

```
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);
```

Bindungsansichten in ViewHolder

```
static class ViewHolder {
    @BindView(R.id.title) TextView name;
    @BindView(R.id.job_title) TextView jobTitle;

    public ViewHolder(View view) {
      ButterKnife.bind(this, view);
    }
  }
```

Bindungsressourcen

Abgesehen davon , dass die für Ansichten zu binden, könnte man auch Buttermesser verwenden , um Ressourcen zu binden , wie diejenigen , die innerhalb `strings.xml` , `drawables.xml` , `colors.xml` , `dimens.xml` usw.

```
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);
    }

}
```

Bindungslisten

Sie können mehrere Ansichten in einer Liste oder einem Array zusammenfassen. Dies ist sehr hilfreich, wenn eine Aktion für mehrere Ansichten gleichzeitig ausgeführt werden muss.

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

Optionale Bindungen

Standardmäßig sind sowohl `@Bind` als auch Listener-Bindungen erforderlich. Eine Ausnahme wird ausgelöst, wenn die Zielansicht nicht gefunden werden kann. Wenn wir jedoch nicht sicher sind, ob eine Ansicht vorhanden ist oder nicht, können wir den Feldern eine `@Nullable` Annotation oder den `@Optional` Annotation Methoden hinzufügen, um dieses Verhalten zu unterdrücken und eine optionale Bindung zu erstellen.

```
@Nullable
@BindView(R.id.might_not_be_there) TextView mightNotBeThere;

@Optional
@OnClick(R.id.maybe_missing)
void onMaybeMissingClicked() {
  // TODO ...
}
```

Binden von Hörern mit ButterKnife

OnClick Listener:

```
@OnClick(R.id.login)
public void login(View view) {
  // Additional logic
}
```

Alle Argumente für die Listener-Methode sind optional:

```
@OnClick(R.id.login)
public void login() {
   // Additional logic
}
```

Spezieller Typ wird automatisch gegossen:

```
@OnClick(R.id.submit)
public void sayHi(Button button) {
  button.setText("Hello!");
}
```

Mehrere IDs in einer einzigen Bindung für die Behandlung allgemeiner Ereignisse:

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

Benutzerdefinierte Ansichten können an ihre eigenen Listener gebunden werden, indem keine ID angegeben wird:

```
public class CustomButton extends Button {
  @OnClick
  public void onClick() {
    // TODO
  }
}
```

Unverbaubare Ansichten in ButterKnife

Fragmente haben einen anderen Ansichtslebenszyklus als Aktivitäten. Setzen Sie beim Binden eines Fragments in onCreateView die Ansichten in onDestroyView auf null. Butter Knife gibt eine Unbinder-Instanz zurück, wenn Sie bind aufrufen, um dies für Sie auszuführen. Rufen Sie seine Unbind-Methode im entsprechenden Lifecycle-Callback auf.

Ein Beispiel:

```
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();
  }
}
```

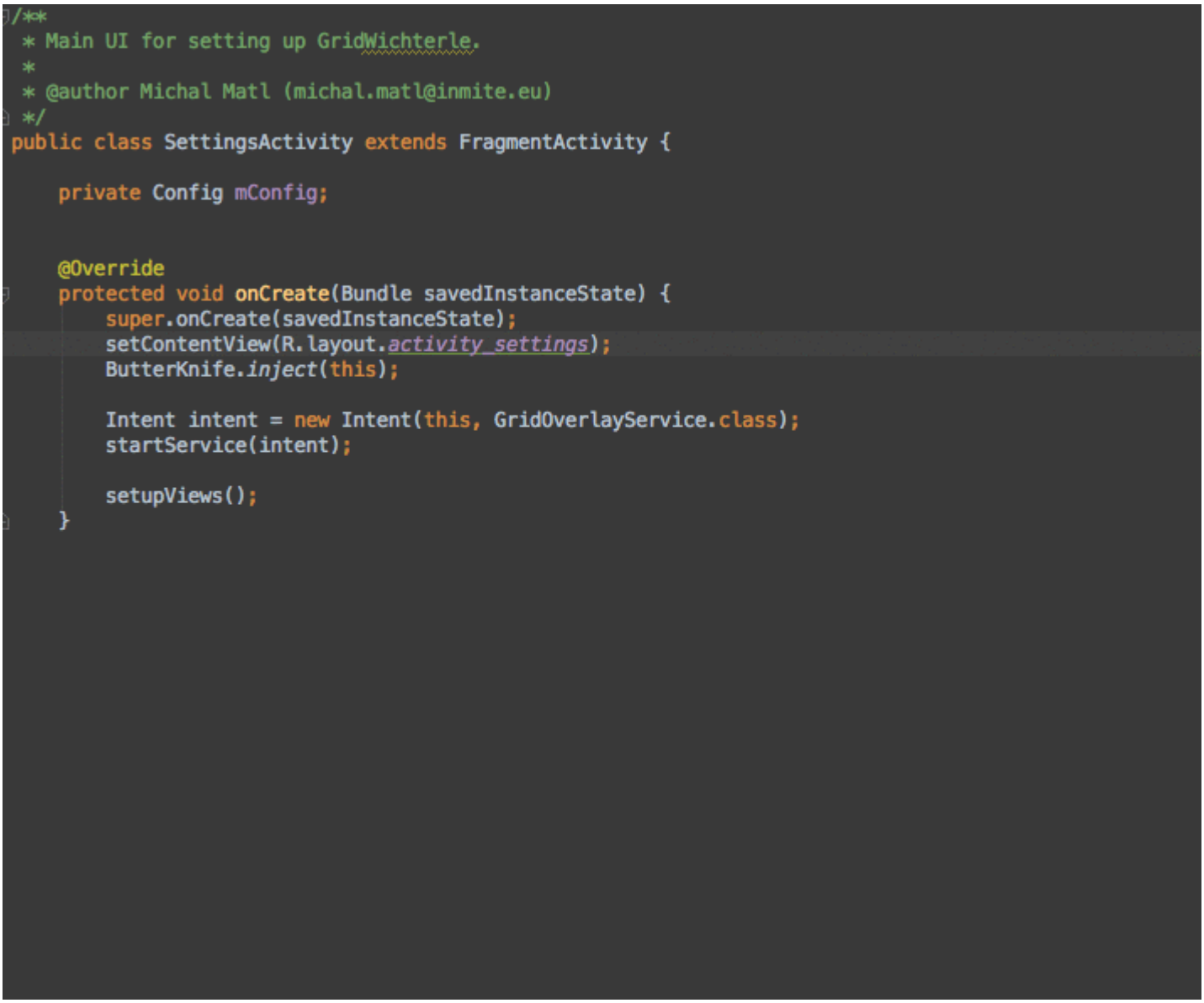
> Hinweis: Das Aufrufen von unbind () in onDestroyView () ist nicht erforderlich, wird jedoch empfohlen, da es viel Speicherplatz spart, wenn Ihre App einen großen Backstack hat.

Android Studio ButterKnife Plugin

Android ButterKnife Zelezny

> Plugin zum Erzeugen von ButterKnife-Injektionen aus ausgewählten Layout-XMLs in Aktivitäten / Fragmenten / Adaptern.

Hinweis: ***Stellen*** Sie sicher, dass Sie mit der rechten ***Maustaste*** auf ***your_xml_layou*** `(R.layout.your_xml_layou`) `(R.layout.your_xml_layou` andernfalls enthält das *Menü Generate* keine Butterknife-Injektoroption.



```java
/**
 * Main UI for setting up GridWichterle.
 *
 * @author Michal Matl (michal.matl@inmite.eu)
 */
public class SettingsActivity extends FragmentActivity {

    private Config mConfig;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);
        ButterKnife.inject(this);

        Intent intent = new Intent(this, GridOverlayService.class);
        startService(intent);

        setupViews();
    }
```

Link: Jetbrains Plugin Android ButterKnife Zelezny

Buttermesser online lesen: https://riptutorial.com/de/android/topic/1072/buttermesser

Kapitel 56: CardView

Einführung

Ein FrameLayout mit abgerundeten Ecken und Schatten.

CardView verwendet auf Lollipop die Elevation-Eigenschaft für Schatten und greift auf eine benutzerdefinierte emulierte Schattenimplementierung auf älteren Plattformen zurück.

Aufgrund des kostenaufwändigen Clipping der abgerundeten Ecken auf Plattformen vor Lollipop schneidet CardView seine untergeordneten Elemente nicht mit abgerundeten Ecken ab. Stattdessen wird eine Auffüllung hinzugefügt, um eine solche Überschneidung zu vermeiden (siehe setPreventCornerOverlap (boolean), um dieses Verhalten zu ändern).

Parameter

Parameter	Einzelheiten
cardBackgroundColor	Hintergrundfarbe für CardView.
cardCornerRadius	Eckenradius für CardView.
cardElevation	Elevation für CardView.
cardMaxElevation	Maximale Höhe für CardView.
cardPreventCornerOverlap	Fügen Sie CardView in Version 20 und zuvor eine Auffüllung hinzu, um Schnittpunkte zwischen dem Karteninhalt und abgerundeten Ecken zu verhindern.
cardUseCompatPadding	Fügen Sie in API v21 + auch das Auffüllen hinzu, um die gleichen Maße wie in früheren Versionen zu erhalten. Kann ein boolescher Wert sein, z. B. "true" oder "false".
contentPadding	Innenpolsterung zwischen den Kartenrändern und den untergeordneten Elementen des CardView.
contentPaddingBottom	Innenpolsterung zwischen der Unterkante der Karte und den Kindern des CardView.
contentPaddingLeft	Innenpolsterung zwischen dem linken Rand der Karte und den untergeordneten Elementen des CardView.
contentPaddingRight	Elevation für CardView.
cardElevation	Innenpolsterung zwischen dem rechten Kartenrand und den Kindern des CardView.

Parameter	Einzelheiten
contentPaddingTop	Innenpolsterung zwischen oberem Kartenrand und untergeordneten Elementen des CardView.

Bemerkungen

`CardView` verwendet echte Höhen und dynamische Schatten auf Lollipop (API 21) und höher. Bevor jedoch Lollipop `CardView` auf eine programmatische Schattenimplementierung `CardView` .

Wenn Sie versuchen, eine `ImageView` in die abgerundeten Ecken einer `CardView` , werden Sie feststellen, dass sie vor dem Lollipop (API 21) nicht richtig aussieht. Um dies zu beheben, sollten Sie `setPreventCornerOverlap(false)` in `CardView` oder die `app:cardPreventCornerOverlap="false"` hinzufügen `app:cardPreventCornerOverlap="false"` zu Ihrem Layout.

Vor der Verwendung von `CardView` Sie die Abhängigkeit der Unterstützungsbibliothek in der Datei `build.gradle` :

```
dependencies{
    compile 'com.android.support:cardview-v7:25.2.0'
}
```

Eine Nummer der neuesten Version finden Sie hier

Offizielle Dokumentation:

https://developer.android.com/reference/android/support/v7/widget/CardView.html
https://developer.android.com/training/material/lists-cards.html

Examples

Erste Schritte mit CardView

`CardView` ist Mitglied der Android Support Library und bietet ein Layout für Karten.

`CardView` Sie zum Hinzufügen von `CardView` zu Ihrem Projekt die folgende Zeile zu Ihren `build.gradle` Abhängigkeiten hinzu.

```
compile 'com.android.support:cardview-v7:25.1.1'
```

Eine Nummer der neuesten Version finden Sie hier

In Ihrem Layout können Sie dann Folgendes hinzufügen, um eine Karte zu erhalten.

```
<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>
```

Sie können dann weitere Layouts hinzufügen, die in einer Karte enthalten sind.

CardView kann außerdem mit einem beliebigen UI-Element gefüllt und vom Code aus manipuliert werden .

```
<?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 anpassen

CardView bietet eine Standardhöhe und einen Eckenradius, damit die Karten auf allen Plattformen einheitlich aussehen.

Sie können diese Standardwerte mithilfe der folgenden Attribute in der XML-Datei anpassen:

1. `card_view:cardElevation` Attribut fügt eine Erhöhung in CardView hinzu.
2. `card_view:cardBackgroundColor` Attribut wird verwendet, um die Hintergrundfarbe des CardView-Hintergrunds anzupassen (Sie können jede Farbe angeben).
3. `card_view:cardCornerRadius` Attribut wird verwendet, um 4 Kanten von CardView zu krümmen
4. `card_view:contentPadding` Attribut fügt `card_view:contentPadding` zwischen Karte und Kinder der Karte ein

Hinweis: card_view ist ein Namespace, der in der obersten übergeordneten Layoutansicht definiert ist. xmlns: card_view = " http://schemas.android.com/apk/res-auto "

Hier ein Beispiel:

```
<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>
```

Sie können das auch programmgesteuert machen mit:

```
card.setCardBackgroundColor(....);
card.setCardElevation(...);
card.setRadius(....);
card.setContentPadding();
```

Überprüfen Sie die offiziellen Javadoc für weitere Eigenschaften.

Ripple-Animation hinzufügen

Fügen Sie die folgenden Attribute hinzu, um die Ripple-Animation in einer CardView zu aktivieren:

```
<android.support.v7.widget.CardView
  ...
  android:clickable="true"
  android:foreground="?android:attr/selectableItemBackground">
  ...
</android.support.v7.widget.CardView>
```

Verwenden von Bildern als Hintergrund in CardView (Probleme mit dem Lollipop-Gerät)

Wenn Sie Image / Color als Hintergrund in einem CardView verwenden, können am Ende leichte weiße Polsterungen (Wenn die Standardfarbe der Karte weiß ist) an den Rändern erhalten. Dies ist auf die standardmäßig abgerundeten Ecken in der Kartenansicht zurückzuführen. So vermeiden Sie diese Ränder in Pre-Lollipop-Geräten.

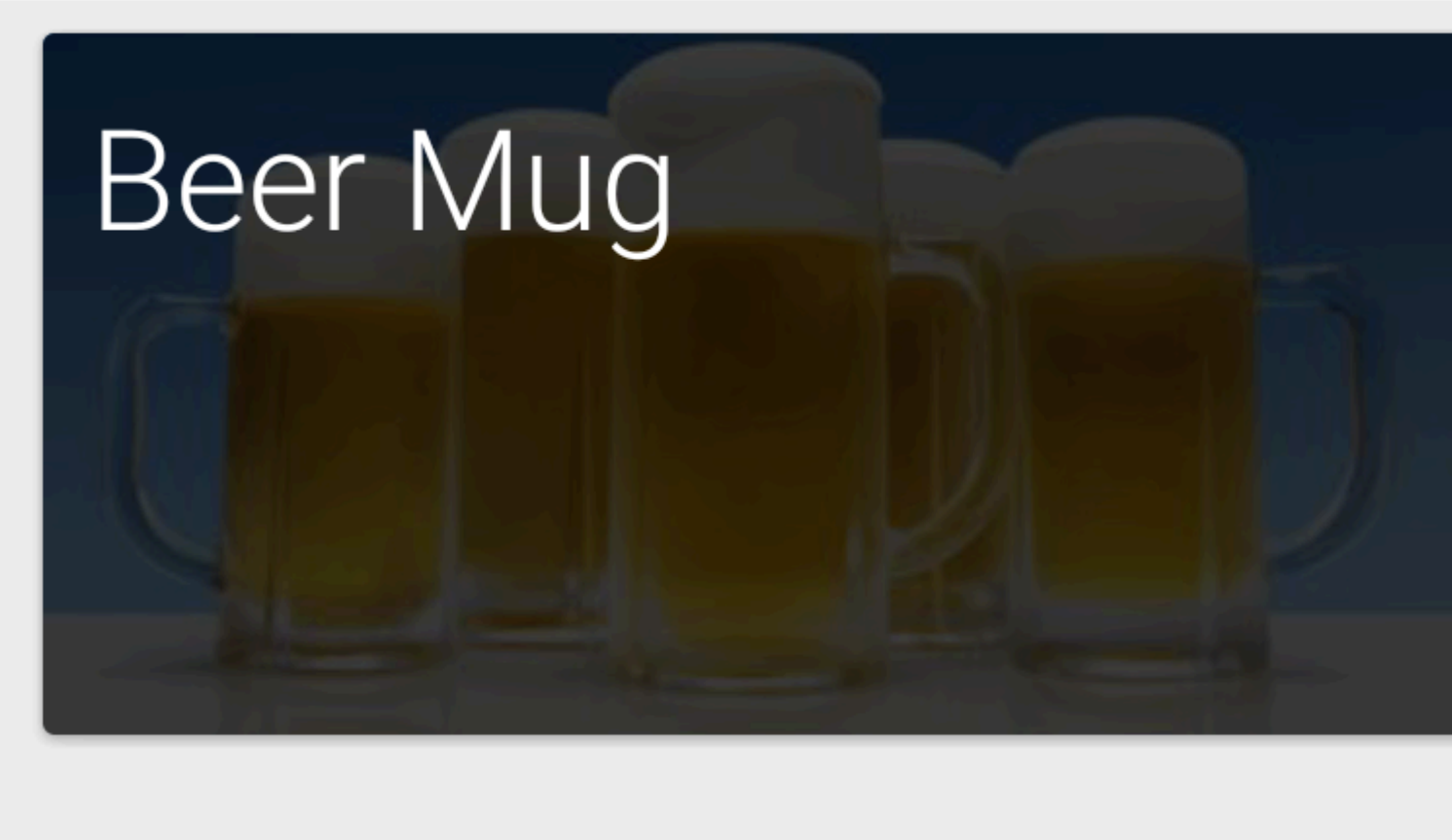
Wir müssen ein Attribut `card_view:cardPreventCornerOverlap="false"` in der CardView. 1). Verwenden Sie in XML das folgende Snippet.

```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    card_view:cardPreventCornerOverlap="false"
    android:layout_height="wrap_content">
      <ImageView
            android:id="@+id/row_wallet_redeem_img"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:adjustViewBounds="true"
            android:scaleType="centerCrop"
            android:src="@drawable/bg_image" />

</android.support.v7.widget.CardView>
```

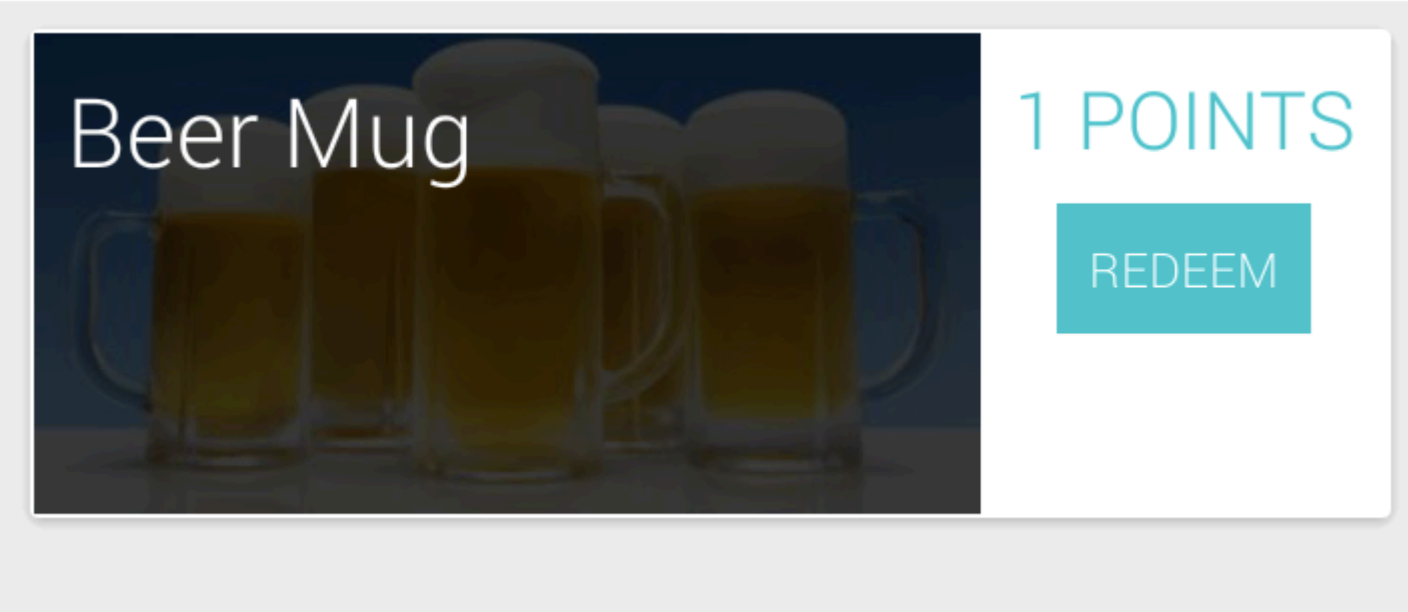
2. In Java wie diesem `cardView.setPreventCornerOverlap(false)` .

Dadurch wird eine unerwünschte Auffüllung der Kartenränder entfernt. Hier sind einige visuelle Beispiele für diese Implementierung.

1 Karte mit Bildhintergrund in API 21 (vollkommen in Ordnung)

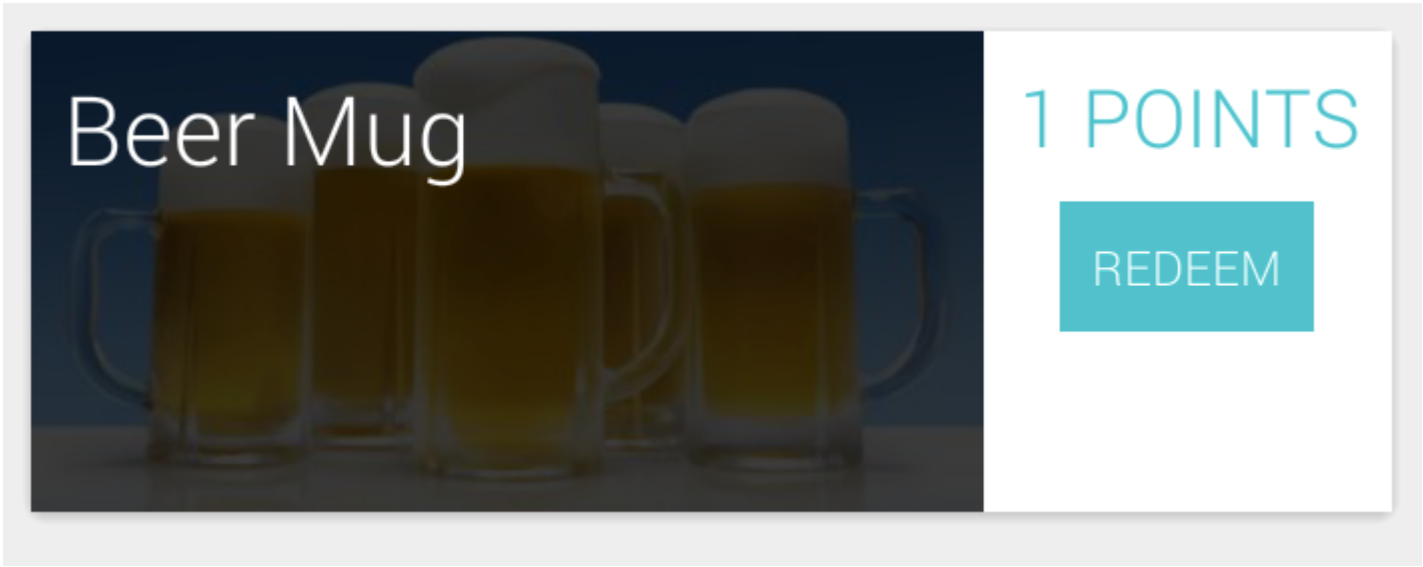


2 Karte mit Bildhintergrund in API 19 ohne Attribut (beachten Sie die Auffüllungen um das Bild)



3 FIXED-Karte mit Bildhintergrund in API 19 mit Attribut

`cardView.setPreventCornerOverlap(false)` (`cardView.setPreventCornerOverlap(false)` jetzt behoben)



Lesen Sie dazu auch die Dokumentation hier
Original SOF Beitrag hier

Animieren Sie die CardView-Hintergrundfarbe mit TransitionDrawable

```
public void setCardColorTran(CardView card) {
    ColorDrawable[] color = {new ColorDrawable(Color.BLUE), new ColorDrawable(Color.RED)};
    TransitionDrawable trans = new TransitionDrawable(color);
    if(Build.VERSION.SDK_INT > Build.VERSION_CODES.ICE_CREAM_SANDWICH_MR1) {
        card.setBackground(trans);
    } else {
        card.setBackgroundDrawable(trans);
    }
    trans.startTransition(5000);
}
```

CardView online lesen: https://riptutorial.com/de/android/topic/726/cardview

Kapitel 57: Chipkarte

Examples

Smartcard senden und empfangen

Zum Verbinden finden Sie hier einen Ausschnitt, der Ihnen hilft zu verstehen:

```
//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);
}
```

Nun müssen Sie verstehen, dass in Java die Kommunikation mit dem Paket javax.smarcard stattfindet, das für Android nicht verfügbar ist. Schauen Sie sich hier an, um eine Idee zu bekommen, wie Sie APDU (Smartcard-Befehl) kommunizieren oder senden / empfangen können.

Nun, wie in der Antwort erwähnt

Sie können nicht einfach eine APDU (Smartcard-Befehl) über den Bulk-Out-Endpunkt senden und eine Antwort-APDU über den Bulk-In-Endpunkt erhalten. Um die Endpunkte zu erhalten, lesen Sie den folgenden Code-Ausschnitt:

```
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;
                }
            }
```

```
        }
    }
```

Jetzt haben Sie die Bulk-In- und Bulk-Out-Endpunkte zum Senden und Empfangen von APDU-Befehlsblöcken und APDU-Antwortblöcken:

Informationen zum Senden von Befehlen finden Sie im folgenden Codeausschnitt:

```
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));
    }
}
```

Informationen zum Empfangen / Lesen einer Antwort finden Sie im folgenden Code-Snippet:

```
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;
}
```

Wenn Sie diese Antwort hier sehen, lautet der erste zu sendende Befehl:

Befehl PC_to_RDR_IccPowerOn zum Aktivieren der Karte.

die Sie durch Lesen des Abschnitts 6.1.1 des Dokuments USB-Gerätespezifikationen hier erstellen können.

Nehmen wir nun ein Beispiel für diesen Befehl wie den folgenden: `62000000000000000000` So können Sie `62000000000000000000` senden:

```
write(connection, epOut, "62000000000000000000");
```

Nachdem Sie den APDU-Befehl erfolgreich gesendet haben, können Sie die Antwort mit folgendem Befehl lesen:

```
read(connection, epIn);
```

Und so etwas bekommen

```
80 18000000 00 00 00 00 00 3BBF11008131FE45455041000000000000000000000000F1
```

Die hier im Code empfangene Antwort befindet sich nun in der `result` der `read()` -Methode aus dem Code

Chipkarte online lesen: https://riptutorial.com/de/android/topic/10945/chipkarte

Kapitel 58: CleverTap

Einführung

Schnelle Hacks für das Analyse- und Engagement-SDK von CleverTap - Android

Bemerkungen

Holen Sie sich Ihre CleverTap-Anmeldeinformationen von https://clevertap.com .

Examples

Rufen Sie eine Instanz des SDK ab, um Ereignisse aufzuzeichnen

```
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
}
```

Debug-Level einstellen

Überschreiben `onCreate()` in Ihrer benutzerdefinierten Anwendungsklasse die `onCreate()` Methode und fügen Sie die folgende Zeile hinzu:

```
CleverTapAPI.setDebugLevel(1);
```

CleverTap online lesen: https://riptutorial.com/de/android/topic/9337/clevertap

Kapitel 59: ConstraintLayout

Einführung

`ConstraintLayout` ist eine `ViewGroup` , mit der Sie Widgets flexibel positionieren und `ViewGroup` können. Es ist kompatibel mit Android 2.3 (API Level 9) und höher.

Sie können damit große und komplexe Layouts mit einer flachen Ansichtshierarchie erstellen. `RelativeLayout` ähnelt `RelativeLayout` , als alle Ansichten nach Beziehungen zwischen den Ansichten von Geschwistern und dem übergeordneten Layout angeordnet sind, es ist jedoch flexibler als `RelativeLayout` und mit dem Layout Editor von Android Studio einfacher zu verwenden.

Syntax

- **ConstraintLayout**
    - public void addView (View child, int index, ViewGroup.LayoutParams-Parameter)
    - public ConstraintLayout.LayoutParams generateLayoutParams (AttributeSet-Attribute)
    - public void onViewAdded (Ansichtsansicht)
    - public void onViewRemoved (Ansichtsansicht)
    - public void removeView (Ansichtsansicht)
    - public void requestLayout ()
    - protected boolean checkLayoutParams (ViewGroup.LayoutParams-Parameter)
    - protected ConstraintLayout.LayoutParams generateDefaultLayoutParams ()
    - protected ViewGroup.LayoutParams generateLayoutParams (ViewGroup.LayoutParams-Parameter)
    - protected void onLayout (boolean geändert, int links, int oben, int rechts, int unten)
    - geschützt void onMeasure (int widthMeasureSpec, int heightMeasureSpec)
- **ConstraintLayout.LayoutParams**
    - public void resolLayoutDirection (int layoutDirection)
    - public void validate ()
    - geschützt void setBaseAttributes (TypedArray a, int widthAttr, int heightAttr)

Parameter

Parameter	Einzelheiten
Kind	Die `View` , die dem Layout hinzugefügt werden soll
Index	Der Index der `View` in der Layouthierarchie
Params	Die `LayoutParams` der `View`
Attrs	Das `AttributeSet` , das die `LayoutParams` definiert
Aussicht	Die hinzugefügte oder entfernte `View`
geändert	Gibt an, ob diese `View` Größe oder Position geändert hat
links	Die linke Position relativ zur übergeordneten `View`
oben	Die obere Position relativ zur übergeordneten `View`
Recht	Die rechte Position relativ zur übergeordneten `View`
Unterseite	Die untere Position relativ zur übergeordneten `View`
widthMeasureSpec	Der horizontale Platzbedarf der übergeordneten `View`
heightMeasureSpec	Der vertikale Platzbedarf der übergeordneten `View`
layoutDirection	-
ein	-
widthAttr	-
heightAttr	-

Bemerkungen

Auf der Google IO 2016 gab Google ein neues Android-Layout mit dem Namen ConstraintLayout bekannt.
Achten Sie darauf, da dieses Layout derzeit eine **Beta-Version ist** .

Weitere Informationen zum Constraint-Layout:

https://codelabs.developers.google.com/codelabs/constraint-layout/index.html

Examples

ConstraintLayout zu Ihrem Projekt hinzufügen

Um mit ConstraintLayout arbeiten zu können, benötigen Sie Android Studio Version 2.2 oder neuer und mindestens die Version 32 (oder höher) des Android Support Repository.

1. Fügen Sie die Constraint Layout-Bibliothek als Abhängigkeit in Ihrer `build.gradle` Datei hinzu:

```
dependencies {
   compile 'com.android.support.constraint:constraint-layout:1.0.2'
}
```

2. Projekt synchronisieren

So fügen Sie Ihrem Projekt ein neues Constraint-Layout hinzu:

1. **Klicken** Sie mit der **rechten Maustaste** auf das Layoutverzeichnis Ihres Moduls, und klicken Sie dann auf `New > XML > Layout XML.`
2. Geben Sie einen **Namen** für das Layout ein und geben Sie `"android.support.constraint.ConstraintLayout"` als Root-Tag ein.
3. Klicken Sie auf **Fertig stellen** .

Ansonsten fügen Sie einfach eine Layoutdatei hinzu:

```
<?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>
```

Ketten

Seit `ConstraintLayout` alpha 9 stehen **Ketten** zur Verfügung. Eine **Kette** ist eine Gruppe von Ansichten innerhalb eines `ConstraintLayout` , die bidirektional miteinander verbunden sind, dh **A** mit **B** mit einer Einschränkung und **B** mit **A** mit einer anderen Einschränkung.

Beispiel:

```
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- this view is linked to the bottomTextView -->
    <TextView
        android:id="@+id/topTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        app:layout_constraintBottom_toTopOf="@+id/bottomTextView"
        app:layout_constraintTop_toTopOf="parent"
```

```
        app:layout_constraintVertical_chainPacked="true"/>

    <!-- this view is linked to the topTextView at the same time -->
    <TextView
        android:id="@+id/bottomTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom\nMkay"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/topTextView"/>

</android.support.constraint.ConstraintLayout>
```

In diesem Beispiel befinden sich die beiden Ansichten untereinander und beide sind vertikal zentriert. Sie können die vertikale Position dieser Ansichten ändern, indem Sie die **Vorspannung** der Kette anpassen. Fügen Sie dem ersten Element einer Kette den folgenden Code hinzu:

```
app:layout_constraintVertical_bias="0.2"
```

In einer vertikalen Kette ist das erste Element eine Ansicht ganz oben und in einer horizontalen Kette ist es die Ansicht ganz links. Das erste Element definiert das Verhalten der gesamten Kette.

Ketten sind eine neue Funktion und werden regelmäßig aktualisiert. Hier ist eine offizielle Android-Dokumentation zu Ketten.

ConstraintLayout online lesen: https://riptutorial.com/de/android/topic/5076/constraintlayout

Kapitel 60: ConstraintSet

Einführung

Mit dieser Klasse können Sie programmatisch eine Gruppe von Einschränkungen definieren, die mit `ConstraintLayout` . Damit können Sie Einschränkungen erstellen und speichern und auf ein vorhandenes `ConstraintLayout` anwenden.

Examples

ConstraintSet mit ContraintLayout programmgesteuert

```
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
        }
    }
}
```

ConstraintSet online lesen: https://riptutorial.com/de/android/topic/9334/constraintset

Kapitel 61: CoordinatorLayout und Verhalten

Einführung

Das CoordinatorLayout ist ein FrameLayout mit Super-Power. Ziel dieser ViewGroup ist es, die darin enthaltenen Ansichten zu koordinieren.

Der Hauptanreiz des CoordinatorLayout ist seine Fähigkeit, die Animationen und Übergänge der Ansichten in der XML-Datei selbst zu koordinieren.

CoordinatorLayout ist für zwei primäre Anwendungsfälle vorgesehen:

: Als Anwendungsdekor auf oberster Ebene oder als Chrom-Layout

: Als Container für eine bestimmte Interaktion mit einer oder mehreren untergeordneten Ansichten

Bemerkungen

Das `CoordinatorLayout` ist ein Container, der das `FrameLayout` .
Durch Anhängen eines `CoordinatorLayout.Behavior` an ein direktes untergeordnetes Element von `CoordinatorLayout` können Sie Berührungsereignisse, Fensterinsets, Messungen, Layout und verschachteltes Scrollen abfangen.

Durch Angeben von `Behaviors` für untergeordnete Ansichten eines `CoordinatorLayout` Sie viele verschiedene Interaktionen innerhalb eines einzelnen übergeordneten `CoordinatorLayout` bereitstellen. Diese Ansichten können auch miteinander interagieren. View-Klassen können ein Standardverhalten angeben, wenn sie als `DefaultBehavior` eines `CoordinatorLayout` mithilfe der Annotation `DefaultBehavior` .

Examples

Ein einfaches Verhalten erstellen

Um ein `Behavior` zu erstellen, erweitern Sie einfach die `CoordinatorLayout.Behavior` Klasse.

Erweitern Sie das CoordinatorLayout.Behavior

Beispiel:

```
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);
    }
}
```

Dieses Verhalten muss an eine untergeordnete Ansicht eines `CoordinatorLayout` angehängt werden, um aufgerufen zu werden.

Binden Sie ein Verhalten programmgesteuert an

```
MyBehavior myBehavior = new MyBehavior();
CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams)
view.getLayoutParams();
params.setBehavior(myBehavior);
```

Fügen Sie ein Verhalten in XML an

Sie können das Attribut `layout_behavior` , um das Verhalten in XML anzuhängen:

```
<View
  android:layout_height="...."
  android:layout_width="...."
  app:layout_behavior=".MyBehavior" />
```

Fügen Sie ein Verhalten automatisch an

Wenn Sie mit einer benutzerdefinierten Ansicht arbeiten, können Sie das Verhalten mithilfe der Annotation `@CoordinatorLayout.DefaultBehavior` anfügen:

```
@CoordinatorLayout.DefaultBehavior(MyBehavior.class)
public class MyView extends ..... {

}
```

Verwenden des SwipeDismissBehavior

Das `SwipeDismissBehavior` funktioniert für jede Ansicht und implementiert die Funktion zum `SwipeDismissBehavior` in unseren Layouts mit einem `CoordinatorLayout` .

Benutz einfach:

```
        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);
```

Erstellen Sie Abhängigkeiten zwischen Ansichten

Sie können `CoordinatorLayout.Behavior` , um Abhängigkeiten zwischen Ansichten zu erstellen. Sie können eine `View` mit einer anderen `View` verankern, indem Sie:

- mit dem Attribut `layout_anchor` .
- Erstellen eines benutzerdefinierten `Behavior` und Implementieren der Methode `layoutDependsOn` , die `true` `layoutDependsOn` .

Um beispielsweise ein `Behavior` zum Verschieben einer `ImageView` zu erstellen, wenn eine andere verschoben wird (Beispielsymbolleiste), führen Sie die folgenden Schritte aus:

- Erstellen Sie das benutzerdefinierte Verhalten :

  ```
  public class MyBehavior extends CoordinatorLayout.Behavior<ImageView> {...}
  ```

- `layoutDependsOn` Methode `layoutDependsOn` den `layoutDependsOn` `true` `layoutDependsOn` . Diese Methode wird bei jeder Änderung des Layouts aufgerufen:

  ```
  @Override
  public boolean layoutDependsOn(CoordinatorLayout parent,
          ImageView child, View dependency) {
      // Returns true to add a dependency.
      return dependency instanceof Toolbar;
  }
  ```

- Immer wenn die Methode `layoutDependsOn` den `onDependentViewChanged` `layoutDependsOn` zurückgibt `true` die Methode `onDependentViewChanged` aufgerufen:

```
@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);
}
```

CoordinatorLayout und Verhalten online lesen:
https://riptutorial.com/de/android/topic/5714/coordinatorlayout-und-verhalten

Kapitel 62: Countdown-Timer

Parameter

Parameter	Einzelheiten
`long millisInFuture`	Die Gesamtdauer, für die der Timer ausgeführt wird, dh wie weit in der Zukunft der Timer beendet werden soll. In Millisekunden.
`long countDownInterval`	Das Intervall, in dem Sie Timer-Updates erhalten möchten. In Millisekunden.
`long millisUntilFinished`	Ein in `onTick()` bereitgestellter Parameter, der `onTick()` , wie lange CountDownTimer noch vorhanden ist. In Millisekunden

Bemerkungen

CountDownTimer ist eine ziemlich schlanke Klasse - sie tut eines sehr gut. Da Sie nur einen CountDownTimer starten oder abbrechen können, müssen Sie die Funktion zum Anhalten / Wiederaufnehmen wie im zweiten Beispiel gezeigt implementieren. Verwenden Sie für komplexere Funktionen oder zur Angabe eines Timers, der unbegrenzt ausgeführt werden soll, das Timer-Objekt.

Examples

Einen einfachen Countdown-Timer erstellen

CountDownTimer ist nützlich, wenn eine Aktion für eine festgelegte Dauer wiederholt in einem konstanten Intervall ausgeführt wird. In diesem Beispiel aktualisieren wir jede Sekunde 30 Sekunden lang eine Textansicht, in der die verbleibende Zeit angezeigt wird. Wenn der Timer beendet ist, setzen wir die Textansicht auf "Fertig".

```
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();
```

Ein komplexeres Beispiel

In diesem Beispiel wird der CountDownTimer außerhalb des Aktivitätslebenszyklus angehalten / fortgesetzt.

```
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;
}
```

Countdown-Timer online lesen: https://riptutorial.com/de/android/topic/6063/countdown-timer

Kapitel 63: Crash-Reporting-Tools

Bemerkungen

Das beste komplette Wiki gibt es hier in github .

Examples

Stoff - Crashlytics

Fabric ist eine modulare mobile Plattform, die nützliche Kits enthält, die Sie zum Erstellen Ihrer Anwendung kombinieren können. **Crashlytics** ist ein von Fabric zur Verfügung gestelltes Tool zum **Absturz-** und **Fehlerbericht** , mit dem Sie Ihre Anwendungen detailliert verfolgen und überwachen können.

So konfigurieren Sie Fabric-Crashlytics

Schritt 1: Ändern Sie Ihr `build.gradle` :

Fügen Sie das Plugin-Repo und das Gradle-Plugin hinzu:

```
 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.+'
  }
}
```

Wenden Sie das Plugin an:

```
apply plugin: 'com.android.application'
//Put Fabric plugin after Android plugin
apply plugin: 'io.fabric'
```

Fügen Sie das Stoff-Repo hinzu:

```
repositories {
  maven { url 'https://maven.fabric.io/public' }
}
```

Füge das Crashlyrics Kit hinzu:

```
dependencies {

  compile('com.crashlytics.sdk.android:crashlytics:2.6.6@aar') {
    transitive = true;
  }
}
```

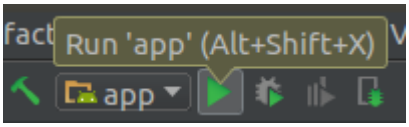
Schritt 2: Fügen Sie Ihren **API-Schlüssel** und die **INTERNET-** Berechtigung in `AndroidManifest.xml`

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

      <meta-data
          android:name="io.fabric.ApiKey"
          android:value="25eeca3bb31cd41577e097cabd1ab9eee9da151d"
          />

  </application>

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

Schritt 3: Initialisieren Sie das Kit zur Laufzeit in Ihrem Code. Beispiel:

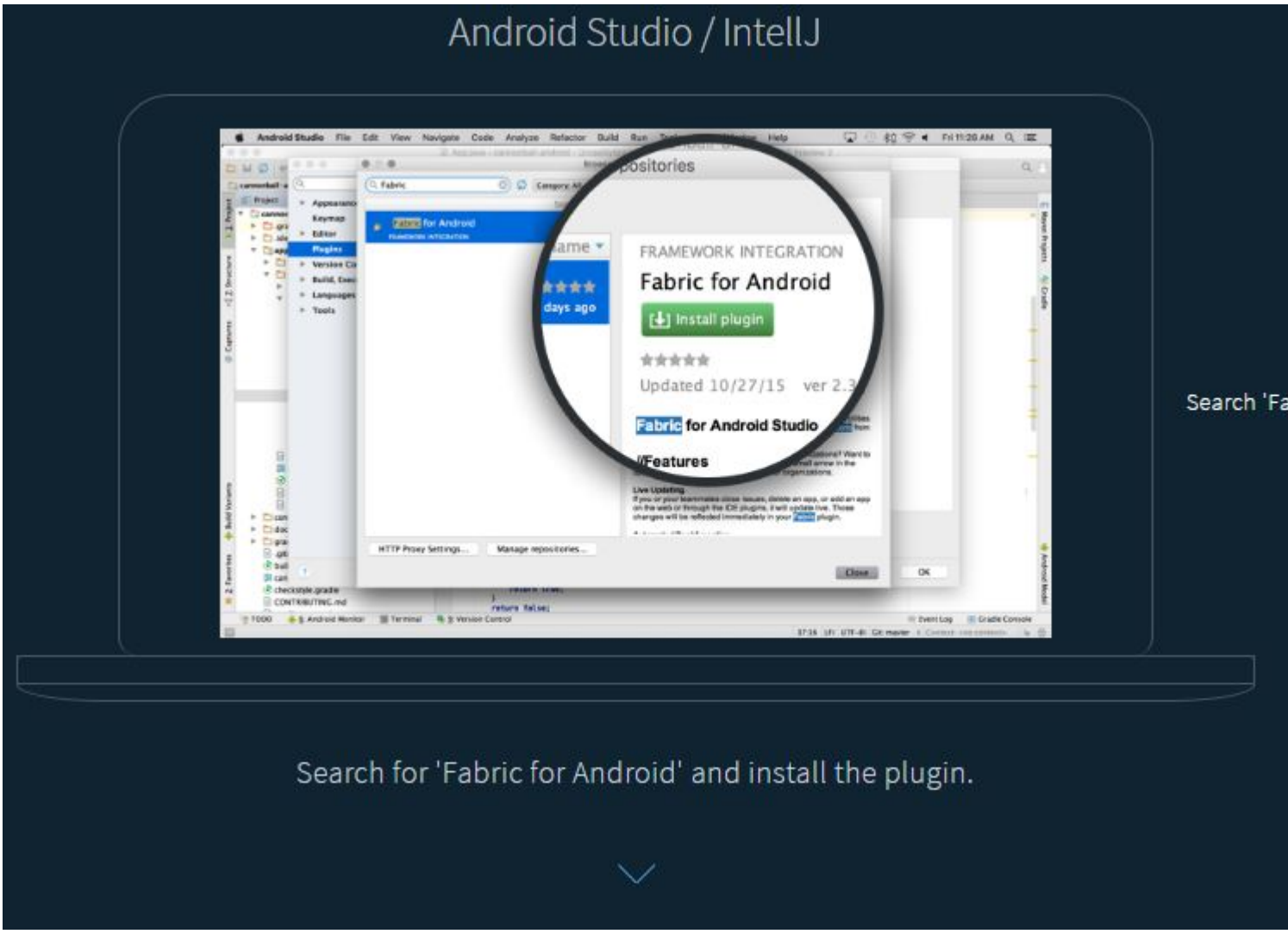
```
public class MainActivity extends ActionBarActivity {

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

      //Init the KIT
      Fabric.with(this, new Crashlytics());

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

Schritt 4: Projekt erstellen. Um zu bauen und auszuführen:



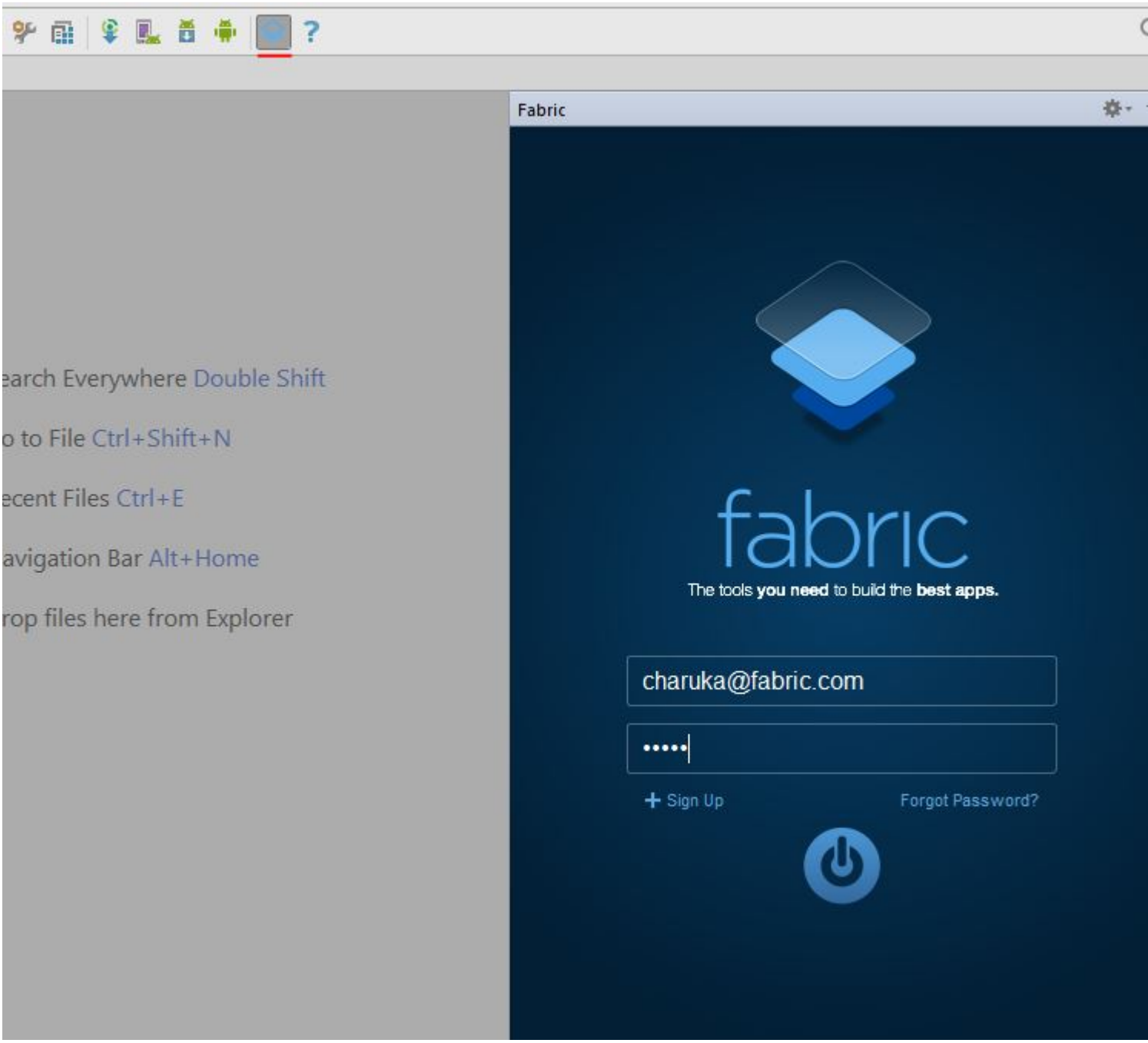
Verwenden des Fabric IDE-Plugins

Kits können mithilfe des Fabric IDE-Plugins für Android Studio oder von IntelliJ unter diesem Link installiert werden.



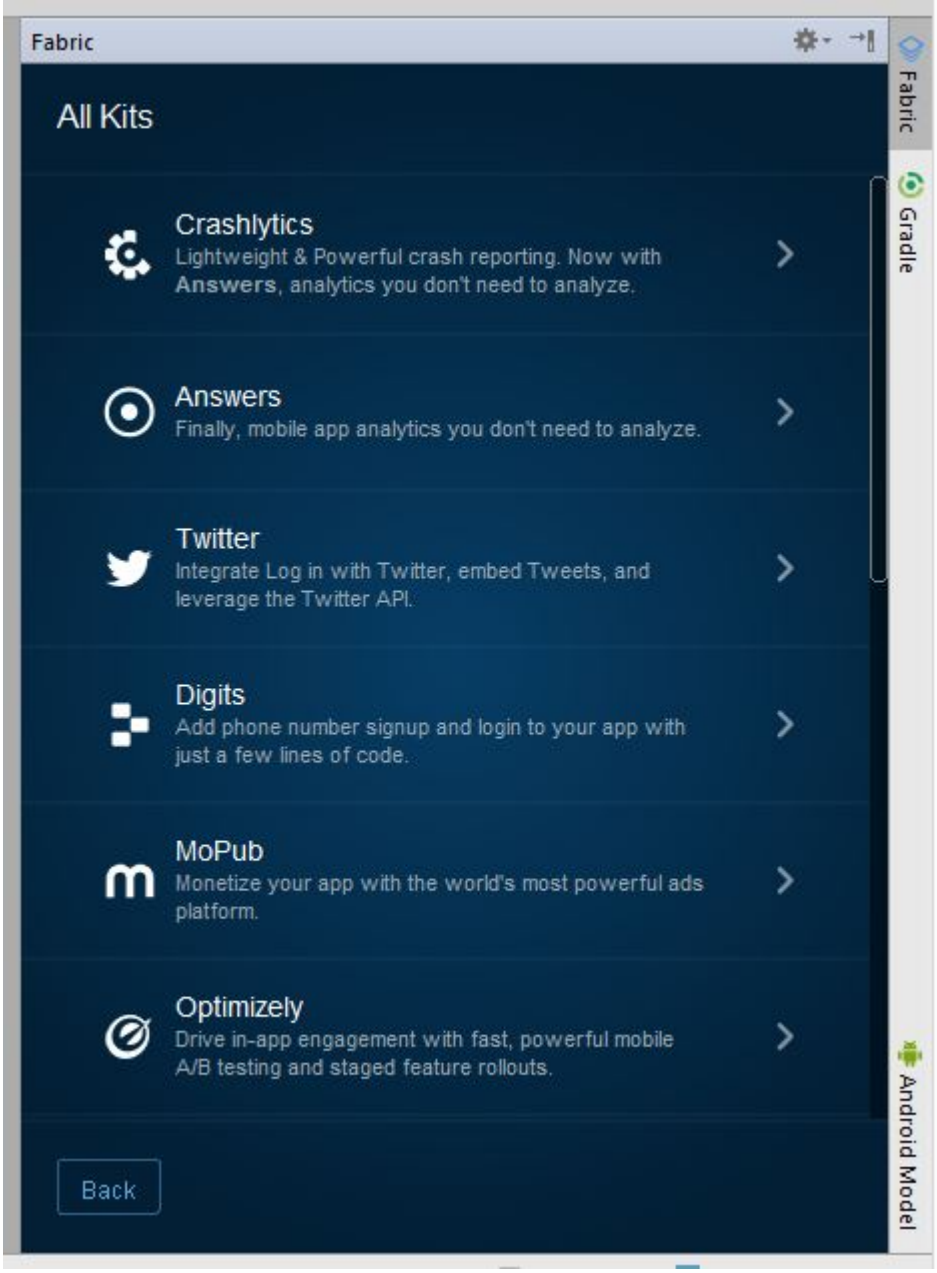
Nach der Installation des Plugin, Android Studio **neu starten** und **melden** Sie sich mit Ihrem Konto mit **Android Studio.**

(Kurztaste> `CTRL + L`)

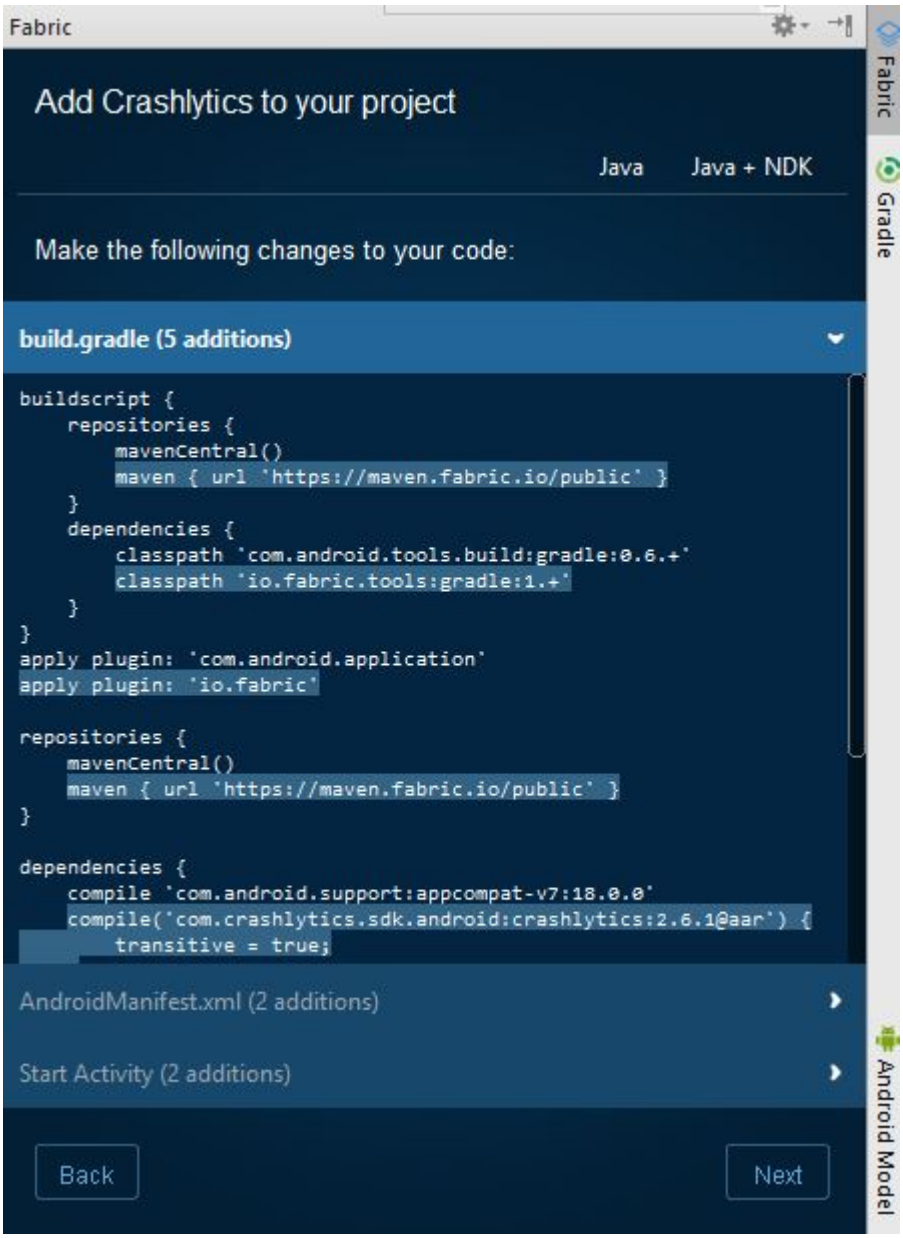


Dann werden die Projekte angezeigt, die Sie haben / das Projekt, das Sie geöffnet haben. Wählen Sie das gewünschte Projekt aus und klicken Sie auf Weiter.

Wählen Sie das Kit aus, das Sie hinzufügen möchten. Für sein Beispiel ist es **Crashlytics** :



`Install` dann auf `Install` . Sie müssen es dieses Mal nicht manuell hinzufügen, wie oben **beschriebenes Plugin** , sondern es wird für Sie erstellt.



Erledigt!

Crash-Berichterstellung mit ACRA

Schritt 1: Fügen Sie die Abhängigkeit des neuesten ACRA- AAR zu Ihrem Bewerbungsgrad (build.gradle) hinzu.

Schritt 2: Fügen Sie in Ihrer Anwendungsklasse (die Klasse, die Application erweitert; falls nicht, sie erstellen) eine Annotation `@ReportsCrashes` und überschreiben Sie die `attachBaseContext()` Methode.

Schritt 3: Initialisieren Sie die ACRA-Klasse in Ihrer Anwendungsklasse

```
@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);
            }
    }
```

Dabei sind AUTH_USERNAME und AUTH_PASSWORD die Berechtigungsnachweise Ihres gewünschten Backends .

Schritt 4: Definieren Sie die Anwendungsklasse in AndroidManifest.xml

```
<application
       android:name=".MyApplication">
       <service></service>
       <activity></activity>
       <receiver></receiver>
</application>
```

Schritt 5: Stellen Sie sicher, dass Sie über eine `internet` verfügen, um den Bericht von einer abgestürzten Anwendung zu erhalten

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

Wenn Sie den stillen Bericht an das Backend senden möchten, verwenden Sie einfach die unten stehende Methode, um dies zu erreichen.

```
ACRA.getErrorReporter().handleSilentException(e);
```

Erzwingen Sie einen Testabsturz mit Stoff

Fügen Sie eine Schaltfläche hinzu, die Sie antippen können, um einen Absturz auszulösen. Fügen Sie diesen Code in Ihr Layout ein, wo die Schaltfläche angezeigt werden soll.

```
<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 auslösen

```
public void forceCrash(View view) {
    throw new RuntimeException("This is a crash");
}
```

Starten Sie Ihre App und tippen Sie auf die neue Schaltfläche, um einen Absturz zu verursachen. In ein oder zwei Minuten sollten Sie in der Lage sein, den Absturz in Ihrem Crashlytics-Dashboard zu sehen sowie eine E-Mail zu erhalten.

Erfassen Sie Abstürze mit Sherlock

Sherlock erfasst alle Ihre Abstürze und meldet sie als Benachrichtigung. Wenn Sie auf die Benachrichtigung tippen, wird eine Aktivität mit allen Absturzdetails sowie Geräte- und Anwendungsinformationen geöffnet

Wie kann ich Sherlock in Ihre Anwendung integrieren?

Sie müssen lediglich Sherlock als gradle Abhängigkeit in Ihrem Projekt hinzufügen.

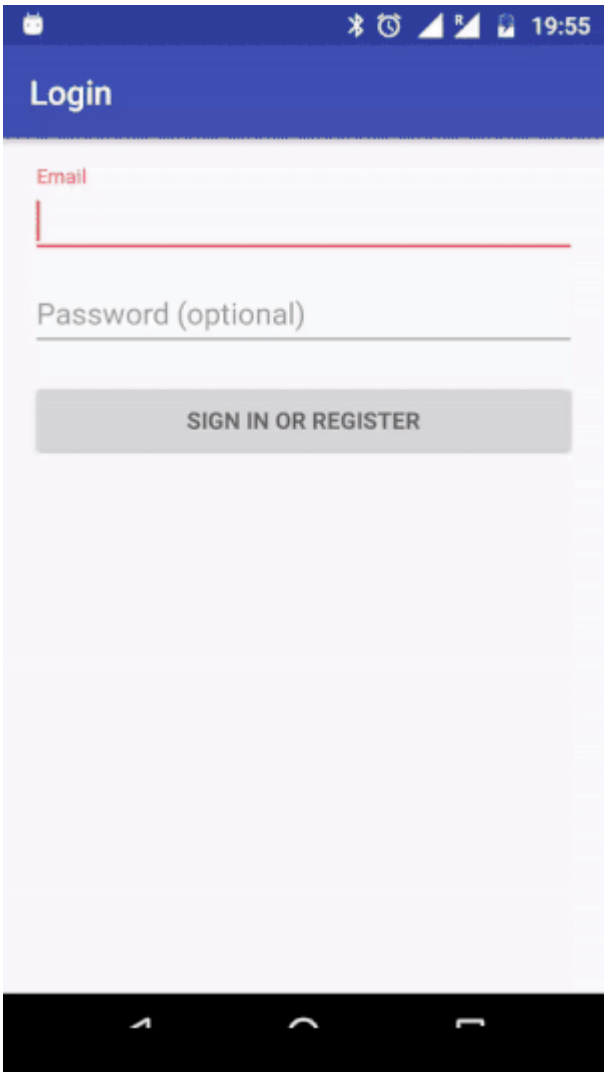
```
dependencies {
    compile('com.github.ajitsing:sherlock:1.0.1@aar') {
        transitive = true
    }
}
```

Nachdem Sie Ihr Android-Studio synchronisiert haben, initialisieren Sie Sherlock in Ihrer Application-Klasse.

```
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);
  }
}
```

Das ist alles was Sie tun müssen. Auch Sherlock kann nicht nur einen Absturz melden. Schauen Sie sich diesen Artikel an, um alle Funktionen zu überprüfen .

Demo



Crash-Reporting-Tools online lesen: https://riptutorial.com/de/android/topic/3871/crash-reporting-tools

Kapitel 64: Datei in Android entpacken

Examples

Datei entpacken

```
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;}
```

Datei in Android entpacken online lesen: https://riptutorial.com/de/android/topic/3927/datei-in-android-entpacken

Kapitel 65: Datenbindungsbibliothek

Bemerkungen

Konfiguration

Bevor Sie die Datenbindung verwenden, müssen Sie das Plugin in Ihrem `build.gradle` .

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

Hinweis: Die Datenbindung wurde dem Android Gradle-Plugin in Version 1.5.0 hinzugefügt

Verbindliche Klassennamen

Das Datenbindungs-Plugin generiert einen Bindungsklassennamen, indem der Dateiname Ihres Layouts in Pascal-Fall konvertiert und am Ende "Binding" hinzugefügt wird. Somit generiert `item_detail_activity.xml` eine Klasse mit dem Namen `ItemDetailActivityBinding` .

Ressourcen

- Offizielle Dokumentation

Examples

Grundlegende Textfeldbindung

Gradle (Modul: App) Konfiguration

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

Datenmodell

```
public class Item {
    public String name;
    public String description;

    public Item(String name, String description) {
        this.name = name;
        this.description = description;
```

```
    }
}
```

Layout XML

Der erste Schritt besteht darin, Ihr Layout in ein `<layout>` -Tag einzuhüllen, ein `<data>` -Element hinzuzufügen und ein `<variable>` -Element für Ihr Datenmodell hinzuzufügen.

Anschließend können Sie XML-Attribute mit `@{model.fieldname}` an Felder im Datenmodell `@{model.fieldname}` , wobei `model` der Name der Variablen und `fieldname` das Feld ist, auf das Sie zugreifen möchten.

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>
```

Für jede XML-Layoutdatei, die ordnungsgemäß mit Bindungen konfiguriert ist, generiert das Android Gradle-Plugin eine entsprechende Klasse: Bindungen. Da wir ein Layout mit dem Namen *item_detail_activity haben* , heißt die entsprechende generierte Bindungsklasse `ItemDetailActivityBinding` .

Diese Bindung kann dann in einer Aktivität verwendet werden:

```
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);
    }
}
```

Binden mit einer Accessor-Methode

Wenn Ihr Modell über private Methoden verfügt, können Sie in der Datenbindungsbibliothek immer noch auf diese zugreifen, ohne den vollständigen Namen der Methode zu verwenden.

Datenmodell

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Layout XML

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
       <variable name="item" type="com.example.Item"/>
   </data>

   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

       <!-- Since the "name" field is private on our data model,
            this binding will utilize the public getName() method instead. -->
       <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@{item.name}"/>

   </LinearLayout>
</layout>
```

Klassen referenzieren

Datenmodell

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Layout XML

Sie müssen referenzierte Klassen genauso importieren wie in Java.

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
       <import type="android.view.View"/>
       <variable name="item" type="com.example.Item"/>
   </data>

   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

       <!-- We reference the View class to set the visibility of this TextView -->
       <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@{item.name}"
           android:visibility="@{item.name == null ? View.VISIBLE : View.GONE"/>

   </LinearLayout>
</layout>
```

Hinweis: Das Paket `java.lang.*` Wird automatisch vom System importiert. (Das gleiche wird von `JVM` für `Java`)

Datenbindung in Fragment

Datenmodell

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name){
        this.name = name;
    }

}
```

Layout XML

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
       <variable name="item" type="com.example.Item"/>
   </data>

   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

       <TextView
           android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="@{item.name}"/>

    </LinearLayout>
</layout>
```

Fragment

```
@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();
}
```

Integrierte bidirektionale Datenbindung

Die bidirektionale Datenbindung unterstützt die folgenden Attribute:

Element	Eigenschaften
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

Element	Eigenschaften
Switch	android:checked

Verwendungszweck

```
<layout ...>
    <data>
        <variable type="com.example.myapp.User" name="user"/>
    </data>
    <RelativeLayout ...>
        <EditText android:text="@={user.firstName}" .../>
    </RelativeLayout>
</layout>
```

Beachten Sie, dass der Bindungsausdruck `@={}` **ein zusätzliches** `=` , das für die **bidirektionale Bindung** erforderlich ist. Methoden können nicht in bidirektionalen Bindungsausdrücken verwendet werden.

Datenbindung im RecyclerView Adapter

Es ist auch möglich, die Datenbindung in Ihrem `RecyclerView` Adapter zu verwenden.

Datenmodell

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

XML-Layout

```
<TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@{item.name}"/>
```

Adapterklasse

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

Klicken Sie auf Listener mit Bindung

Erstellen Sie eine Schnittstelle für clickHandler

```
public interface ClickHandler {
    public void onButtonClick(View v);
}
```

Layout XML

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

    <data>
        <variable
            name="handler"
            type="com.example.ClickHandler"/>
    </data>

    <RelativeLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="click me"
            android:onClick="@{handler.onButtonClick}"/>
    </RelativeLayout>
</layout>
```

Ereignis in Ihrer Aktivität behandeln

```
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();
    }
}
```

Benutzerdefiniertes Ereignis mit Lambda-Ausdruck

Schnittstelle definieren

```
public interface ClickHandler {
    public void onButtonClick(User user);
}
```

Erstellen Sie eine Modellklasse

```
public class User {
    private String name;

    public User(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Layout XML

```
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>
        <variable
            name="handler"
            type="com.example.ClickHandler"/>

        <variable
            name="user"
            type="com.example.User"/>
    </data>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{user.name}"
            android:onClick="@{() -> handler.onButtonClick(user)}"/>
    </RelativeLayout>
</layout>
```

Aktivitätscode:

```
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();
    }
}
```

Für einige Ansichtslistener, die nicht im XML-Code verfügbar sind, aber im Java-Code festgelegt werden können, kann sie mit benutzerdefinierten Bindungen gebunden werden.

Benutzerdefinierte Klasse

```
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);
    }
}
```

Handler-Klasse

```
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>
```

Standardwert in Datenbindung

> Der Vorschaufenster zeigt Standardwerte für Datenbindungsausdrücke an, sofern angegeben.

Zum Beispiel :

```
android:layout_height="@{@dimen/main_layout_height, default=wrap_content}"
```

Es `wrap_content` während des Entwurfs `wrap_content` und `wrap_content` im Vorschaufenster als `wrap_content` .

Ein anderes Beispiel ist

```
android:text="@{user.name, default=`Preview Text`}"
```

Es wird `Preview Text` im Vorschaufenster angezeigt. Wenn Sie ihn jedoch im Gerät / Emulator ausführen, wird der tatsächlich an ihn gebundene Text angezeigt

DataBinding mit benutzerdefinierten Variablen (int, boolean)

Manchmal müssen wir grundlegende Vorgänge ausführen, wie zum Beispiel die Ansicht verbergen / anzeigen, die auf einem einzelnen Wert basiert. Für diese einzelne Variable können wir kein Modell erstellen, oder es ist nicht ratsam, ein Modell dafür zu erstellen. DataBinding unterstützt grundlegende Datentypen, um diese Operationen auszuführen.

```
<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>
```

und seinen Wert von der Java-Klasse festlegen.

```
binding.setSelected(true);
```

Datenbindung im Dialog

```
public void doSomething() {
    DialogTestBinding binding = DataBindingUtil
            .inflate(LayoutInflater.from(context), R.layout.dialog_test, null, false);

    Dialog dialog = new Dialog(context);
    dialog.setContentView(binding.getRoot());
    dialog.show();
}
```

Übergeben Sie das Widget als Referenz in BindingAdapter

Layout XML

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

   </data>

   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

       <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

        <ImageView
        android:id="@+id/img"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        app:imageUrl="@{url}"
        app:progressbar="@{progressBar}"/>

   </LinearLayout>
</layout>
```

BindingAdapter-Methode

```
@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);
}
```

Datenbindungsbibliothek online lesen:
https://riptutorial.com/de/android/topic/111/datenbindungsbibliothek

Kapitel 66: Datensynchronisation mit dem Sync Adapter

Examples

Dummy-Sync-Adapter mit Stub-Provider

SyncAdapter

```
/**
 * Define a sync adapter for the app.
 * <p/>
 * <p>This class is instantiated in {@link SyncService}, which also binds SyncAdapter to the
system.
 * SyncAdapter should only be initialized in SyncService, never anywhere else.
 * <p/>
 * <p>The system calls onPerformSync() via an RPC call through the IBinder object supplied by
 * SyncService.
 */
class SyncAdapter extends AbstractThreadedSyncAdapter {
    /**
     * Constructor. Obtains handle to content resolver for later use.
     */
    public SyncAdapter(Context context, boolean autoInitialize) {
        super(context, autoInitialize);
    }

    /**
     * Constructor. Obtains handle to content resolver for later use.
     */
    public SyncAdapter(Context context, boolean autoInitialize, boolean allowParallelSyncs) {
        super(context, autoInitialize, allowParallelSyncs);
    }

    @Override
    public void onPerformSync(Account account, Bundle extras, String authority,
                              ContentProviderClient provider, SyncResult syncResult) {
        //Jobs you want to perform in background.
        Log.e("" + account.name, "Sync Start");
    }

}
```

Synchronisierungsdienst

```
/**
 * Define a Service that returns an IBinder for the
 * sync adapter class, allowing the sync adapter framework to call
 * onPerformSync().
 */
public class SyncService extends Service {
    // Storage for an instance of the sync adapter
    private static SyncAdapter sSyncAdapter = null;
    // Object to use as a thread-safe lock
```

```
    private static final Object sSyncAdapterLock = new Object();

    /*
     * Instantiate the sync adapter object.
     */
    @Override
    public void onCreate() {
        /*
         * Create the sync adapter as a singleton.
         * Set the sync adapter as syncable
         * Disallow parallel syncs
         */
        synchronized (sSyncAdapterLock) {
            if (sSyncAdapter == null) {
                sSyncAdapter = new SyncAdapter(getApplicationContext(), true);
            }
        }
    }

    /**
     * Return an object that allows the system to invoke
     * the sync adapter.
     */
    @Override
    public IBinder onBind(Intent intent) {
        /*
         * Get the object that allows external processes
         * to call onPerformSync(). The object is created
         * in the base class code when the SyncAdapter
         * constructors call super()
         */
        return sSyncAdapter.getSyncAdapterBinder();
    }
}
```

Authenticator

```
public class Authenticator extends AbstractAccountAuthenticator {
    // Simple constructor
    public Authenticator(Context context) {
        super(context);
    }

    // Editing properties is not supported
    @Override
    public Bundle editProperties(
            AccountAuthenticatorResponse r, String s) {
        throw new UnsupportedOperationException();
    }

    // Don't add additional accounts
    @Override
    public Bundle addAccount(
            AccountAuthenticatorResponse r,
            String s,
            String s2,
            String[] strings,
            Bundle bundle) throws NetworkErrorException {
        return null;
    }
```

```
    // Ignore attempts to confirm credentials
    @Override
    public Bundle confirmCredentials(
            AccountAuthenticatorResponse r,
            Account account,
            Bundle bundle) throws NetworkErrorException {
        return null;
    }

    // Getting an authentication token is not supported
    @Override
    public Bundle getAuthToken(
            AccountAuthenticatorResponse r,
            Account account,
            String s,
            Bundle bundle) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }

    // Getting a label for the auth token is not supported
    @Override
    public String getAuthTokenLabel(String s) {
        throw new UnsupportedOperationException();
    }

    // Updating user credentials is not supported
    @Override
    public Bundle updateCredentials(
            AccountAuthenticatorResponse r,
            Account account,
            String s, Bundle bundle) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }

    // Checking features for the account is not supported
    @Override
    public Bundle hasFeatures(
            AccountAuthenticatorResponse r,
            Account account, String[] strings) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }
}
```

Authenticator Service

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

Ergänzungen zu AndroidManifest.xml

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.READ_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />

     <service
        android:name=".syncAdapter.SyncService"
        android:exported="true">
        <intent-filter>
            <action android:name="android.content.SyncAdapter" />
        </intent-filter>
        <meta-data
            android:name="android.content.SyncAdapter"
            android:resource="@xml/syncadapter" />
    </service>

    <service android:name=".authenticator.AuthenticatorService">
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator" />
        </intent-filter>
        <meta-data
            android:name="android.accounts.AccountAuthenticator"
            android:resource="@xml/authenticator" />
    </service>

    <provider
        android:name=".provider.StubProvider"
        android:authorities="com.yourpackage.provider"
        android:exported="false"
        android:syncable="true" />
```

res / xml / authenticator.xml

```
<?xml version="1.0" encoding="utf-8"?>
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.yourpackage"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:smallIcon="@mipmap/ic_launcher" />
```

res / xml / syncadapter.xml

```
<?xml version="1.0" encoding="utf-8"?>
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.yourpackage.android"
    android:allowParallelSyncs="false"
    android:contentAuthority="com.yourpackage.provider"
    android:isAlwaysSyncable="true"
    android:supportsUploading="false"
```

```
    android:userVisible="false" />
```

StubProvider

```
/*
 * Define an implementation of ContentProvider that stubs out
 * all methods
 */
public class StubProvider extends ContentProvider {
    /*
     * Always return true, indicating that the
     * provider loaded correctly.
     */
    @Override
    public boolean onCreate() {
        return true;
    }

    /*
     * Return no type for MIME type
     */
    @Override
    public String getType(Uri uri) {
        return null;
    }

    /*
     * query() always returns no results
     *
     */
    @Override
    public Cursor query(
            Uri uri,
            String[] projection,
            String selection,
            String[] selectionArgs,
            String sortOrder) {
        return null;
    }

    /*
     * insert() always returns null (no URI)
     */
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        return null;
    }

    /*
     * delete() always returns "no rows affected" (0)
     */
    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        return 0;
    }

    /*
     * update() always returns "no rows affected" (0)
     */
    public int update(
```

```
            Uri uri,
            ContentValues values,
            String selection,
            String[] selectionArgs) {
        return 0;
    }
}
```

Rufen Sie diese Funktion bei erfolgreicher Anmeldung auf, um ein Konto mit der angemeldeten Benutzer-ID zu erstellen

```
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;
    }
```

Eine Synchronisierung erzwingen

```
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);
```

Datensynchronisation mit dem Sync Adapter online lesen:
https://riptutorial.com/de/android/topic/1944/datensynchronisation-mit-dem-sync-adapter

Kapitel 67: Datenverschlüsselung / Entschlüsselung

Einführung

In diesem Thema wird beschrieben, wie Verschlüsselung und Entschlüsselung in Android funktionieren.

Examples

AES-Verschlüsselung von Daten mit Passwort auf sichere Weise

Im folgenden Beispiel wird ein bestimmter Datenblock mit AES verschlüsselt. Der Verschlüsselungsschlüssel wird auf sichere Weise abgeleitet (zufälliges Salt, 1000 SHA-256-Runden). Die Verschlüsselung verwendet AES im CBC- Modus mit zufälliger IV .

Beachten Sie, dass die in der Klasse `EncryptedData` (`salt` , `iv` und `encryptedData`) gespeicherten Daten zu einem einzelnen Byte-Array verkettet werden können. Sie können die Daten dann speichern oder an den Empfänger übertragen.

```
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;
}
```

Der folgende Beispielcode zeigt, wie Sie die Verschlüsselung und Entschlüsselung testen:

```
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();
}
```

Datenverschlüsselung / Entschlüsselung online lesen:
https://riptutorial.com/de/android/topic/3471/datenverschlusselung---entschlusselung

Kapitel 68: Datums- und Uhrzeitauswahl

Examples

Material DatePicker

`build.gradle` der `build.gradle` Datei im Abhängigkeitsbereich folgende Abhängigkeiten hinzu. (Dies ist eine nicht offizielle Bibliothek für die Datumsauswahl)

```
  compile 'com.wdullaer:materialdatetimepicker:2.3.0'
```

Jetzt müssen wir das `DatePicker` Ereignis beim Button click öffnen.

Erstellen Sie also einen Button in der XML-Datei wie unten.

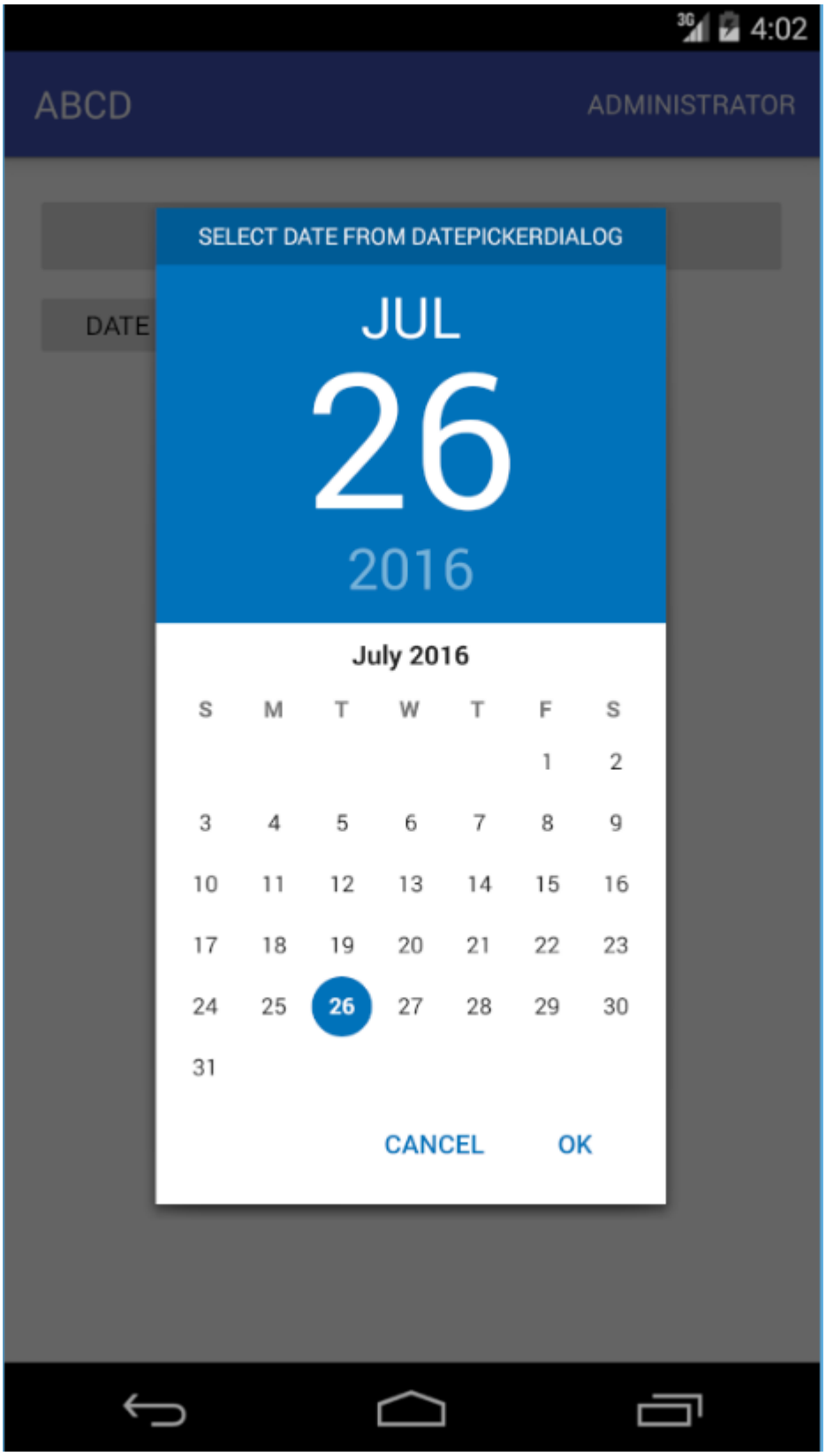
```
     <Button
            android:id="@+id/dialog_bt_date"
            android:layout_below="@+id/resetButton"
            android:layout_width="wrap_content"
            android:layout_height="40dp"
            android:textColor="#FF000000"
            android:gravity="center"
            android:text="DATE"/>
```

und in MainActivity verwenden Sie diesen Weg.

```
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;
    }

}
```

Ausgabe :



Datumsauswahldialog

In diesem Dialogfeld wird der Benutzer aufgefordert, das Datum mithilfe von `DatePicker` auszuwählen. Der Dialog erfordert Kontext, Anfangsjahr, Monat und Tag, um den Dialog mit dem Startdatum anzuzeigen. Wenn der Benutzer das Datum auswählt, `DatePickerDialog.OnDateSetListener` er über `DatePickerDialog.OnDateSetListener` .

```
public void showDatePicker(Context context,int initialYear, int initialMonth, int initialDay)
{
       DatePickerDialog datePickerDialog = new DatePickerDialog(context,
                new DatePickerDialog.OnDateSetListener() {
                    @Override
                    public void onDateSet(DatePicker datepicker,int year ,int month, int day)
{
                    //this condition is necessary to work properly on all android versions
                    if(view.isShown()){
                            //You now have the selected year, month and day
                    }

}
                }, initialYear, initialMonth , initialDay);

        //Call show() to simply show the dialog
        datePickerDialog.show();

    }
```

Bitte beachten Sie, dass der Monat einen Monat beginnt, der von 0 für Januar bis 11 für Dezember beginnt

Datums- und Uhrzeitauswahl online lesen: https://riptutorial.com/de/android/topic/2836/datums--und-uhrzeitauswahl

Kapitel 69: DayNight-Theme (AppCompat v23.2 / API 14+)

Examples

Hinzufügen des DayNight-Designs zu einer App

Das DayNight-Design gibt einer App die coole Möglichkeit, Farbschemen basierend auf der Tageszeit und dem letzten bekannten Standort des Geräts zu wechseln.

Fügen Sie Ihrer `styles.xml` Folgendes `styles.xml` :

```
<style name="AppTheme" parent="Theme.AppCompat.DayNight">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
</style>
```

Die Themen, die Sie erweitern können, um die Funktionalität zum Wechseln des Tags bei Nacht hinzuzufügen, sind die folgenden:

- `"Theme.AppCompat.DayNight"`
- `"Theme.AppCompat.DayNight.NoActionBar"`
- `"Theme.AppCompat.DayNight.DarkActionBar"`

Neben `colorPrimary` , `colorPrimaryDark` und `colorAccent` können Sie auch andere Farben hinzufügen, die Sie `textColorPrimary` `textColorSecondary` , z. B. `textColorPrimary` oder `textColorSecondary` . Sie können diesem `style` auch die benutzerdefinierten Farben Ihrer App hinzufügen.

Damit das Theme funktionieren kann, müssen Sie eine Standardfarbe `colors.xml` im Verzeichnis `res/values` und eine andere `colors.xml` im `colors.xml` `res/values-night` definieren und die Farben für Tag und Nacht entsprechend definieren.

Rufen Sie zum `AppCompatDelegate.setDefaultNightMode(int)` des `AppCompatDelegate.setDefaultNightMode(int)` die `AppCompatDelegate.setDefaultNightMode(int)` -Methode in Ihrem Java-Code auf. (Dadurch wird das Farbschema für die gesamte App geändert, nicht nur für eine Aktivität oder ein Fragment.) Zum Beispiel:

```
AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
```

Sie können je nach Wahl eine der folgenden drei Optionen passieren:

- `AppCompatDelegate.MODE_NIGHT_NO` : `AppCompatDelegate.MODE_NIGHT_NO` das Standarddesign für Ihre App festgelegt und die im Verzeichnis `res/values` definierten Farben übernommen. Es wird empfohlen, helle Farben für dieses Thema zu verwenden.

- `AppCompatDelegate.MODE_NIGHT_YES` : `AppCompatDelegate.MODE_NIGHT_YES` ein `AppCompatDelegate.MODE_NIGHT_YES` für Ihre App festgelegt und die im Verzeichnis `res/values-night` definierten Farben übernommen. Es wird empfohlen, dunkle Farben für dieses Thema zu verwenden.
- `AppCompatDelegate.MODE_NIGHT_AUTO` : Diese `AppCompatDelegate.MODE_NIGHT_AUTO` automatisch die Farben der App basierend auf der Tageszeit und den Farben, die Sie in `values` und `values-night` .

Es ist auch möglich, den aktuellen Nachtmodus mithilfe der `getDefaultNightMode()` -Methode `getDefaultNightMode()` . Zum Beispiel:

```
int modeType = AppCompatDelegate.getDefaultNightMode();
```

Beachten Sie jedoch, dass der Themenwechsel nicht bestehen bleibt, wenn Sie die App beenden und erneut öffnen. Wenn Sie das tun, wechselt das `AppCompatDelegate.MODE_NIGHT_AUTO` wieder zu `AppCompatDelegate.MODE_NIGHT_AUTO` , dem Standardwert. Wenn Sie möchten, dass der Design-Schalter bestehen bleibt, müssen Sie den Wert in den gemeinsamen Einstellungen speichern und den gespeicherten Wert jedes Mal laden, wenn die App nach der Zerstörung geöffnet wird.

DayNight-Theme (AppCompat v23.2 / API 14+) online lesen:
https://riptutorial.com/de/android/topic/7650/daynight-theme--appcompat-v23-2---api-14plus-

Kapitel 70: Definieren Sie den Schrittwert (Inkrement) für die benutzerdefinierte RangeSeekBar

Einführung

Eine Anpassung der Android RangeSeekBar von Alex Florescu unter https://github.com/anothem/android-range-seek-bar

Sie können einen Schrittwert (Inkrement) festlegen, wenn Sie den Suchbalken verschieben

Bemerkungen

1- Fügen Sie das Attribut inkrement in attrs.xml hinzu

```
<attr name="increment" format="integer|float"/>
```

2- Definieren Sie einen Standardwert in RangeSeekBar.java und erstellen Sie auch das Attribut

```
private static final int DEFAULT_INCREMENT = 1;
private int increment;
```

3- Init der Inkrementierungswert in privatem void init (Kontextkontext, AttributeSet-Attribute)

```
if (attrs == null)
    increment = DEFAULT_INCREMENT;
else
    increment = a.getInt(R.styleable.RangeSeekBar_increment, DEFAULT_INCREMENT);
```

4- Definieren Sie den Inkrementwert in geschütztem synchronisiertem void onDraw (@NonNull Canvas Canvas

Sie müssen den Wert für minText und maxText ersetzen. Also statt:

- minText = valueToString (getSelectedMinValue ());
- maxText = valueToString (getSelectedMaxValue ());

Sie haben: 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 - Jetzt müssen Sie es nur noch verwenden. Ich hoffe es hilft

Examples

Definieren Sie einen Schrittwert von 7

```
<RangeSeekBar
        android:id="@+id/barPrice"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        app:barHeight="0.2dp"
        app:barHeight2="4dp"
        app:increment="7"
        app:showLabels="false" />
```

Definieren Sie den Schrittwert (Inkrement) für die benutzerdefinierte RangeSeekBar online lesen: https://riptutorial.com/de/android/topic/8627/definieren-sie-den-schrittwert--inkrement--fur-die-benutzerdefinierte-rangeseekbar

Kapitel 71: Designmuster

Einführung

Entwurfsmuster sind formalisierte Best Practices, die der Programmierer verwenden kann, um häufige Probleme beim Entwurf einer Anwendung oder eines Systems zu lösen.

Entwurfsmuster können den Entwicklungsprozess beschleunigen, indem bewährte Entwicklungsparadigmen bereitgestellt werden.

Durch die Wiederverwendung von Entwurfsmustern werden subtile Probleme vermieden, die zu großen Problemen führen können. Außerdem wird die Lesbarkeit von Code für Codierer und Architekten verbessert, die mit den Mustern vertraut sind.

Examples

Singleton-Klassenbeispiel

Java Singleton Pattern

Um das Singleton-Muster zu implementieren, haben wir unterschiedliche Ansätze, aber alle haben gemeinsame Konzepte.

- Privater Konstruktor, um die Instantiierung der Klasse von anderen Klassen zu beschränken.
- Private statische Variable derselben Klasse, die die einzige Instanz der Klasse ist.
- Öffentliche statische Methode, die die Instanz der Klasse zurückgibt. Dies ist der globale Zugriff
- Punkt für die äußere Welt, um die Instanz der Singleton-Klasse zu erhalten.

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

Beobachtermuster

Das Beobachtermuster ist ein allgemeines Muster, das in vielen Zusammenhängen weit verbreitet ist. Ein reales Beispiel kann aus YouTube entnommen werden: Wenn Sie einen Kanal mögen und alle Nachrichten abrufen und neue Videos von diesem Kanal ansehen möchten, müssen Sie diesen Kanal abonnieren. Immer wenn dieser Kanal Neuigkeiten veröffentlicht, erhalten Sie (und alle anderen Abonnenten) eine Benachrichtigung.

Ein Beobachter hat zwei Komponenten. Einer ist ein Sender (Kanal) und der andere ist ein Empfänger (Sie oder ein anderer Teilnehmer). Der Sender wird alle Empfängerinstanzen behandeln, die ihn abonniert haben. Wenn der Sender ein neues Ereignis auslöst, wird dies allen Empfängerinstanzen mitgeteilt. Wenn der Empfänger ein Ereignis empfängt, muss er auf dieses Ereignis reagieren, z. B. indem Sie YouTube einschalten und das neue Video abspielen.

Beobachtermuster implementieren

1. Der Sender muss Methoden zur Verfügung stellen, die es Empfängern ermöglichen, ihn zu abonnieren und abzubestellen. Wenn der Sender ein Ereignis auslöst, müssen die Abonnenten darüber informiert werden, dass ein Ereignis aufgetreten ist:

```
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. Der Empfänger muss eine Methode implementieren, die das Ereignis vom Sender übernimmt:

```
interface Subscriber {
    void doSubscribe(Channel channel);
    void doUnsubscribe(Channel channel);
    void handleEvent();  // Process the new event.
}
```

Designmuster online lesen: https://riptutorial.com/de/android/topic/9949/designmuster

Kapitel 72: Dialog

Parameter

Linie	Beschreibung
Show();	Zeigt den Dialog
setContentView (R.layout.yourlayout);	Setzt die ContentView des Dialogs auf Ihr benutzerdefiniertes Layout.
entlassen()	Schließt den Dialog

Bemerkungen

- Das Dialogfeld im ersten Beispiel (Dialog) muss `show()` nicht aufrufen `show()` wenn es bei der Verarbeitung im Konstruktor erstellt wird
- Alert Dialogs müssen durch eine neue Instanz der `AlertDialog.Builder()` Klasse erstellt werden. Dem Builder-Muster folgend können alle Mitglieder des AlertDialog.Builder-Verfahrens mit einer Kette verknüpft werden, um die Dialoginstanz 'aufzubauen'.
- Der Alert - Dialog Builder kann direkt `show()` den Dialog - Sie brauchen nicht zu nennen `create()` dann `show()` auf der Alertdialog Instanz

Examples

Alert-Dialog

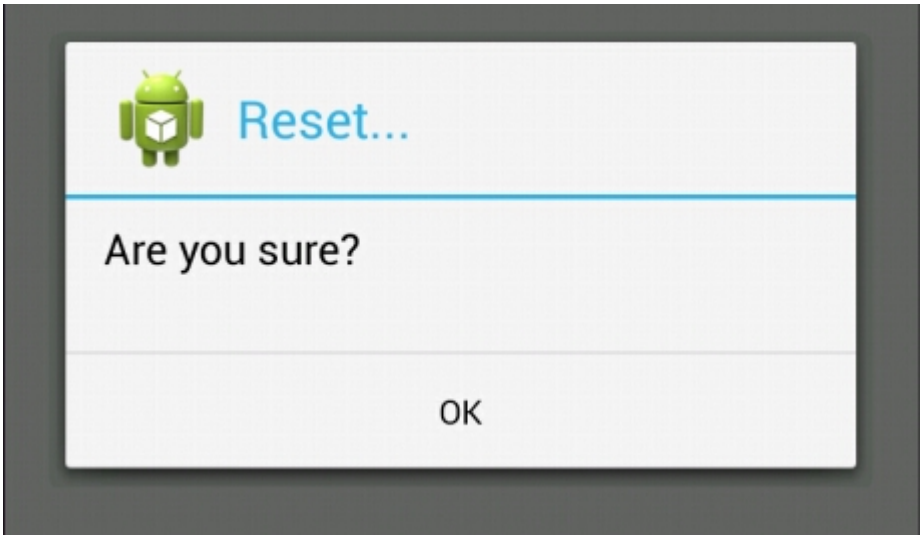
```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
                MainActivity.this);

        alertDialogBuilder.setTitle("Title Dialog");
        alertDialogBuilder
                .setMessage("Message Dialog")
                .setCancelable(true)
                .setPositiveButton("Yes",
                        new DialogInterface.OnClickListener() {

                            public void onClick(DialogInterface dialog, int arg1) {
                                // Handle Positive Button

                            }
                        })
                .setNegativeButton("No",
                        new DialogInterface.OnClickListener() {

                            public void onClick(DialogInterface dialog, int arg1) {
```

```
                                // Handle Negative Button
                                dialog.cancel();
                            }
                        });

        AlertDialog alertDialog = alertDialogBuilder.create();
        alertDialog.show();
```

Ein grundlegender Alert-Dialog

```
    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.
```

Jetzt wird dieser Code dies erreichen:



(*Bildquelle: WikiHow*)

Datumsauswahl in DialogFragment

XML des Dialogs:

```
<?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>
```

Dialogklasse:

```
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;
    }

}
```

Aktivität, die den Dialog aufruft:

```
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` ist die einfachste Möglichkeit, `DatePicker` zu verwenden, da Sie Dialoge an jeder beliebigen Stelle in Ihrer App `DatePicker` können. Sie müssen kein eigenes Layout mit dem `DatePicker` Widget `DatePicker` .

So zeigen Sie den Dialog:

```
DatePickerDialog datePickerDialog = new DatePickerDialog(context, listener, year, month, day);
datePickerDialog.show();
```

Sie können das `DataPicker` Widget aus dem obigen Dialog `DataPicker` , um auf weitere Funktionen zuzugreifen und beispielsweise ein Mindestdatum in Millisekunden `DataPicker` :

```
DatePicker datePicker = datePickerDialog.getDatePicker();
datePicker.setMinDate(System.currentTimeMillis());
```

Datumsauswahl

`DatePicker` kann der Benutzer das Datum auswählen. Wenn wir eine neue Instanz von `DatePicker` , können wir das Startdatum festlegen. Wenn wir kein Anfangsdatum festlegen, wird das aktuelle Datum standardmäßig festgelegt.

Wir können `DatePicker` dem Benutzer `DatePicker` , indem Sie `DatePickerDialog` oder unser eigenes Layout mit dem `DatePicker` Widget `DatePicker` .

Außerdem können wir den Datumsbereich einschränken, den der Benutzer auswählen kann.

Durch Einstellen des Mindestdatums in Millisekunden

```
//In this case user can pick date only from future
datePicker.setMinDate(System.currentTimeMillis());
```

Durch Festlegen des maximalen Datums in Millisekunden

```
//In this case user can pick date only, before following week.
datePicker.setMaxDate(System.currentTimeMillis() + TimeUnit.DAYS.toMillis(7));
```

Um Informationen zu erhalten, welches Datum vom Benutzer ausgewählt wurde, müssen wir `Listener` .

Wenn wir `DatePickerDialog` , können wir `OnDateSetListener` im Konstruktor setzen, wenn wir eine neue Instanz von `DatePickerDialog` :

Verwendungsbeispiel von `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) {

    }
}
```

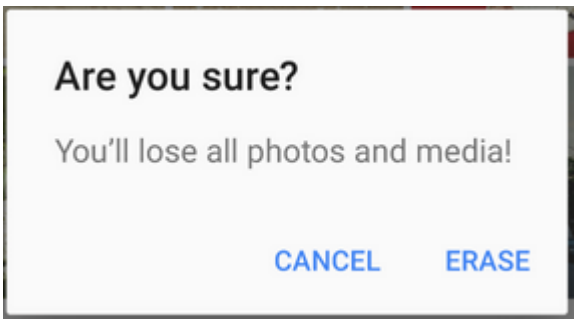
Ansonsten, wenn wir unser eigenes Layout mit dem `DatePicker` Widget erstellen, müssen wir auch einen eigenen Listener erstellen, wie er in einem anderen Beispiel gezeigt wurde

Hinzufügen von Material Design AlertDialog zu Ihrer App mit Appcompat

`AlertDialog` ist eine Unterklasse von `Dialog` , die eine, zwei oder drei Schaltflächen anzeigen kann. Wenn Sie nur eine `setMessage()` in diesem Dialogfeld anzeigen möchten, verwenden Sie die `setMessage()` -Methode.

Das `AlertDialog` Paket von `android.app` wird auf verschiedenen Android-Betriebssystemversionen unterschiedlich `android.app` .

Die Android V7 Appcompat-Bibliothek bietet eine `AlertDialog` Implementierung, die in allen unterstützten Android-Betriebssystemversionen zusammen mit Material Design angezeigt wird:



Zuerst müssen Sie die V7 Appcompat-Bibliothek zu Ihrem Projekt hinzufügen. Sie können dies in der App-Ebene build.gradle tun:

```
dependencies {
```

```
    compile 'com.android.support:appcompat-v7:24.2.1'
    //........
}
```

Stellen Sie sicher, dass Sie die richtige Klasse importieren:

```
import android.support.v7.app.AlertDialog;
```

Dann erstellen Sie einen AlertDialog wie folgt:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Are you sure?");
builder.setMessage("You'll lose all photos and media!");
builder.setPositiveButton("ERASE", null);
builder.setNegativeButton("CANCEL", null);
builder.show();
```

ListView in AlertDialog

Wir können `ListView` oder `RecyclerView` für die Auswahl aus der Liste der Elemente verwenden. Wenn wir jedoch nur eine geringe Anzahl von Auswahlmöglichkeiten haben, können wir `AlertDialog.Builder setAdapter` .

```
    private void showDialog()
    {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Choose any item");

        final List<String> lables = new ArrayList<>();
        lables.add("Item 1");
        lables.add("Item 2");
        lables.add("Item 3");
        lables.add("Item 4");

        ArrayAdapter<String> dataAdapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line, lables);
        builder.setAdapter(dataAdapter, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(MainActivity.this,"You have selected " +
lables.get(which),Toast.LENGTH_LONG).show();
            }
        });
        AlertDialog dialog = builder.create();
        dialog.show();
    }
```

Wenn wir keine spezielle `ListView` benötigen, können wir vielleicht eine grundlegende Methode verwenden:

```
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();
```

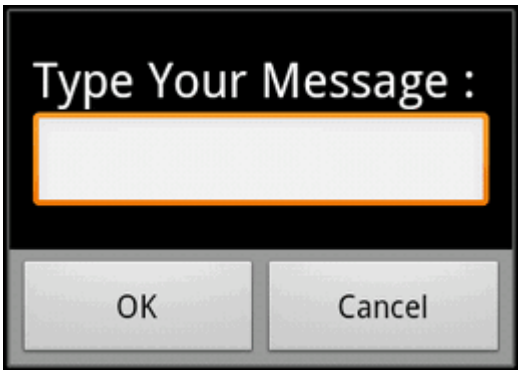
Benutzerdefinierter Alert-Dialog mit EditText

```
void alertDialogDemo() {
        // get alert_dialog.xml view
        LayoutInflater li = LayoutInflater.from(getApplicationContext());
        View promptsView = li.inflate(R.layout.alert_dialog, null);

        AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
                getApplicationContext());

        // set alert_dialog.xml to alertdialog builder
        alertDialogBuilder.setView(promptsView);

        final EditText userInput = (EditText) promptsView.findViewById(R.id.etUserInput);

        // set dialog message
        alertDialogBuilder
                .setCancelable(false)
                .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        // get user input and set it to result
                        // edit text
                        Toast.makeText(getApplicationContext(), "Entered:
"+userInput.getText().toString(), Toast.LENGTH_LONG).show();
                    }
                })
                .setNegativeButton("Cancel",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                dialog.cancel();
                            }
                        });

        // create alert dialog
        AlertDialog alertDialog = alertDialogBuilder.create();

        // show it
        alertDialog.show();
    }
```

XML-Datei: 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>
```



Fullscreen Custom Dialog ohne Hintergrund und ohne Titel

In `styles.xml` fügen Sie Ihren eigenen Stil hinzu:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppBaseTheme" parent="@android:style/Theme.Light.NoTitleBar.Fullscreen">
    </style>
</resources>
```

Erstellen Sie Ihr benutzerdefiniertes Layout für den Dialog: `fullscreen.xml` :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

</RelativeLayout>
```

Dann können Sie es in der Java-Datei für eine Aktivität oder einen Dialog verwenden.

```
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();
    }
}
```

Alert-Dialog mit mehrzeiligem Titel

Mit der setCustomTitle () - Methode von AlertDialog.Builder können Sie eine beliebige Ansicht für den Dialogtitel angeben. Eine übliche Verwendung für diese Methode ist das Erstellen eines Warnungsdialogfelds mit einem langen Titel.

```
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();
```

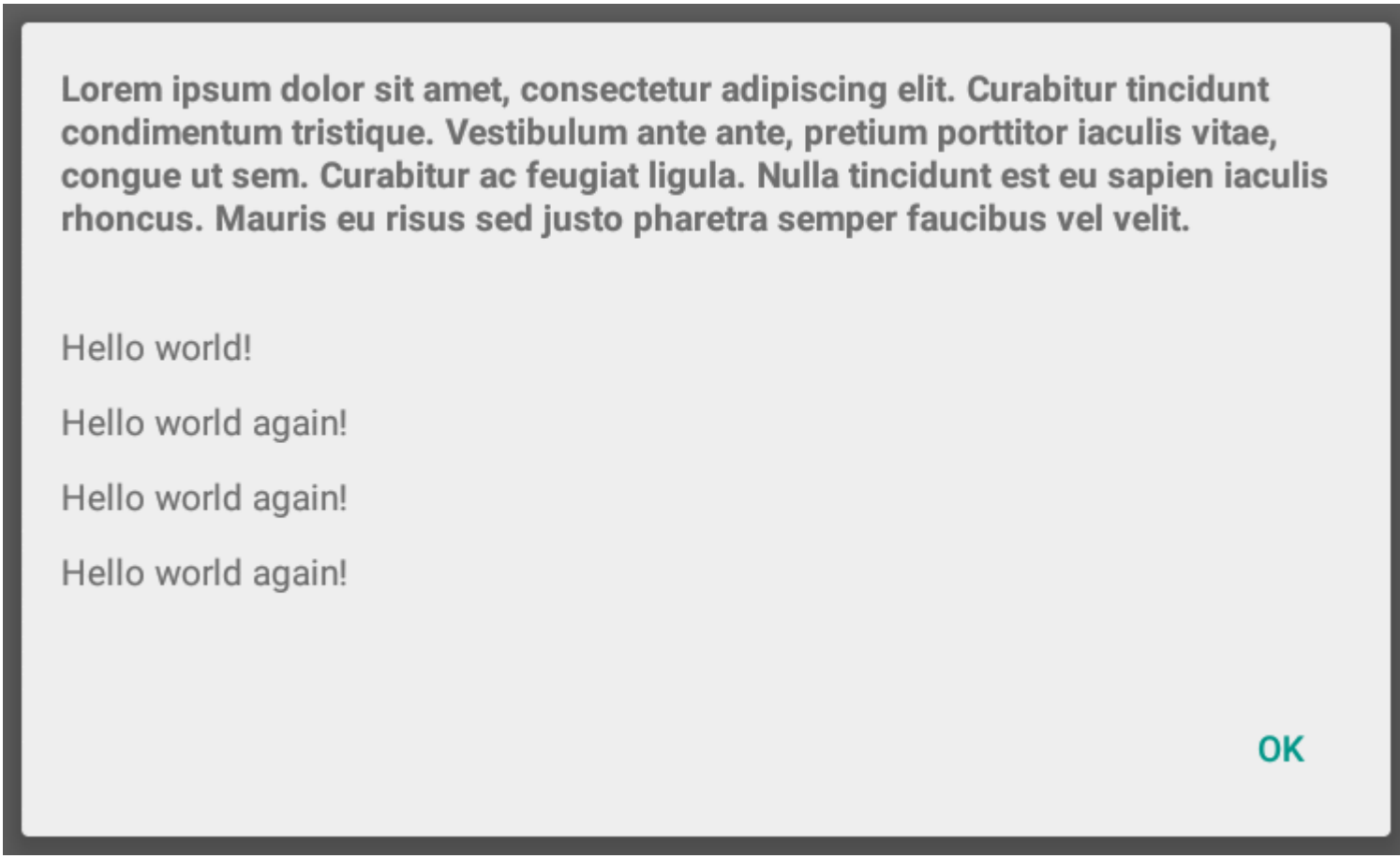
my_dialog_title.xml:

```
<?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>
```



Dialog online lesen: https://riptutorial.com/de/android/topic/1225/dialog

Kapitel 73: Die Manifestdatei

Einführung

Das Manifest ist eine obligatorische Datei mit dem Namen "AndroidManifest.xml", die sich im Stammverzeichnis der App befindet. Sie gibt den App-Namen, das Symbol, den Java-Paketnamen, die Version, die Deklaration von Aktivitäten, Services, App-Berechtigungen und andere Informationen an.

Examples

Komponenten deklarieren

Die Hauptaufgabe des Manifests besteht darin, das System über die Komponenten der App zu informieren. Eine Manifestdatei kann beispielsweise eine Aktivität wie folgt deklarieren:

```
<?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>
```

Im Element `<application>` zeigt das Attribut `android:icon` auf Ressourcen für ein Symbol, das die App identifiziert.

In dem Element gibt das Attribut `android:name` den vollständig qualifizierten Klassennamen der Activity-Unterklasse an, und das Attribut android: label gibt eine Zeichenfolge an, die als vom Benutzer sichtbares Label für die Aktivität verwendet werden soll.

Sie müssen alle App-Komponenten auf diese Weise deklarieren:

- `<activity>` -Elemente für Aktivitäten

- `<service>` -Elemente für Dienste

- `<receiver>` -Elemente für Rundfunkempfänger

- `<provider>` -Elemente für Inhaltsanbieter

Aktivitäten, Dienste und Inhaltsanbieter, die Sie in Ihre Quelle aufnehmen, jedoch nicht im Manifest deklarieren, sind für das System nicht sichtbar und können daher nicht ausgeführt werden. Broadcast-Empfänger können jedoch entweder im Manifest deklariert oder dynamisch als Code erstellt werden (als `BroadcastReceiver` Objekte) und durch Aufrufen von `registerReceiver()` beim System `registerReceiver()` .

Weitere Informationen zum Strukturieren der Manifestdatei für Ihre App finden Sie in der Dokumentation zur AndroidManifest.xml-Datei.

Deklarieren von Berechtigungen in Ihrer Manifestdatei

Jede von Ihrer Anwendung für den Zugriff auf einen geschützten Teil der API oder für die Interaktion mit anderen Anwendungen erforderliche `AndroidManifest.xml` muss in Ihrer `AndroidManifest.xml` Datei angegeben werden. Dies geschieht mit dem Tag `<uses-permission />` .

Syntax

```
<uses-permission android:name="string"
   android:maxSdkVersion="integer"/>
```

android: name: Dies ist der Name der erforderlichen Berechtigung

android: maxSdkVersion: Die höchste API-Ebene, auf der diese Berechtigung für Ihre App erteilt werden soll. Das Festlegen dieser Berechtigung ist optional und sollte nur festgelegt werden, wenn die Berechtigung, die Ihre App benötigt, auf einer bestimmten API-Ebene nicht mehr benötigt wird.

Beispiel 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>
```

* Siehe auch das Thema Berechtigungen .

Die Manifestdatei online lesen: https://riptutorial.com/de/android/topic/1848/die-manifestdatei

Kapitel 74: Dolch 2

Syntax

- @Modul
- @Component (Abhängigkeiten = {OtherComponent.class}, modules = {ModuleA.class, ModuleB.class})
- DaggerMyComponent.create ()
- DaggerMyComponent.builder (). MyModule (newMyModule ()). Create ()

Bemerkungen

Nicht mit Dolch für Quadrat zu verwechseln, der Vorgänger von Dolch 2

Examples

Komponenteneinrichtung für Applikation und Aktivitätsinjektion

Eine grundlegende `AppComponent` , die von einem einzelnen `AppModule` , um anwendungsweite Singleton-Objekte bereitzustellen.

```
@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {

    void inject(App app);

    Context provideContext();

    Gson provideGson();
}
```

Ein Modul, das zusammen mit `AppComponent` wird und dessen Einzelobjekte bereitstellt, z. B. eine `Gson` Instanz, die in der gesamten Anwendung wiederverwendet werden kann.

```
@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;
    }
}
```

Eine untergeordnete Anwendung zum Einrichten von Dolch und der Einzelkomponente.

```
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;
    }
}
```

Jetzt eine Komponente mit Aktivitätsbereich, die von der `AppComponent` abhängig ist, um Zugriff auf die Einzelobjekte zu erhalten.

```
@ActivityScope
@Component(dependencies = AppComponent.class, modules = ActivityModule.class)
public interface MainActivityComponent {

    void inject(MainActivity activity);
}
```

Und ein wiederverwendbares `ActivityModule` , das grundlegende Abhängigkeiten wie einen `FragmentManager` bereitstellt

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

Wenn wir alles zusammenstellen, sind wir aufgestellt und können unsere Aktivitäten `Gson` lassen.

```
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);
    }
}
```

Benutzerdefinierte Bereiche

```
@Scope
@Documented
@Retention(RUNTIME)
public @interface ActivityScope {
}
```

Bereiche sind lediglich Anmerkungen und Sie können bei Bedarf eigene erstellen.

Konstruktorinjektion

Klassen ohne Abhängigkeiten können einfach mit dem Dolch erstellt werden.

```
public class Engine {

    @Inject // <-- Annotate your constructor.
    public Engine() {
    }
}
```

Diese Klasse kann von *jeder* Komponente bereitgestellt werden. Es hat selbst *keine Abhängigkeiten* und ist *nicht im Geltungsbereich* . Es ist kein weiterer Code notwendig.

Abhängigkeiten werden im Konstruktor als Parameter deklariert. Dagger ruft den Konstruktor auf und liefert die Abhängigkeiten, sofern diese Abhängigkeiten bereitgestellt werden können.

```
public class Car {
```

```
    private Engine engine;

    @Inject
    public Car(Engine engine) {
        this.engine = engine;
    }
}
```

Diese Klasse kann von jeder Komponente bereitgestellt werden, *wenn* diese Komponente auch alle ihre Abhängigkeiten bereitstellen kann. `Engine` in diesem Fall. Da der `Engine` auch Konstruktor eingespritzt werden kann, kann *jede* Komponente ein `Car` bereitstellen.

Sie können die Konstruktorinjektion immer dann verwenden, wenn alle Abhängigkeiten von der Komponente bereitgestellt werden können. Eine Komponente kann eine Abhängigkeit bereitstellen, wenn

- Es kann es mit Konstruktorinjektion erstellen
- ein Modul der Komponente kann es bereitstellen
- es kann von der übergeordneten Komponente bereitgestellt werden (wenn es sich um eine `@Subcomponent`)
- Es kann ein Objekt verwenden, das von einer Komponente verfügbar gemacht wird, von der es abhängig ist (Abhängigkeit von Komponenten).

Verwendung von @Subcomponent anstelle von @Component (Abhängigkeiten = {...})

```
@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);
```

```
    }
}
```

So fügen Sie Dagger 2 in build.gradle hinzu

Seit der Veröffentlichung von Gradle 2.2 wird das Plug-in für Android-apt nicht mehr verwendet. Die folgende Methode zum Einrichten von Dolch 2 sollte verwendet werden. Verwenden Sie für ältere Versionen von Gradle die zuvor gezeigte Methode.

Für Gradle> = 2.2

```
dependencies {
    // apt command comes from the android-apt plugin
    annotationProcessor 'com.google.dagger:dagger-compiler:2.8'
    compile 'com.google.dagger:dagger:2.8'
    provided 'javax.annotation:jsr250-api:1.0'
}
```

Für Gradle <2.2

Um Dagger 2 verwenden zu können, müssen Sie das `android-apt` Plugin hinzufügen. Fügen Sie dies dem root build.gradle hinzu:

```
buildscript {
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.0'
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}
```

Das build.gradle des Anwendungsmoduls sollte dann Folgendes enthalten:

```
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}"
}
```

Referenz: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Eine Komponente aus mehreren Modulen erstellen

Dolch 2 unterstützt das Erstellen einer Komponente aus mehreren Modulen. Sie können Ihre

Komponente auf diese Weise erstellen:

```
@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);
}
```

Die zwei Referenzmodule `GeneralPurposeModule` und `SpecificModule` können dann wie folgt implementiert werden:

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;
    }
```

```
}
```

Während der Abhängigkeitseingriffsphase entnimmt die Komponente den beiden Objekten Objekte entsprechend den Anforderungen.

Dieser Ansatz ist im Hinblick auf die *Modularität* sehr nützlich. In dem Beispiel gibt es ein Universalmodul, das zum Instanziieren von Komponenten verwendet wird, wie z. B. das `Retrofit` Objekt (zur Verarbeitung der Netzwerkkommunikation) und ein `PropertiesReader` (für die Verarbeitung von Konfigurationsdateien). Es gibt auch ein spezielles Modul, das die Instantiierung bestimmter Controller und Serviceklassen in Bezug auf diese bestimmte Anwendungskomponente abwickelt.

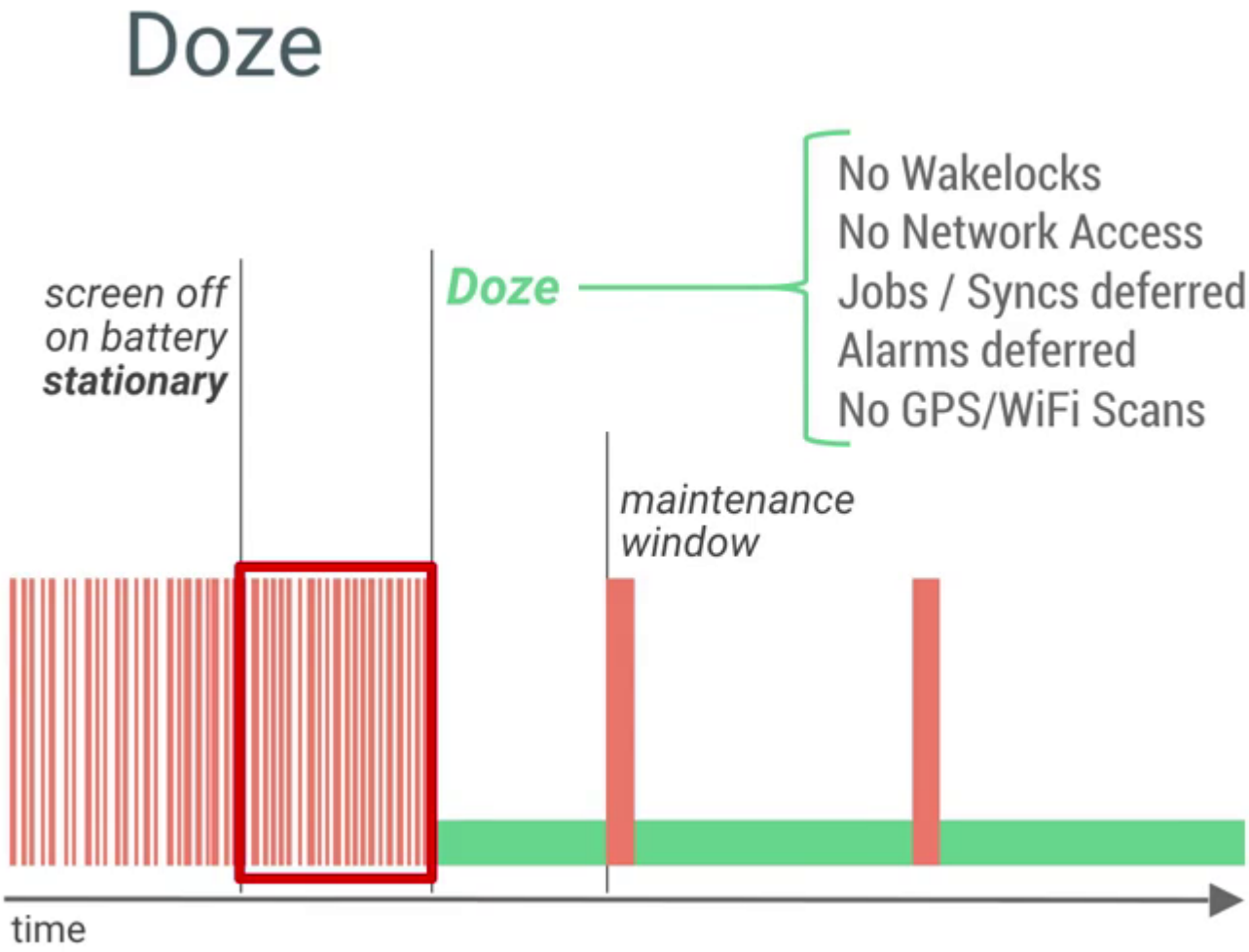
Dolch 2 online lesen: https://riptutorial.com/de/android/topic/3088/dolch-2

Kapitel 75: Doze-Modus

Bemerkungen

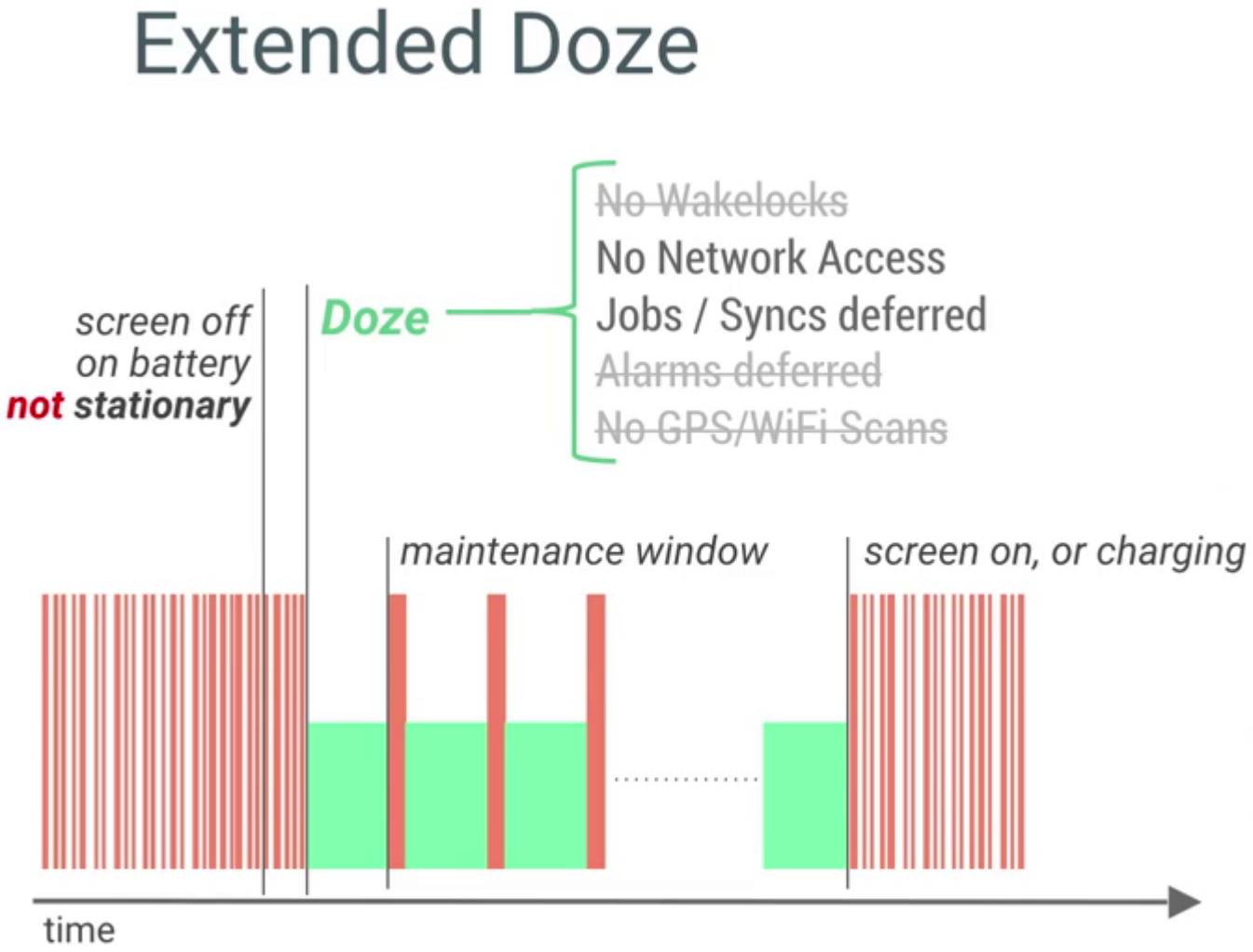
Der Doze-Modus ist ein Satz von Änderungen und Regeln, die das Telefon in den Ruhezustand versetzen, wenn es nicht verwendet wird.

Unter Android 6.0 Marshmallow: Der Doze-Modus wird aktiviert, nachdem der Bildschirm ausgeschaltet ist, das Gerät steht und der Akku leer ist.



Wie Sie im obigen Diagramm sehen können, erhält das Gerät bei Aktivierung des Doze-Modus keine Wakelocks, Netzwerkzugriff, Jobs / Synchronisierungen, Alarme, GPS- / WLAN-Scans.

Unter Android 7.0 Nougat: Stellen Sie sich vor, wenn sich Ihr Telefon in Ihrer Tasche befindet (der Bildschirm ist ausgeschaltet, der Akku läuft, aber es ist nicht stationär), möchten Sie vielleicht auch die Doze-Modus-Funktionen nutzen, oder? Deshalb hat Google den Extended Doze-Modus angekündigt: Er läuft, wenn der Bildschirm ausgeschaltet ist, aber nicht stationär.



Wie Sie in diesem Diagramm sehen können, sind nur Netzwerkzugriff und Jobs / Synchronisierungen deaktiviert. Beachten Sie, dass die erweiterte Doze den ersten Doze-Modus nicht ersetzt. Sie arbeiten zusammen, abhängig vom Telefonstatus (stationär oder nicht). Hier sind die Unterschiede:

	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

Entwickler sollten sich bewusst sein, dass:

- Doze behält möglicherweise temporären Wakelock- und Netzwerkzugriff für GCM-Nachrichten (Google Cloud Messaging) mit hoher Priorität bei (wenn der Benutzer eine sofortige Benachrichtigung benötigt).
- Vordergrunddienste (z. B. Musikwiedergabe) funktionieren weiterhin.

Weitere Informationen finden Sie hier: https://developer.android.com/training/monitoring-device-state/doze-standby.html

Examples

App vom Doze-Modus ausschließen

1. Telefoneinstellungen öffnen
2. Batterie öffnen
3. Menü öffnen und "Batterieoptimierung" auswählen
4. Wählen Sie im Dropdown-Menü "Alle Apps" aus.
5. Wählen Sie die App aus, die Sie auf die Positivliste setzen möchten
6. wähle "nicht optimieren"

Diese App wird jetzt unter nicht optimierten Apps angezeigt.

Eine App kann durch Aufrufen von `isIgnoringBatteryOptimizations()` prüfen, ob sie als Whitelist `isIgnoringBatteryOptimizations()`

Whitelisting einer Android-Anwendung programmgesteuert

Bei der Whitelisting-Funktion wird der Doze-Modus für Ihre App nicht deaktiviert. Sie können dies jedoch über Netzwerk- und Hold-Wake-Sperren tun.

Das programmgesteuerte Whitelisting einer Android-Anwendung kann wie folgt durchgeführt werden:

```
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);
}
```

Das Ergebnis des Startens der obigen Aktivität kann durch den folgenden Code überprüft werden:

```
@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
        }
    }
}
```

Doze-Modus online lesen: https://riptutorial.com/de/android/topic/4719/doze-modus

Kapitel 76: Drawables

Examples

Tönen Sie ein Zeichen

Ein Drawable kann mit einer bestimmten Farbe eingefärbt werden. Dies ist hilfreich, um verschiedene Designs in Ihrer Anwendung zu unterstützen und die Anzahl an zeichnungsfähigen Ressourcendateien zu reduzieren.

Verwenden von Framework-APIs für SDK 21+:

```
Drawable d = context.getDrawable(R.drawable.ic_launcher);
d.setTint(Color.WHITE);
```

Verwenden der android.support.v4-Bibliothek im SDK 4+:

```
//Load the untinted resource
final Drawable drawableRes = ContextCompat.getDrawable(context, R.drawable.ic_launcher);
//Wrap it with the compatibility library so it can be altered
Drawable tintedDrawable = DrawableCompat.wrap(drawableRes);
//Apply a coloured tint
DrawableCompat.setTint(tintedDrawable, Color.WHITE);
//At this point you may use the tintedDrawable just as you usually would
//(and drawableRes can be discarded)

//NOTE: If your original drawableRes was in use somewhere (i.e. it was the result of
//a call to a `getBackground()` method then at this point you still need to replace
//the background. setTint does *not* alter the instance that drawableRes points to,
//but instead creates a new drawable instance
```

Bitte beachten Sie nicht, dass sich `int color` **nicht** auf eine `int color` bezieht. Sie sind jedoch nicht auf die in der Klasse "Color" definierten Farben beschränkt. Wenn Sie in Ihrem XML eine Farbe definiert haben, die Sie verwenden möchten, müssen Sie zunächst dessen Wert ermitteln.

Sie können die Verwendung von `Color.WHITE` mithilfe der folgenden Methoden ersetzen

Beim Targeting älterer APIs:

```
getResources().getColor(R.color.your_color);
```

Oder zu neueren Zielen:

```
ContextCompat.getColor(context, R.color.your_color);
```

Ansicht mit abgerundeten Ecken erstellen

Erstellen Sie eine **zeichnungsfähige** Datei mit dem Namen **custom_rectangle.xml** in einem

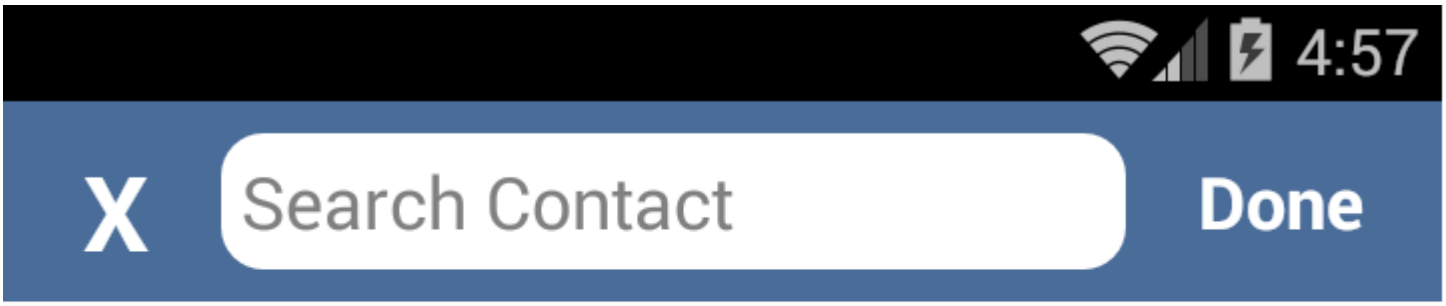
Zeichnungsordner :

```
<?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>
```

Wenden Sie jetzt den **Rechteckhintergrund** auf **Ansicht an** :

```
mView.setBackGround(R.drawlable.custom_rectangle);
```

Referenz-Screenshot:



Kreisförmige Ansicht

Für eine kreisförmige Ansicht (in diesem Fall `TextView`) erstellen **Sie** im **drawble-** Ordner eine drawble **round_view.xml** :

```
<?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>
```

Weisen Sie das Drawable der Ansicht zu:

```
    <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" />
```

Jetzt sollte es wie der orange Kreis aussehen:

Kundenspezifisch gezeichnet

Erweitern Sie Ihre Klasse mit Drawable und überschreiben Sie diese Methoden

```
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;
    }
}
```

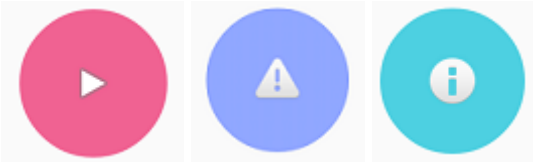
Deklarieren Sie eine ImageView in Ihrem Layout

```
<ImageView
   android:layout_width="80dp"
   android:id="@+id/imageView"
   android:layout_height="80dp" />
```

Stellen Sie Ihre benutzerdefinierte Zeichnung auf die ImageView ein

```
 IconDrawable iconDrawable=new
IconDrawable(ContextCompat.getDrawable(this,android.R.drawable.ic_media_play),ContextCompat.getColor(th

 imageView.setImageDrawable(iconDrawable);
```

Bildschirmfoto



Drawables online lesen: https://riptutorial.com/de/android/topic/4841/drawables

Kapitel 77: E-Mail-Bestätigung

Examples

Überprüfung der E-Mail-Adresse

Fügen Sie die folgende Methode hinzu, um zu prüfen, ob eine E-Mail-Adresse gültig ist oder nicht:

```
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();
}
```

Die obige Methode kann leicht überprüft werden, indem der Text eines `EditText` Widgets in einen `String`:

```
if(isValidEmailId(edtEmailId.getText().toString().trim())){
  Toast.makeText(getApplicationContext(), "Valid Email Address.", Toast.LENGTH_SHORT).show();
}else{
  Toast.makeText(getApplicationContext(), "InValid Email Address.",
Toast.LENGTH_SHORT).show();
}
```

Überprüfung der E-Mail-Adresse mit Mustern

```
  if (Patterns.EMAIL_ADDRESS.matcher(email).matches()){
       Log.i("EmailCheck","It is valid");
  }
```

E-Mail-Bestätigung online lesen: https://riptutorial.com/de/android/topic/5605/e-mail-bestatigung

Kapitel 78: Emulator

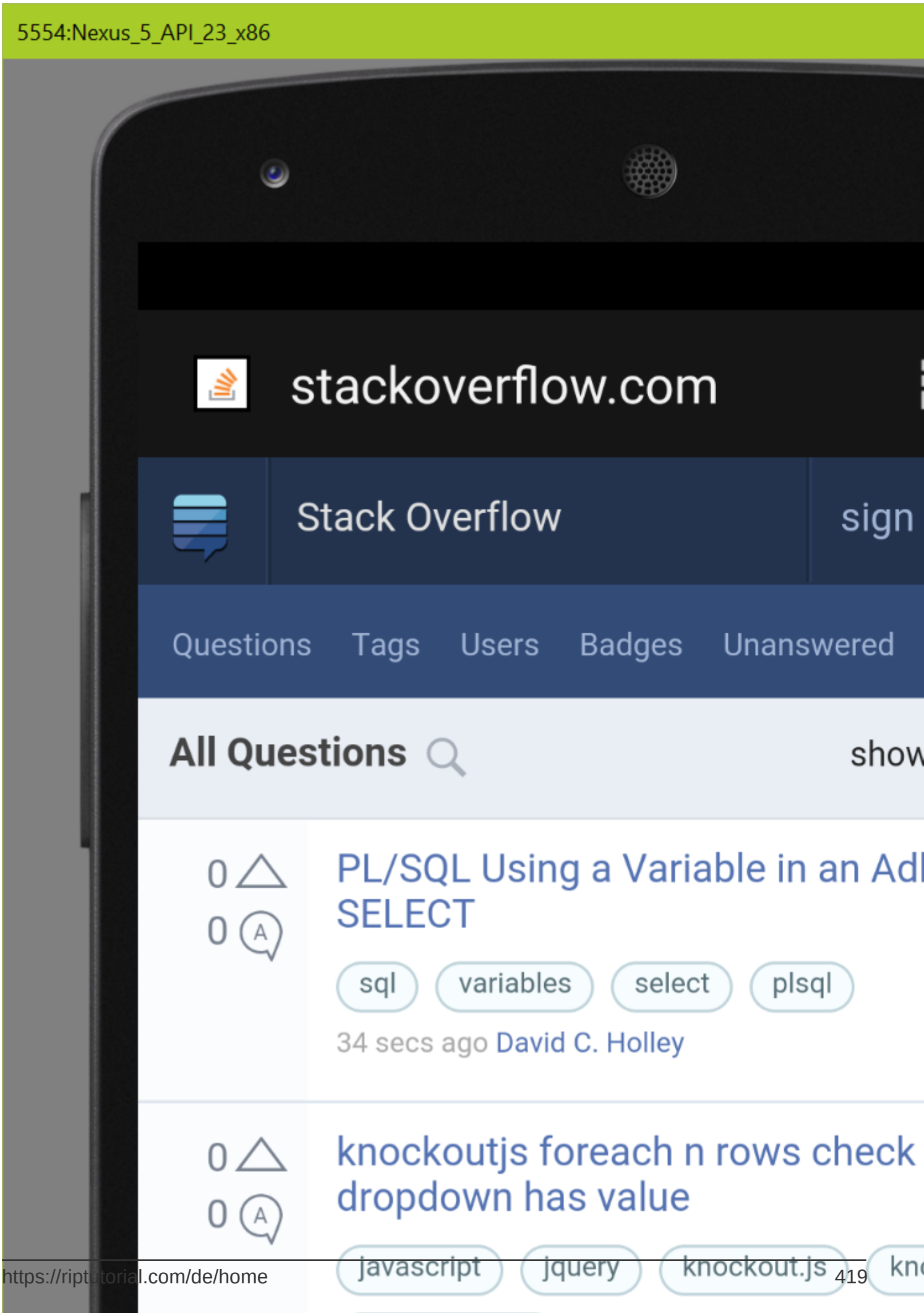
Bemerkungen

AVD steht für *Android Virtual Device*

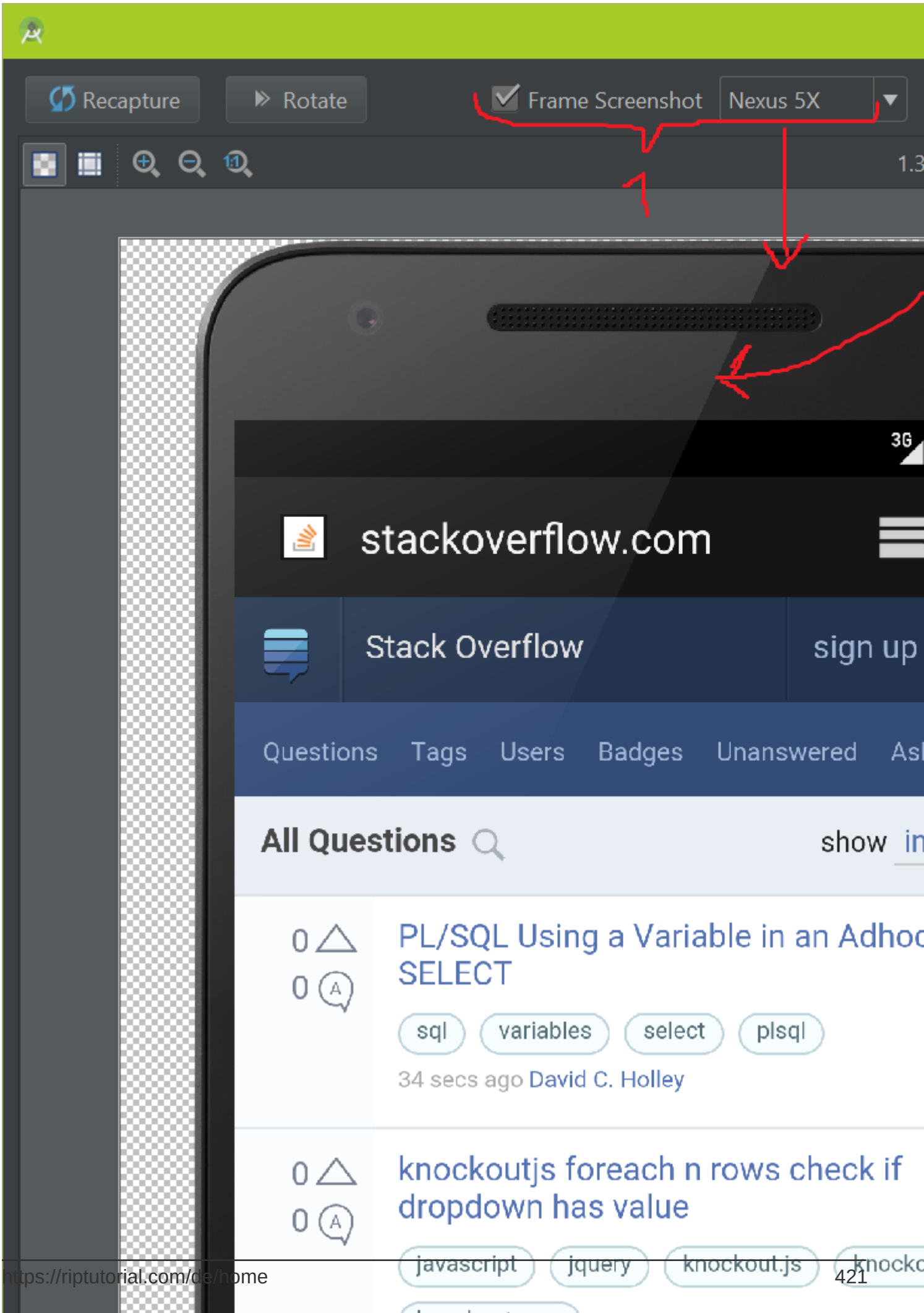
Examples

Screenshots machen

Wenn Sie einen Screenshot vom Android Emulator (2.0) aufnehmen möchten, drücken Sie einfach `Strg` + `S` oder klicken Sie auf das Kamerasymbol in der Seitenleiste:


5554:Nexus_5_API_23_x86
stackoverflow.com
Stack Overflow
sign
Questions
Tags
Users
Badges
Unanswered
All Questions
show
0
0
PL/SQL Using a Variable in an Adl
SELECT
sql
variables
select
plsql
34 secs ago David C. Holley
0
0
knockoutjs foreach n rows check
dropdown has value
javascript
jquery
knockout.js

2. Ein Schatten unter dem Geräterahmen.
3. Eine Bildschirmblendung über den Geräterahmen und den Screenshot.

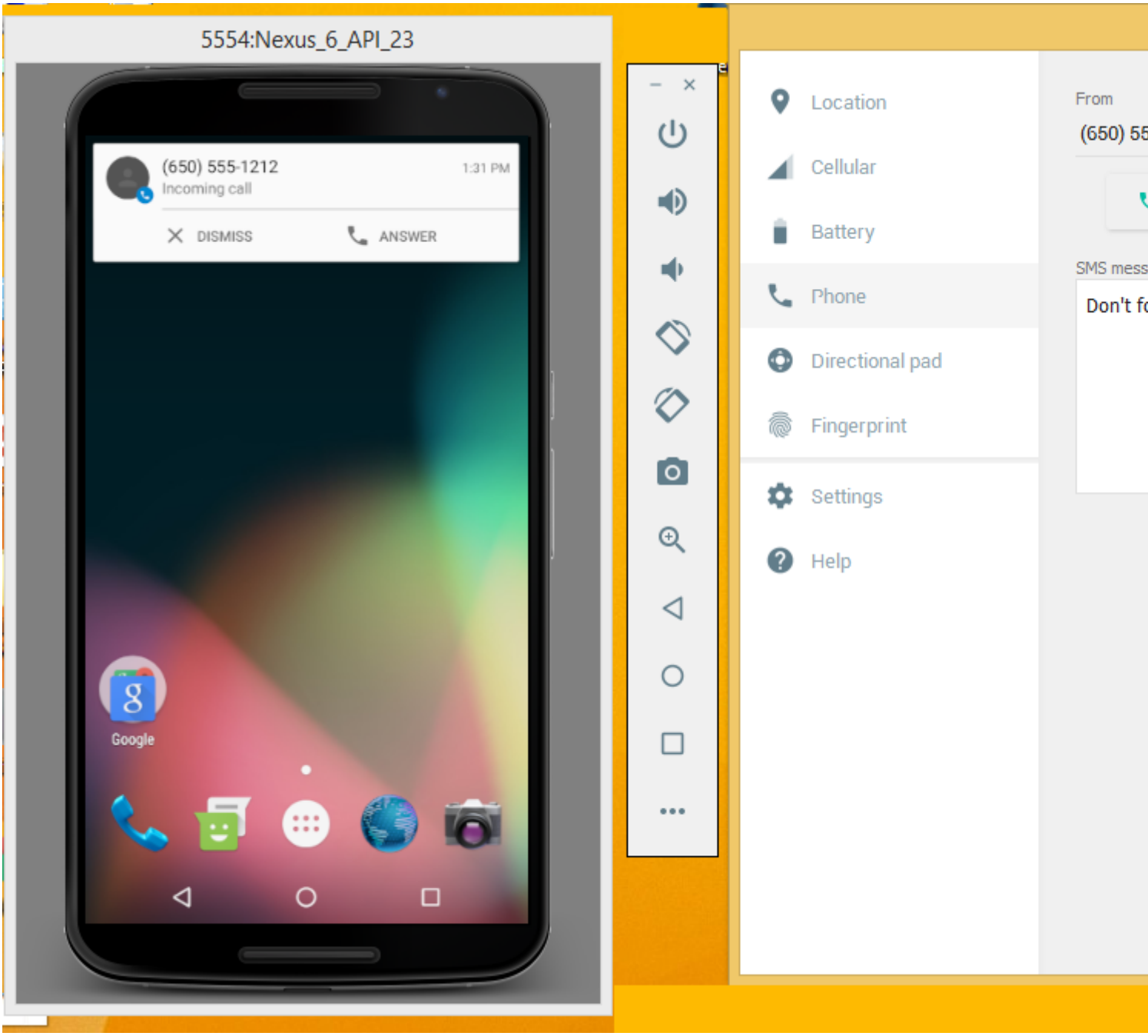

Recapture
Rotate
Frame Screenshot
Nexus 5X
3G
stackoverflow.com
Stack Overflow
sign up
Questions
Tags
Users
Badges
Unanswered
All Questions
show
0
0
PL/SQL Using a Variable in an Adhoc SELECT
sql
variables
select
plsql
34 secs ago David C. Holley
0
0
knockoutjs foreach n rows check if dropdown has value
javascript
jquery
knockout.js

die Befehlszeile öffnen.

Sie können AVD Manager auch von Android Studio aus über `Tools` > `Android` > `AVD Manager` aufrufen oder indem Sie in der Symbolleiste auf das Symbol `AVD Manager` klicken.

Anruf simulieren

Um einen Anruf zu simulieren, drücken Sie die durch drei Punkte gekennzeichnete Schaltfläche "Erweiterte Bedienelemente", wählen Sie "Telefon" und anschließend "Anrufen". Sie können die Telefonnummer auch optional ändern.



Fehler beim Starten des Emulators beheben

Stellen Sie zunächst sicher, dass Sie die " **Virtualisierung"** in Ihrem BIOS-Setup aktiviert haben.

Starten Sie den **Android SDK Manager** , wählen Sie **Extras** und dann **Intel Hardware Accelerated Execution Manager.** Warten Sie, bis der Download abgeschlossen ist. Wenn es immer noch nicht funktioniert, öffnen Sie den SDK-Ordner und führen Sie `/extras/intel/Hardware_Accelerated_Execution_Manager/IntelHAXM.exe` .

Befolgen Sie die Anweisungen auf dem Bildschirm, um die Installation abzuschließen.

Für OS X können Sie dies auch ohne Bildschirmanweisungen wie `/extras/intel/Hardware_Accelerated_Execution_Manager/HAXM\ installation` tun:
`/extras/intel/Hardware_Accelerated_Execution_Manager/HAXM\ installation`

> Wenn Ihre CPU VT-x oder SVM nicht unterstützt, können Sie keine x86-basierten Android-Images verwenden. Bitte verwenden Sie stattdessen ARM-basierte Bilder.

Vergewissern Sie sich nach Abschluss der Installation, dass der Virtualisierungstreiber ordnungsgemäß funktioniert, indem Sie ein Eingabeaufforderungsfenster öffnen und den folgenden Befehl `sc query intelhaxm` : `sc query intelhaxm`

So führen Sie einen x86-basierten Emulator mit VM-Beschleunigung aus: Wenn Sie den Emulator über die Befehlszeile `emulator -avd <avd_name>` , geben Sie einfach eine x86-basierte AVD an:
`emulator -avd <avd_name>`

Wenn Sie alle oben genannten Schritte korrekt ausführen, sollten Sie in der Lage sein, Ihren AVD mit HAXM normal zu sehen.

Emulator online lesen: https://riptutorial.com/de/android/topic/122/emulator

Kapitel 79: Erkennen Sie ein Shake-Ereignis in Android

Examples

Shake Detector im Android-Beispiel

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

Verwenden der seismischen Erschütterungserkennung

Seismic ist eine Android-Gerät zur Erkennung der Verwacklungserkennung von Square. Um es zu benutzen, hören Sie einfach die Shake-Ereignisse, die von ihm gesendet werden.

```
@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();
}
```

Um einen anderen Beschleunigungsschwellenwert zu definieren, verwenden Sie `sd.setSensitivity(sensitivity)` mit einer `sensitivity` von `SENSITIVITY_LIGHT` , `SENSITIVITY_MEDIUM` , `SENSITIVITY_HARD` oder einem anderen sinnvollen ganzzahligen Wert. Die angegebenen Standardwerte liegen zwischen *11* und *15* .

Installation

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

Erkennen Sie ein Shake-Ereignis in Android online lesen:
https://riptutorial.com/de/android/topic/4501/erkennen-sie-ein-shake-ereignis-in-android

Kapitel 80: Erste Schritte mit OpenGL ES 2.0+

Einführung

In diesem Thema wird die Einrichtung und Verwendung von **OpenGL ES 2.0+** unter Android beschrieben. OpenGL ES ist der Standard für 2D- und 3D-beschleunigte Grafiken in eingebetteten Systemen - einschließlich Konsolen, Smartphones, Geräten und Fahrzeugen.

Examples

Einrichten von GLSurfaceView und OpenGL ES 2.0+

Um OpenGL ES in Ihrer Anwendung verwenden zu können, müssen Sie dies dem Manifest hinzufügen:

```
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
```

Erstellen Sie Ihre erweiterte GLSurfaceView:

```
import static android.opengl.GLES20.*; // To use all OpenGL ES 2.0 methods and constants
statically

public class MyGLSurfaceView extends GLSurfaceView {

    public MyGLSurfaceView(Context context, AttributeSet attrs) {
        super(context, attrs);

        setEGLContextClientVersion(2); // OpenGL ES version 2.0
        setRenderer(new MyRenderer());
        setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
    }

    public final class MyRenderer implements GLSurfaceView.Renderer{
        public final void onSurfaceCreated(GL10 unused, EGLConfig config) {
            // Your OpenGL ES init methods
            glClearColor(1f, 0f, 0f, 1f);
        }
        public final void onSurfaceChanged(GL10 unused, int width, int height) {
            glViewport(0, 0, width, height);
        }

        public final void onDrawFrame(GL10 unused) {
            // Your OpenGL ES draw methods
            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        }
    }
}
```

Fügen Sie `MyGLSurfaceView` Ihrem Layout hinzu:

```
<com.example.app.MyGLSurfaceView
```

```
    android:id="@+id/gles_renderer"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Um eine neuere Version von OpenGL ES zu verwenden, ändern Sie einfach die Versionsnummer in Ihrem Manifest, im statischen Import und ändern Sie `setEGLContextClientVersion` .

GLSL-ES-Shader aus Asset-Datei zusammenstellen und verknüpfen

Der Ordner Assets ist der häufigste Speicherort für Ihre GLSL-ES-Shader-Dateien. Um sie in Ihrer OpenGL ES-Anwendung verwenden zu können, müssen Sie sie zunächst in eine Zeichenfolge laden. Diese Funktion erstellt eine Zeichenfolge aus der Asset-Datei:

```
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;
    }
}
```

Nun müssen Sie eine Funktion erstellen, die einen in einem Stich gespeicherten Shader kompiliert:

```
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;
}
```

Jetzt können Sie Ihre Shader laden, kompilieren und verknüpfen:

```
// 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);
}
```

Wenn keine Fehler auftreten, ist Ihr Shader-Programm einsatzbereit:

```
glUseProgram(shaderProgram);
```

Erste Schritte mit OpenGL ES 2.0+ online lesen:
https://riptutorial.com/de/android/topic/8662/erste-schritte-mit-opengl-es-2-0plus

Kapitel 81: Erstellen Sie benutzerdefinierte Android-ROMs

Examples

Machen Sie Ihre Maschine für den Bau bereit!

Bevor Sie etwas bauen können, müssen Sie Ihre Maschine für den Bau vorbereiten. Dafür müssen Sie viele Bibliotheken und Module installieren. Die am meisten empfohlene Linux-Distribution ist Ubuntu. Daher konzentriert sich dieses Beispiel auf die Installation alles, was auf Ubuntu benötigt wird.

Java installieren

`sudo apt-add-repository ppa:openjdk-r/ppa` Sie zunächst das folgende persönliche Paketarchiv (PPA) hinzu: `sudo apt-add-repository ppa:openjdk-r/ppa` .

Aktualisieren Sie dann die Quellen, indem Sie `sudo apt-get update` ausführen: `sudo apt-get update` .

Zusätzliche Abhängigkeiten installieren

Alle erforderlichen zusätzlichen Abhängigkeiten können mit dem folgenden Befehl installiert werden:

```
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
```

Das System für die Entwicklung vorbereiten

Nachdem nun alle Abhängigkeiten installiert sind, bereiten wir das System für die Entwicklung vor, indem Sie Folgendes ausführen:

```
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
```

Lassen Sie uns zum Schluss den Cache und das Repo mit den folgenden Befehlen einrichten:

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

Bitte beachten Sie: Wir können dieses Setup auch erreichen, indem Sie die automatisierten Skripts *ausführen* , die von Akhil Narang (*Akhilnarang*), einem der *Verwalter* von Resurrection Remix OS, erstellt wurden . Diese Skripte finden Sie auf GitHub .

Erstellen Sie benutzerdefinierte Android-ROMs online lesen:
https://riptutorial.com/de/android/topic/9212/erstellen-sie-benutzerdefinierte-android-roms

Kapitel 82: Erstellen Sie Ihre eigenen Bibliotheken für Android-Anwendungen

Examples

Bibliotheksprojekt erstellen

Um eine Bibliothek zu erstellen, sollten Sie `File -> New -> New Module -> Android Library`. Dadurch wird ein einfaches Bibliotheksprojekt erstellt.

Wenn dies erledigt ist, müssen Sie über ein Projekt verfügen, das folgendermaßen eingerichtet ist:

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

Ihre Datei " `settings.gradle` " muss Folgendes enthalten:

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

Ihr `[library root directory]` muss Folgendes enthalten:

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

Ihre "app" `build.gradle` -Datei muss Folgendes enthalten:

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

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.2"

    defaultConfig {
        minSdkVersion 14
        targetSdkVersion 23
```

```
    }
}
```

Damit sollte Ihr Projekt gut funktionieren!

Verwenden der Bibliothek im Projekt als Modul

Um die Bibliothek verwenden zu können, müssen Sie sie als Abhängigkeit in die folgende Zeile aufnehmen:

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

Erstellen Sie eine Bibliothek, die auf Jitpack.io verfügbar ist

Führen Sie die folgenden Schritte aus, um die Bibliothek zu erstellen:

1. Erstellen Sie ein GitHub-Konto.
2. Erstellen Sie ein Git-Repository mit Ihrem Bibliotheksprojekt.
3. Ändern Sie die `build.gradle` Datei Ihres Bibliotheksprojekts, indem Sie den folgenden Code hinzufügen:

   ```
   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
   }
   ```

   Stellen Sie sicher, dass Sie die oben genannten Änderungen in GitHub festschreiben / verschieben.

4. Erstellen Sie eine Version aus dem aktuellen Code auf Github.

5. Führen Sie `gradlew install` auf Ihrem Code aus.

6. Ihre Bibliothek ist jetzt in der folgenden Abhängigkeit verfügbar:

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

Erstellen Sie Ihre eigenen Bibliotheken für Android-Anwendungen online lesen: https://riptutorial.com/de/android/topic/4118/erstellen-sie-ihre-eigenen-bibliotheken-fur-android-anwendungen

Kapitel 83: Erstellen von Overlay-Fenstern (immer im Vordergrund)

Examples

Popup-Überlagerung

Um Ihre Ansicht auf jede Anwendung zu setzen, müssen Sie sie dem entsprechenden Fenstermanager zuordnen. Dazu benötigen Sie die Systemalarmberechtigung, die Sie durch Hinzufügen der folgenden Zeile zu Ihrer Manifestdatei anfordern können:

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

Hinweis: Wenn Ihre Anwendung zerstört wird, wird Ihre Ansicht aus dem Fenstermanager entfernt. Daher ist es besser, die Ansicht zu erstellen und sie durch einen Vordergrunddienst dem Fenstermanager zuzuweisen.

Ansicht dem WindowManager zuweisen

Sie können eine Window Manager-Instanz wie folgt abrufen:

```
WindowManager mWindowManager = (WindowManager)
mContext.getSystemService(Context.WINDOW_SERVICE);
```

Um die Position Ihrer Ansicht zu definieren, müssen Sie einige Layout-Parameter wie folgt erstellen:

```
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;
```

Nun können Sie Ihre Ansicht zusammen mit den erstellten Layoutparametern der Window Manager-Instanz wie folgt zuordnen:

```
mWindowManager.addView(yourView, mLayoutParams);
```

Voila! Ihre Ansicht wurde erfolgreich auf alle anderen Anwendungen platziert.

Hinweis: Ihre Ansicht wird nicht auf die Tastensperre gestellt.

Erteilen der SYSTEM_ALERT_WINDOW-Berechtigung für Android 6.0 und höher

Ab Android 6.0 muss diese Berechtigung dynamisch erteilt werden.

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

Fehler unter 6.0 unter der Berechtigung verweigert,

```
Caused by: android.view.WindowManager$BadTokenException: Unable to add window
android.view.ViewRootImpl$W@86fb55b -- permission denied for this window type
```

Lösung: -

Anfordern der Overlay-Berechtigung wie unten

```
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);
}
```

Überprüfen Sie das Ergebnis

```
@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...
        }
    }
}
```

Erstellen von Overlay-Fenstern (immer im Vordergrund) online lesen:
https://riptutorial.com/de/android/topic/6214/erstellen-von-overlay-fenstern--immer-im-vordergrund-

Kapitel 84: ExoPlayer

Examples

Fügen Sie den ExoPlayer zum Projekt hinzu

Über jCenter

Fügen Sie Folgendes in die build.gradle-Datei Ihres Projekts ein:

```
compile 'com.google.android.exoplayer:exoplayer:rX.X.X'
```

wobei rX.XX die bevorzugte Version ist. Die neueste Version finden Sie in den Releases des Projekts. Weitere Informationen finden Sie im Projekt auf Bintray .

ExoPlayer verwenden

Instantiieren Sie Ihren ExoPlayer:

```
exoPlayer = ExoPlayer.Factory.newInstance(RENDERER_COUNT, minBufferMs, minRebufferMs);
```

Um nur Audio abzuspielen, können Sie diese Werte verwenden:

```
RENDERER_COUNT = 1 //since you want to render simple audio
minBufferMs = 1000
minRebufferMs = 5000
```

Beide Pufferwerte können entsprechend Ihren Anforderungen angepasst werden.

Nun müssen Sie eine DataSource anlegen. Wenn Sie MP3 streamen möchten, können Sie die DefaultUriDataSource verwenden. Sie müssen den Kontext und einen UserAgent übergeben. Um es einfach zu halten, spielen Sie eine lokale Datei ab und übergeben Sie null als userAgent:

```
DataSource dataSource = new DefaultUriDataSource(context, null);
```

Dann erstellen Sie die sampleSource:

```
ExtractorSampleSource sampleSource = new ExtractorSampleSource(
                uri, dataSource, new Mp3Extractor(), RENDERER_COUNT, requestedBufferSize);
```

uri verweist auf Ihre Datei. Als Extractor können Sie einen einfachen Standard-Mp3Extractor verwenden, wenn Sie MP3 abspielen möchten. requestBufferSize kann entsprechend Ihren Anforderungen erneut angepasst werden. Verwenden Sie zum Beispiel 5000.

Jetzt können Sie Ihren Audiospur-Renderer mit der Beispielquelle wie folgt erstellen:

```
MediaCodecAudioTrackRenderer audioRenderer = new MediaCodecAudioTrackRenderer(sampleSource);
```

Rufen Sie schließlich die Vorbereitungen für Ihre ExoPlayer-Instanz auf:

```
exoPlayer.prepare(audioRenderer);
```

Wiedergabeanruf starten:

```
exoPlayer.setPlayWhenReady(true);
```

Hauptschritte zum Abspielen von Video und Audio mit den standardmäßigen TrackRenderer-Implementierungen

```
// 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!
```

ExoPlayer online lesen: https://riptutorial.com/de/android/topic/6248/exoplayer

Kapitel 85: Facebook SDK für Android

Syntax

- **newInstance** : Zum Erstellen einer einzelnen Instanz der Facebook-Helper-Klasse.
- **loginUser** : Zum Anmelden des Benutzers.
- **signOut** : Zum **Abmelden des** Benutzers.
- **getCallbackManager** : Um Rückruf für Facebook zu erhalten.
- **getLoginCallback** : Um einen Rückruf für die **Anmeldung** zu erhalten.
- **getKeyHash** : Um Facebook Key Hash zu generieren.

Parameter

Parameter	Einzelheiten
ETIKETT	Ein String, der während der Protokollierung verwendet wird
FacebookSignInHelper	Ein statischer Hinweis auf Facebook-Helfer
CallbackManager	Ein Rückruf für Facebook-Operationen
Aktivität	Ein Kontext
PERMISSION_LOGIN	Ein Array, das alle erforderlichen Berechtigungen von Facebook für die Anmeldung enthält
loginCallback	Ein Rückruf für Facebook-Login

Examples

Wie füge ich Facebook-Login in Android hinzu?

Fügen Sie Ihrem `build.gradle` folgende Abhängigkeiten `build.gradle`

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

Fügen Sie Ihrem Hilfsprogramm folgende Hilfsklasse hinzu:

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

Fügen Sie den folgenden Code in Ihre Aktivität ein:

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

Fügen Sie folgenden Code zu Ihrem `OnActivityResult` :

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

Festlegen von Berechtigungen für den Zugriff auf Daten aus dem Facebook-Profil

Wenn Sie die Details eines Facebook-Profils eines Benutzers abrufen möchten, müssen Sie Berechtigungen für dieses Profil festlegen:

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

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

Sie können weitere Berechtigungen hinzufügen, z. B. Freunde, Beiträge, Fotos usw. Wählen Sie einfach die richtige Berechtigung aus und fügen Sie sie der obigen Liste hinzu.

Hinweis: Sie müssen keine expliziten Berechtigungen für den Zugriff auf das öffentliche Profil festlegen (Vorname, Nachname, ID, Geschlecht usw.).

Erstellen Sie Ihren eigenen benutzerdefinierten Button für das Facebook-Login

Nachdem Sie das Facebook-Login / die Anmeldung zum ersten Mal hinzugefügt haben, sieht die Schaltfläche etwa so aus:

In den meisten Fällen passt es nicht zu den Design-Spezifikationen Ihrer App. Und so können Sie es anpassen:

```
<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>
```

`com.facebook.login.widget.LoginButton` einfach den ursprünglichen `com.facebook.login.widget.LoginButton` in ein `FrameLayout` und machen Sie seine Sichtbarkeit verloren.

`FrameLayout` Sie anschließend Ihre benutzerdefinierte Schaltfläche in demselben `FrameLayout` . Ich habe einige Beispielspezifikationen hinzugefügt. Sie können jederzeit einen eigenen Hintergrund

für die Facebook-Schaltfläche erstellen und als Hintergrund für die Schaltfläche festlegen.

Als letztes konvertieren Sie einfach den Klick auf meinen benutzerdefinierten Button in einen Klick auf den Facecbook-Button:

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

Großartig! Nun sieht der Button so aus:

Ein minimalistischer Leitfaden für die Facebook-Anmeldung / Anmeldung

1. Sie müssen die Voraussetzungen festlegen.
2. Fügen Sie die Facebook-Aktivität zur Datei *AndroidManifest.xml hinzu* :

   ```
   <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. Fügen Sie der XML-Layoutdatei die Anmeldeschaltfläche hinzu:

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

4. Jetzt haben Sie den Facebook-Button. Wenn der Benutzer darauf klickt, wird der Facebook-Anmeldedialog oben auf dem Bildschirm der App angezeigt. Hier kann der Benutzer seine Anmeldeinformationen eingeben und auf die Schaltfläche *Anmelden* klicken. Wenn die Anmeldeinformationen korrekt sind, erteilt das Dialogfeld die entsprechenden Berechtigungen und ein Rückruf wird an Ihre ursprüngliche Aktivität gesendet, die die Schaltfläche enthält. Der folgende Code zeigt, wie Sie diesen Rückruf erhalten können:

   ```
   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.
    }
});
```

Abmeldung von Facebook

Facebook SDK 4.0 oder höher, so loggen wir uns aus:

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

Bei Versionen vor 4.0 wird das Abmelden durch das explizite Löschen des Zugriffstokens aufgehoben:

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

Facebook SDK für Android online lesen: https://riptutorial.com/de/android/topic/3919/facebook-sdk-fur-android

Kapitel 86: Faden

Examples

Thread-Beispiel mit seiner Beschreibung

Beim Start einer Anwendung wird zunächst der Haupt-Thread ausgeführt. Dieser Haupt-Thread behandelt das gesamte Anwendungskonzept der Benutzeroberfläche. Wenn wir die Task, in der wir die Benutzeroberfläche nicht benötigen, lange ausführen möchten, verwenden wir den Thread, um diese Task im Hintergrund auszuführen.

Hier ist das Beispiel von Thread, das Schlag beschreibt:

```
new Thread(new Runnable() {
    public void run() {
        for(int i = 1; i < 5;i++) {
            System.out.println(i);
        }
    }
}).start();
```

Wir können Thread erstellen, indem wir das Objekt von Thread erstellen, das die `Thread.run()` Methode zum Ausführen der thread.Here hat. Die `run()` Methode wird von der `start()` Methode aufgerufen.

Wir können die mehreren Threads auch unabhängig voneinander ausführen, was als MultiThreading bezeichnet wird. Dieser Thread verfügt auch über die Funktionalität des Ruhemodus, durch den der derzeit ausgeführte Thread für die angegebene Zeitspanne in den Ruhezustand versetzt wird (die Ausführung vorübergehend abbrechen). Aber der Schlaf wirft die InterruptedException. Also müssen wir damit umgehen, indem wir try / catch verwenden.

```
try{Thread.sleep(500);}catch(InterruptedException e){System.out.println(e);}
```

Aktualisieren der Benutzeroberfläche aus einem Hintergrund-Thread

Es ist üblich, einen Hintergrund-Thread für Netzwerkvorgänge oder für lange laufende Aufgaben zu verwenden und dann die Benutzeroberfläche bei Bedarf mit den Ergebnissen zu aktualisieren.

Dies stellt ein Problem dar, da nur der Haupt-Thread die Benutzeroberfläche aktualisieren kann.

Die Lösung ist die Verwendung der `runOnUiThread()` -Methode, da Sie damit die Codeausführung des UI-Threads von einem Hintergrund-Thread aus initiieren können.

In diesem einfachen Beispiel wird beim `runOnUiThread()` der Aktivität ein Thread gestartet, der ausgeführt wird, bis die magische Zahl `42` zufällig generiert wird. Anschließend wird die `runOnUiThread()` mit der `runOnUiThread()` Methode aktualisiert, sobald diese Bedingung erfüllt ist.

```
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();
    }
}
```

Faden online lesen: https://riptutorial.com/de/android/topic/7131/faden

Kapitel 87: Farbe

Einführung

Eine Farbe ist eines der vier zum Zeichnen erforderlichen Objekte, zusammen mit einem Canvas (für Zeichnungsaufrufe), einer Bitmap (enthält die Pixel) und einem Zeichnungsgrundelement (Rect, Path, Bitmap ...)

Examples

Einen Anstrich erstellen

Sie können einen neuen Anstrich mit einem dieser drei Konstruktoren erstellen:

- `new Paint()` Mit Standardeinstellungen erstellen
- `new Paint(int flags)` Mit Flags erstellen
- `new Paint(Paint from)` Kopiert die Einstellungen von einer anderen Farbe

Im Allgemeinen wird empfohlen, niemals ein Paint-Objekt oder ein anderes Objekt in onDraw () zu erstellen, da dies zu Leistungsproblemen führen kann. (Android Studio wird Sie wahrscheinlich warnen.) Machen Sie es stattdessen global und initialisieren Sie es in Ihrem Klassenkonstruktor wie folgt:

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

Einrichten von Paint für Text

Textzeichnungseinstellungen

- `setTypeface(Typeface typeface)` die Schriftart fest. Siehe Schriftbild
- `setTextSize(int size)` die Schriftgröße in Pixel fest.
- `setColor(int color)` die Farbe der Zeichenfarbe einschließlich der `setColor(int color)` . Sie

können auch `setARGB(int a, int r, int g, int b` und `setAlpha(int alpha)`

- `setLetterSpacing(float size)` den Abstand zwischen den Zeichen in ems fest. Der Standardwert ist 0, ein negativer Wert wird den Text enger machen, während ein positiver den Text erweitert.
- `setTextAlign(Paint.Align align)` Textausrichtung relativ zum Ursprung fest. `Paint.Align.LEFT` zeichnet es rechts vom Ursprung, `RIGHT` zeichnet es nach links und `CENTER` zeichnet es zentriert auf den Ursprung (horizontal).
- `setTextSkewX(float skewX)` Dies kann als gefälschte Kursivschrift betrachtet werden. SkewX steht für den horizontalen Versatz des unteren Texts. (Verwenden Sie -0,25 für Kursivschrift)
- `setStyle(Paint.Style style)` Füllen Sie den Text `FILL` , den Stroke-Text `STROKE` oder beide `FILL_AND_STROKE`

Beachten Sie, dass Sie `TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, size, getResources().getDisplayMetrics())` zum Konvertieren von SP oder DP in Pixel verwenden können.

Text messen

- `float width = paint.measureText(String text)` die Breite des Textes
- `float height = paint.ascent()` Messen Sie die Höhe des Texts
- `paint.getTextBounds(String text, int start, int end, Rect bounds` Speichert die `paint.getTextBounds(String text, int start, int end, Rect bounds` . Sie haben das Rect zugewiesen, es darf nicht null sein:

```
String text = "Hello world!";
Rect bounds = new Rect();
paint.getTextBounds(text, 0, text.length(), bounds);
```

Es gibt andere Messmethoden, diese drei sollten jedoch für die meisten Zwecke geeignet sein.

Einrichten von Paint zum Zeichnen von Formen

- `setStyle(Paint.Style style)` Gefüllte Form `FILL` , `STROKE` oder beide `FILL_AND_STROKE`
- `setColor(int color)` die Farbe der Zeichenfarbe fest. Sie können auch `setARGB(int a, int r, int g, int b` und `setAlpha(int alpha)`
- `setStrokeCap(Paint.Cap cap)` `ROUND` , `SQUARE` oder `BUTT` (keine). Siehe dies .
- `setStrokeJoin(Paint.Join join)` Setzt Linienverbindungen entweder `MITER` (pointy), `ROUND` oder `BEVEL` . Sehen Sie das hier .
- `setStrokeMiter(float miter)` Join-Grenze fest. Dies kann verhindern, dass die Gehrungsverknüpfung unbegrenzt fortfährt und nach x Pixel in eine Abschrägungsverknüpfung umgewandelt wird. Sehen Sie das hier .
- `setStrokeWidth(float width)` Strichbreite ein. `0` wird unabhängig von der Canvas-Matrix im Hairline-Modus gezeichnet. (immer 1 Pixel)

Flags setzen

Sie können die folgenden Flags im Konstruktor oder mit `setFlags(int flags)`

- `Paint.ANTI_ALIAS_FLAG` Antialiasing `Paint.ANTI_ALIAS_FLAG` , glättet die Zeichnung.
- `Paint.DITHER_FLAG` Dithering `Paint.DITHER_FLAG` . Wenn die Farbgenauigkeit höher ist als die des Geräts, wird dies passieren .
- `Paint.EMBEDDED_BITMAP_TEXT_FLAG` Ermöglicht die Verwendung von Bitmap-Schriftarten.
- `Paint.FAKE_BOLD_TEXT_FLAG` zeichnet Text mit einem gefälschten `Paint.FAKE_BOLD_TEXT_FLAG` kann anstelle einer Fettschrift verwendet werden. Einige Schriftarten sind fett dargestellt, fake fett nicht
- `Paint.FILTER_BITMAP_FLAG` die Umwandlung von Bitmaps.
- `Paint.HINTING_OFF` , `Paint.HINTING_ON` Schaltet die Schriftarthinweise um, siehe dies
- `Paint.LINEAR_TEXT_FLAG` Deaktiviert die Schriftskalierung. Zeichnen wird stattdessen skaliert
- `Paint.SUBPIXEL_TEXT_FLAG` Text wird mit Subpixel-Genauigkeit berechnet.
- `Paint.STRIKE_THRU_TEXT_FLAG` gezeichnete Text wird markiert
- `Paint.UNDERLINE_TEXT_FLAG` gezeichnete Text wird unterstrichen

Sie können eine Flagge hinzufügen und Flaggen wie folgt entfernen:

```
Paint paint = new Paint();
paint.setFlags(paint.getFlags() | Paint.FLAG);   // Add flag
paint.setFlags(paint.getFlags() & ~Paint.FLAG);  // Remove flag
```

Wenn Sie versuchen, ein Flag zu entfernen, das nicht vorhanden ist, oder das Hinzufügen eines Flag, das bereits vorhanden ist, ändert dies nichts. Beachten Sie auch, dass die meisten Flags auch mit `set<Flag>(boolean enabled)` , beispielsweise `setAntialias(true)` .

Sie können `paint.reset()` , um die Farbe auf die Standardeinstellungen zurückzusetzen. Das einzige Standardflag ist `EMBEDDED_BITMAP_TEXT_FLAG` . Es wird auch eingestellt, wenn Sie einen `new Paint(0)`

Farbe online lesen: https://riptutorial.com/de/android/topic/9141/farbe

Kapitel 88: Farben

Examples

Farbmanipulation

Um Farben zu ändern, ändern wir die argb-Werte (Alpha, Rot, Grün und Blau) einer Farbe.

Extrahieren Sie zunächst RGB-Werte aus Ihrer Farbe.

```
int yourColor = Color.parse("#ae1f67");

int red = Color.red(yourColor);
int green = Color.green(yourColor);
int blue = Color.blue(yourColor);
```

Jetzt können Sie die Werte für Rot, Grün und Blau verringern oder erhöhen und diese wieder zu einer Farbe kombinieren:

```
int newColor = Color.rgb(red, green, blue);
```

Wenn Sie etwas Alpha hinzufügen möchten, können Sie es beim Erstellen der Farbe hinzufügen:

```
int newColor = Color.argb(alpha, red, green, blue);
```

Alpha- und RGB-Werte sollten im Bereich [0-225] liegen.

Farben online lesen: https://riptutorial.com/de/android/topic/4986/farben

Kapitel 89: Fastjson

Einführung

Fastjson ist eine Java-Bibliothek, mit der Java-Objekte in ihre JSON-Darstellung konvertiert werden können. Es kann auch verwendet werden, um eine JSON-Zeichenfolge in ein entsprechendes Java-Objekt zu konvertieren.

Fastjson-Funktionen:

Bieten Sie die beste Leistung für Server und Android-Client

Stellen Sie einfache `toJSONString()` und `parseObject()` Methoden `parseObject()` , um Java-Objekte in JSON und umgekehrt zu konvertieren

Zulassen, dass vorhandene, nicht veränderbare Objekte in und aus JSON konvertiert werden

Umfassende Unterstützung von Java Generics

Syntax

- Objektanalyse (String-Text)
- JSONObject parseObject (Zeichenfolgentext)
- T parseObject (String-Text, Klasse <T> clazz)
- JSONArray parseArray (String-Text)
- <T> Liste <T> parseArray (String-Text, Klasse <T> clazz)
- String toJSONString (Objektobjekt)
- String toJSONString (Objektobjekt, boolean prettyFormat)
- Objekt toJSON (Objekt javaObject)

Examples

Parsen von JSON mit Fastjson

Sie können ein Beispiel in der Fastjson-Bibliothek betrachten

Kodieren

```
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);
```

Ausgabe

```
{"id":0,"name":"admin","users":[{"id":2,"name":"guest"},{"id":3,"name":"root"}]}
```

Dekodieren

```
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;
    }
}
```

Konvertieren Sie die Daten des Typs Map in JSON String

Code

```
    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);
```

Ausgabe

```
{1:"No.1",2:"No.2",3:[{"id":2,"name":"Liu"},{"id":3,"name":"Yue"}]}
```

Fastjson online lesen: https://riptutorial.com/de/android/topic/10865/fastjson

Kapitel 90: FileIO mit Android

Einführung

Das Lesen und Schreiben von Dateien in Android unterscheidet sich nicht vom Lesen und Schreiben von Dateien im Standard-Java. `java.io` Paket kann verwendet werden. Es gibt jedoch einige spezifische Informationen zu den Ordnern, in die Sie schreiben dürfen, Berechtigungen allgemein und MTP-Problemumgehungen.

Bemerkungen

Android bietet die Möglichkeit, die Datei für mehrere Anwendungen freizugeben, wie hier dokumentiert. Dies ist nicht erforderlich, wenn nur eine App die Datei erstellt und verwendet.

Android bietet alternative Speicheroptionen wie freigegebene und private Einstellungen, gespeicherte Bundles und integrierte Datenbank. In einigen Fällen sind sie die bessere Wahl, als nur einfache Dateien zu verwenden.

Für Android-Aktivitäten gibt es einige spezifische Methoden, die wie ein Ersatz der Java-Standard-File-IO-Methoden aussehen. Zum Beispiel, anstatt für `File.delete()` Sie können anrufen `Context.deleteFile()` , und anstelle der Anwendung `File.listFiles()` rekursiv können Sie anrufen `Context.fileList()` die Liste aller Ihrer Anwendung bestimmte Dateien mit etwas weniger zu bekommen Code. Sie bieten jedoch keine zusätzlichen Funktionen über das Standardpaket `java.io` hinaus.

Examples

Arbeitsordner erhalten

Sie erhalten Ihren Arbeitsordner, indem Sie die Methode `getFilesDir()` für Ihre Aktivität aufrufen (Activity ist die zentrale Klasse in Ihrer Anwendung, die von Context erbt. Siehe hier). Lesen ist nicht anders. Nur Ihre Anwendung hat Zugriff auf diesen Ordner.

Ihre Aktivität könnte zum Beispiel den folgenden Code enthalten:

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

Rohes Array von Bytes schreiben

```
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()
```

Es gibt nichts Android-spezifisches mit diesem Code. Wenn Sie häufig kleine Werte schreiben, verwenden Sie BufferedOutputStream , um den Verschleiß der geräteinternen SSD zu reduzieren.

Objekt serialisieren

Die alte, gute Java-Objektserialisierung steht Ihnen in Android zur Verfügung. Sie können serialisierbare Klassen definieren wie:

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

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

und schreibe dann in den ObjectOutputStream:

```
File myFile = new File(getFilesDir(), "myObjects.bin");
FileOutputStream out = new FileOutputStream(myFile);
ObjectOutputStream oout = new ObjectOutputStream(new BufferedOutputStream(out));

oout.writeObject(new Circle(10, "One"));
oout.writeObject(new Circle(12, "Two"));

oout.close()
```

Die Java-Objektserialisierung kann entweder perfekt oder wirklich schlecht sein, je nachdem, was Sie damit machen möchten - außerhalb des Rahmens dieses Tutorials und manchmal auf Meinungen beruhend. Lesen Sie zuerst die Versionsverwaltung, wenn Sie sie verwenden möchten.

Auf externen Speicher schreiben (SD-Karte)

Sie können auch von einer auf vielen Android-Geräten vorhandenen Speicherkarte (SD-Karte) lesen und darauf schreiben. Auf Dateien an diesem Speicherort kann von anderen Programmen zugegriffen werden, auch direkt vom Benutzer, nachdem das Gerät über ein USB-Kabel an den PC angeschlossen und das MTP-Protokoll aktiviert wurde.

Das Auffinden der SD-Karte ist etwas problematischer. Die Environment- Klasse enthält statische Methoden, um "externe Verzeichnisse" zu erhalten, die sich normalerweise auf der SD-Karte befinden sollten. Außerdem wird angegeben, ob die SD-Karte überhaupt vorhanden ist und beschreibbar ist. Diese Frage enthält wertvolle Antworten, um sicherzustellen, dass der richtige Ort gefunden wird.

Für den Zugriff auf externen Speicher sind Berechtigungen in Ihrem Android-Manifest erforderlich:

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

Bei älteren Versionsberechtigungen von Android reicht es aus, diese Berechtigungen in Manifest zu setzen (der Benutzer muss während der Installation zustimmen). Ab Android 6.0 fordert Android den Benutzer jedoch beim ersten Zugriff zur Genehmigung auf, und Sie müssen diesen neuen Ansatz unterstützen. Ansonsten wird der Zugriff unabhängig von Ihrem Manifest verweigert.

In Android 6.0 müssen Sie zunächst die Erlaubnis prüfen und dann, falls nicht erteilt, diese anfordern. Die Codebeispiele finden Sie in dieser SO-Frage .

Problem mit "unsichtbaren MTP-Dateien" behoben.

Wenn Sie Dateien zum Exportieren über ein USB-Kabel an den Desktop mithilfe des MTP-Protokolls erstellen, besteht das Problem, dass neu erstellte Dateien nicht sofort im Dateiexplorer des angeschlossenen Desktop-PCs angezeigt werden. Um neue Dateien sichtbar zu machen, müssen Sie MediaScannerConnection aufrufen :

```
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)));
```

Dieser MediaScannerConnection-Aufrufcode funktioniert nur für Dateien, nicht für Verzeichnisse. Das Problem wird in diesem Android-Fehlerbericht beschrieben . Dies kann für einige Versionen in der Zukunft oder auf einigen Geräten behoben sein.

Mit großen Dateien arbeiten

Kleine Dateien werden in Sekundenbruchteilen verarbeitet und Sie können sie anstelle des Codes lesen und schreiben, wo Sie dies benötigen. Wenn die Datei jedoch größer oder anderweitig langsamer verarbeitet werden muss, müssen Sie möglicherweise AsyncTask in Android verwenden, um mit der Datei im Hintergrund zu arbeiten:

```
    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
        }
    }
}
```

und dann

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

Diese SO-Frage enthält das vollständige Beispiel zum Erstellen und Aufrufen der AsyncTask. Lesen Sie auch die Frage zur Fehlerbehandlung , wie IOExceptions und andere Fehler behandelt werden.

FileIO mit Android online lesen: https://riptutorial.com/de/android/topic/8689/fileio-mit-android

Kapitel 91: FileProvider

Examples

Freigeben einer Datei

In diesem Beispiel erfahren Sie, wie Sie eine Datei für andere Apps freigeben. Wir verwenden in diesem Beispiel eine PDF-Datei, obwohl der Code auch mit jedem anderen Format funktioniert.

Die Roadmap:

Geben Sie die Verzeichnisse an, in denen sich die Dateien befinden, die Sie freigeben möchten

Um Dateien gemeinsam zu nutzen, verwenden wir einen FileProvider, eine Klasse, die eine sichere Dateifreigabe zwischen Apps ermöglicht. Ein FileProvider kann Dateien nur in vordefinierten Verzeichnissen freigeben. Definieren Sie diese also.

1. Erstellen Sie eine neue XML-Datei, die die Pfade enthält, z. B. *res / xml / filepaths.xml*
2. Fügen Sie die Pfade hinzu

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <files-path name="pdf_folder" path="documents/"/>
</paths>
```

Definieren Sie einen FileProvider und verknüpfen Sie ihn mit den Dateipfaden

Dies geschieht im Manifest:

```
<manifest>
    ...
    <application>
        ...
        <provider
            android:name="android.support.v4.context.FileProvider"
            android:authorities="com.mydomain.fileprovider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
```

```
                android:resource="@xml/filepaths" />
        </provider>
        ...
    </application>
    ...
</manifest>
```

Generieren Sie den URI für die Datei

Um die Datei gemeinsam zu nutzen, müssen wir einen Bezeichner für die Datei angeben. Dies geschieht mithilfe einer URI (Uniform Resource Identifier).

```
// We assume the file we want to load is in the documents/ subdirectory
// of the internal storage
File documentsPath = new File(Context.getFilesDir(), "documents");
File file = new File(documentsPath, "sample.pdf");
// This can also in one line of course:
// File file = new File(Context.getFilesDir(), "documents/sample.pdf");

Uri uri = FileProvider.getUriForFile(getContext(), "com.mydomain.fileprovider", file);
```

Wie Sie im Code sehen können, erstellen wir zunächst eine neue Dateiklasse, die die Datei darstellt. Um eine URI zu erhalten, bitten wir FileProvider, uns eine zu besorgen. Das zweite Argument ist wichtig: Es übergibt die Berechtigung eines FileProviders. Sie muss der Berechtigung des im Manifest definierten FileProviders entsprechen.

Teilen Sie die Datei mit anderen Apps

Wir verwenden ShareCompat, um die Datei mit anderen Apps zu teilen:

```
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);
```

Ein Wähler ist ein Menü, aus dem der Benutzer auswählen kann, mit welcher App er die Datei freigeben möchte. Das Flag Intent.FLAG_GRANT_READ_URI_PERMISSION wird benötigt, um dem URI temporären Lesezugriff zu gewähren.

FileProvider online lesen: https://riptutorial.com/de/android/topic/6266/fileprovider

Kapitel 92: Fingerprint-API in Android

Bemerkungen

siehe auch

Github-Beispielprojekt

Android-Entwicklerblogspot

Android-Entwicklerseite

Examples

Hinzufügen des Fingerabdruckscanners in der Android-Anwendung

Android unterstützt Fingerprint-API von Android 6.0 (Marshmallow) SDK 23

> Um diese Funktion in Ihrer App zu verwenden, fügen Sie zunächst die Berechtigung USE_FINGERPRINT in Ihr Manifest ein.

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

Hier das Verfahren, das zu befolgen ist

> Zunächst müssen Sie im Android Key Store einen symmetrischen Schlüssel mit KeyGenerator erstellen, der nur verwendet werden kann, nachdem sich der Benutzer mit Fingerabdruck authentifiziert hat und einen KeyGenParameterSpec übergeben hat.

```
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();
```

> Durch Festlegen von KeyGenParameterSpec.Builder.setUserAuthenticationRequired auf true können Sie die Verwendung des Schlüssels nur nach der Authentifizierung durch den Benutzer zulassen, auch wenn er mit dem Fingerabdruck des Benutzers authentifiziert wird.

```
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);
```

Beginnen Sie dann mit dem Abhören eines Fingerabdrucks auf dem Fingerabdrucksensor, indem Sie FingerprintManager.authenticate mit einem Cipher aufrufen, der mit dem erstellten symmetrischen Schlüssel initialisiert wurde. Alternativ können Sie auf das serverseitige verifizierte Kennwort als Authentifikator zurückgreifen.

Erstellen und initialisieren Sie den `FingerprintManger` von `fingerprintManger.class`

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

Verwenden Sie zur Authentifizierung die `FingerprintManger` API und erstellen Sie eine Unterklasse mit

`FingerprintManager.AuthenticationCallback` und überschreibt die Methoden

```
onAuthenticationError
onAuthenticationHelp
onAuthenticationSucceeded
onAuthenticationFailed
```

Anfangen

So starten Sie das FingerPrint-Ereignisaufruf als Authentifizierungsmethode mit Crypto

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

Stornieren

um das Auflisten des Scanneraufrufs zu beenden

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

Sobald der Fingerabdruck (oder das Kennwort) überprüft wurde, wird der FingerprintManager.AuthenticationCallback # onAuthenticationSucceeded () - Callback aufgerufen.

```
@Override

public void onAuthenticationSucceeded(AuthenticationResult result) {

            }
```

So verwenden Sie die Android Fingerprint API zum Speichern von

Benutzerkennwörtern

Diese Beispiel-Helferklasse interagiert mit dem Fingerprint Manager und führt die Verschlüsselung und Entschlüsselung des Kennworts durch. Beachten Sie, dass in diesem Beispiel als Verschlüsselungsverfahren AES verwendet wird. Dies ist nicht der einzige Weg zum Verschlüsseln, und es gibt andere Beispiele . In diesem Beispiel werden die Daten auf folgende Weise verschlüsselt und entschlüsselt:

Verschlüsselung:

1. Der Benutzer gibt dem Helfer das gewünschte nicht verschlüsselte Passwort.
2. Der Benutzer muss einen Fingerabdruck bereitstellen.
3. Nach der Authentifizierung erhält der Helfer einen Schlüssel vom `KeyStore` und verschlüsselt das Kennwort mithilfe einer `Cipher` .
4. Kennwort und IV-Salt (IV wird für jede Verschlüsselung neu erstellt und nicht wiederverwendet) werden in den gemeinsamen Einstellungen gespeichert, die später im Entschlüsselungsprozess verwendet werden.

Entschlüsselung:

1. Benutzer fordert an, das Kennwort zu entschlüsseln.
2. Der Benutzer muss einen Fingerabdruck bereitstellen.
3. Der Helfer erstellt eine `Cipher` mithilfe des IV. Sobald der Benutzer authentifiziert ist, erhält der KeyStore einen Schlüssel vom `KeyStore` und entschlüsselt das Kennwort.

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

```
    }
}
```

Die folgende Aktivität ist ein sehr einfaches Beispiel dafür, wie Sie ein vom Benutzer gespeichertes Passwort erhalten und mit dem Helfer interagieren können.

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

Fingerprint-API in Android online lesen: https://riptutorial.com/de/android/topic/7523/fingerprint-api-in-android

Kapitel 93: Firebase

Einführung

Firebase ist eine Plattform für Mobil- und Webanwendungen mit Tools und Infrastruktur, die Entwicklern dabei helfen soll, qualitativ hochwertige Apps zu erstellen.

Eigenschaften

Firebase Cloud Messaging, Firebase Auth, Echtzeitdatenbank, Firebase-Speicher, Firebase-Hosting, Firebase-Testlabor für Android, Firebase-Absturzberichte.

Bemerkungen

Firebase - Erweiterte Dokumentation:

An einem anderen Tag finden Sie weitere Themen und Beispiele zur Verwendung von Firebase.

Andere verwandte Themen:

- Firebase Echtzeitdatenbank
- Indizierung der Firebase-App
- Firebase-Absturzberichterstattung
- Firebase Cloud Messaging

Examples

Erstellen Sie einen Firebase-Benutzer

```
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;
    }
}
```

Anmelden Firebase-Benutzer mit E-Mail-Adresse und Kennwort

```
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));
    }
}
```

E-Mail zum Zurücksetzen des Firebase-Passworts senden

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

Aktualisieren der E-Mail-Adresse eines Firebase-Benutzers

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

Ändere das Passwort

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

Authentifizieren Sie den Firebase-Benutzer erneut

```
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-Speichervorgänge

In diesem Beispiel können Sie folgende Vorgänge ausführen:

1. Verbinden Sie sich mit Firebase Storage
2. Erstellen Sie ein Verzeichnis mit dem Namen "images"
3. Laden Sie eine Datei in das Bilderverzeichnis hoch
4. Laden Sie eine Datei aus dem Bilderverzeichnis herunter
5. Löschen Sie eine Datei aus dem Bilderverzeichnis

```
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();
    }
}
```

Standardmäßig wenden Firebase Storage-Regeln die Authentifizierungseinschränkung an. Wenn der Benutzer nur dann authentifiziert ist, kann er Firebase Storage-Vorgänge ausführen. Ich habe den Authentifizierungsteil in dieser Demo durch Aktualisieren der Speicherregeln deaktiviert. Bisher sahen die Regeln so aus:

```
 service firebase.storage {
   match /b/**something**.appspot.com/o {
     match /{allPaths=**} {
       allow read, write: if request.auth != null;
     }
   }
 }
```

Ich habe aber die Authentifizierung übersprungen:

```
service firebase.storage {
```

```
  match /b/**something**.appspot.com/o {
    match /{allPaths=**} {
      allow read, write;
    }
  }
}
```

Firebase Cloud Messaging

Zunächst müssen Sie Ihr Projekt einrichten, indem Sie Firebase zu Ihrem Android-Projekt hinzufügen. Befolgen Sie dazu die in diesem Thema beschriebenen Schritte .

Richten Sie Firebase und das FCM-SDK ein

Fügen Sie der `build.gradle` Datei auf App-Ebene die FCM-Abhängigkeit `build.gradle`

```
dependencies {
 compile 'com.google.firebase:firebase-messaging:11.0.4'
}
```

Und ganz unten (das ist wichtig):

```
// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

Bearbeiten Sie Ihr App-Manifest

Fügen Sie dem Manifest Ihrer App Folgendes hinzu:

- Ein Dienst, der `FirebaseMessagingService` . Dies ist erforderlich, wenn Sie Nachrichten bearbeiten möchten, die über das Empfangen von Benachrichtigungen über Apps im Hintergrund hinausgehen.
- Ein Dienst, der `FirebaseInstanceIdService` um die Erstellung, Rotation und Aktualisierung von Registrierungstoken erweitert.

Zum Beispiel:

```
    <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>
```

Hier sind einfache Implementierungen der 2 Dienste.

Um das aktuelle Registrierungstoken abzurufen, erweitern Sie die `FirebaseInstanceIdService` Klasse und überschreiben die `onTokenRefresh()` Methode:

```
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
    }
}
```

Verwenden Sie zum Empfangen von Nachrichten einen Dienst, der `FirebaseMessagingService` und überschreiben Sie die `onMessageReceived` Methode.

```
public class MyFcmListenerService extends FirebaseMessagingService {

    /**
     * Called when message is received.
     *
     * @param remoteMessage Object representing the message received from Firebase Cloud
Messaging.
     */
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        String from = remoteMessage.getFrom();

        // Check if message contains a data payload.
        if (remoteMessage.getData().size() > 0) {
            Log.d(TAG, "Message data payload: " + remoteMessage.getData());
            Map<String, String> data = remoteMessage.getData();
        }

        // Check if message contains a notification payload.
        if (remoteMessage.getNotification() != null) {
            Log.d(TAG, "Message Notification Body: " +
remoteMessage.getNotification().getBody());
        }

        // do whatever you want with this, post your own notification, or update local state
    }
```

in **Firebase** können Benutzer durch ihr Verhalten wie „AppVersion, kostenlos Benutzer, Benutzer erwerben, noch spezielle Regeln“ zusammengefasst und dann Benachrichtigung an bestimmten Gruppe von Senden **Topic** - Funktion in Firebase senden.
Benutzer in der Themennutzung registrieren

```
FirebaseMessaging.getInstance().subscribeToTopic("Free");
```

Senden Sie dann in der FireBase-Konsole eine Benachrichtigung nach Themenname

Weitere Informationen zum Thema Firebase Cloud Messaging .

Fügen Sie Firebase zu Ihrem Android-Projekt hinzu

Hier sind vereinfachte Schritte (basierend auf der offiziellen Dokumentation) erforderlich, um ein Firebase-Projekt zu erstellen und es mit einer Android-App zu verbinden.

Fügen Sie Firebase Ihrer App hinzu

1. Erstellen Sie ein Firebase-Projekt in der Firebase-Konsole und klicken **Sie auf Neues Projekt erstellen** .
2. Klicken **Sie auf Firebase zu Ihrer Android-App hinzufügen** und folgen Sie den Installationsschritten.
3. Wenn Sie dazu aufgefordert werden, geben Sie **den Paketnamen** Ihrer **App ein** .
   Es ist wichtig, den vollständig qualifizierten Paketnamen einzugeben, den Ihre App verwendet. Dies kann nur eingestellt werden, wenn Sie Ihrem Firebase-Projekt eine App hinzufügen.
4. Am Ende laden Sie eine `google-services.json` Datei herunter. Sie können diese Datei jederzeit erneut herunterladen.
5. Wenn Sie dies noch nicht getan haben, kopieren Sie die Datei `google-services.json` in den `google-services.json` Ihres Projekts, normalerweise `app/` .

Im nächsten Schritt fügen Sie das SDK hinzu, um die Firebase-Bibliotheken in das Projekt zu integrieren.

Fügen Sie das SDK hinzu

Um die Firebase-Bibliotheken in eines Ihrer eigenen Projekte zu integrieren, müssen Sie einige grundlegende Aufgaben zur Vorbereitung Ihres Android Studio-Projekts ausführen. Möglicherweise haben Sie dies bereits beim Hinzufügen von Firebase zu Ihrer App getan.

1. Fügen Sie der `build.gradle` Datei `build.gradle` Regeln `build.gradle` , um das **google-services-Plugin aufzunehmen** :

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.1.0'
```

```
    }
}
```

`app/build.gradle` dann in Ihrer Modul-Gradle-Datei (normalerweise `app/build.gradle`) die `app/build.gradle` Plugin-Zeile am Ende der Datei hinzu, um das Gradle-Plugin zu aktivieren:

```
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'
```

Der letzte Schritt ist das Hinzufügen der Abhängigkeiten für das Firebase-SDK mithilfe einer oder mehrerer **Bibliotheken, die** für die verschiedenen Firebase-Funktionen **verfügbar** sind.

Gradle-Abhängigkeitslinie	Bedienung
com.google.firebase: firebase-core: 11.0.4	Analytics
com.google.firebase: firebase-database: 11.0.4	Echtzeit-Datenbank
com.google.firebase: firebase-storage: 11.0.4	Lager
com.google.firebase: firebase-crash: 11.0.4	Crash-Berichterstellung
com.google.firebase: firebase-auth: 11.0.4	Authentifizierung
com.google.firebase: firebase-messaging: 11.0.4	Cloud Messaging / Benachrichtigungen
com.google.firebase: firebase-config: 11.0.4	Remote-Konfig
com.google.firebase: firebase-invites: 11.0.4	Einladungen / Dynamische Links
com.google.firebase: firebase-ads: 11.0.4	AdMob
com.google.android.gms: play-services-appindexing: 11.0.4	App-Indizierung

Firebase-Echtzeitdatenbank: Wie werden Daten eingestellt / abgerufen?

Hinweis: Lassen Sie uns eine anonyme Authentifizierung für das Beispiel einrichten

```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

Erstellen Sie anschließend ein Kind, indem Sie Ihre Datenbankadresse bearbeiten. Zum Beispiel:

https://your-project.firebaseio.com/ an **https://your-project.firebaseio.com/chat**

Wir werden Daten von unserem Android-Gerät an diesen Ort übertragen. Sie **müssen** die Datenbankstruktur (Registerkarten, Felder usw.) **nicht** erstellen. Sie wird automatisch erstellt, wenn Sie ein Java-Objekt an Firebase senden.

Erstellen Sie ein Java-Objekt, das alle Attribute enthält, die Sie an die Datenbank senden möchten:

```
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;
    }
}
```

Dann in Ihrer Aktivität:

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

So senden Sie einen Wert:

```
ChatMessage msg = new ChatMessage("user1", "Hello World!");
reference.push().setValue(msg);
```

So empfangen Sie Änderungen, die in der Datenbank auftreten:

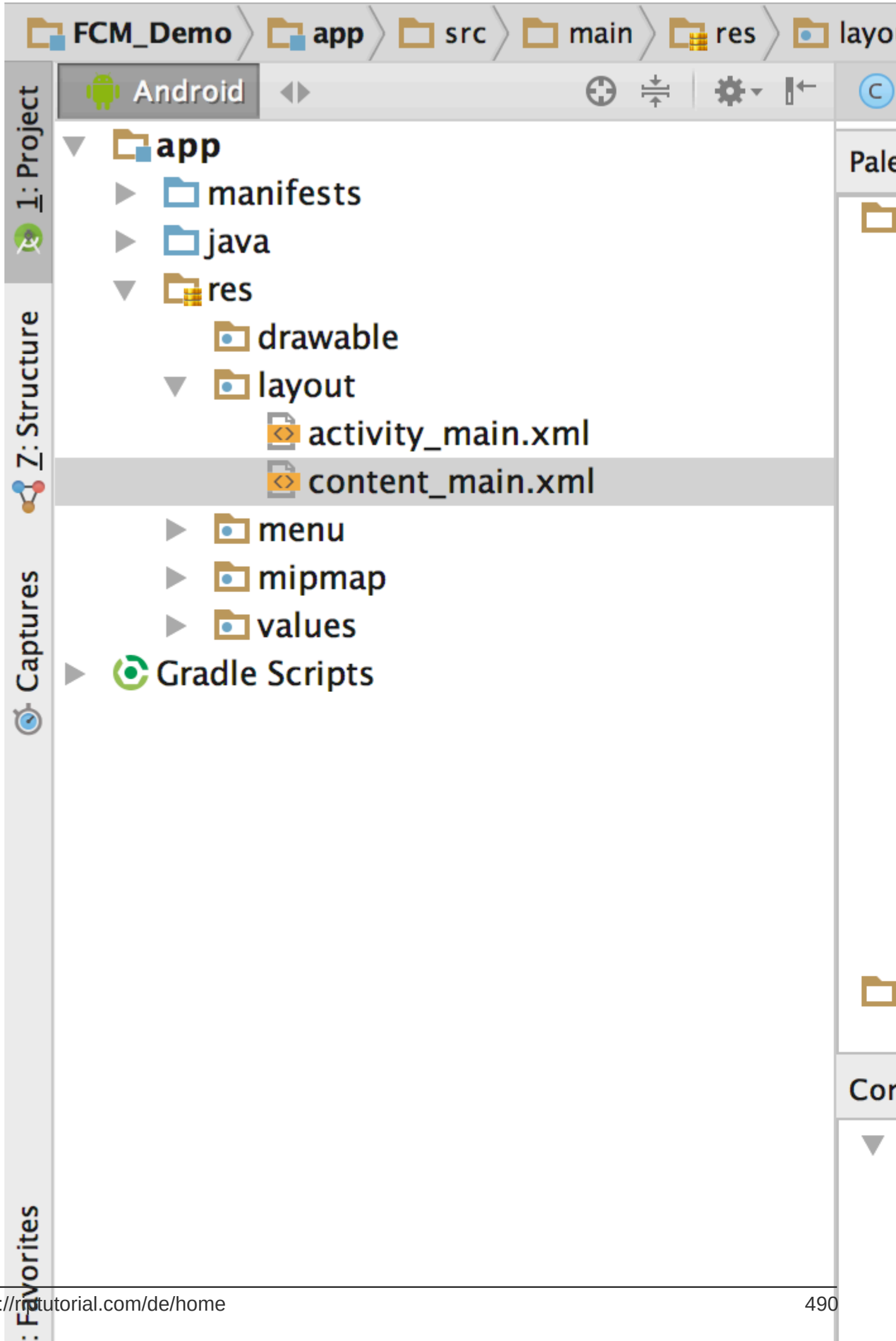
```
reference.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        ChatMessage msg = dataSnapshot.getValue(ChatMessage.class);
        Log.d(TAG, msg.getUsername()+" "+msg.getMessage());
    }

    public void onChildChanged(DataSnapshot dataSnapshot, String s) {}
    public void onChildRemoved(DataSnapshot dataSnapshot) {}
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {}
    public void onCancelled(DatabaseError databaseError) {}
});
```

Demo von FCM-basierten Benachrichtigungen

In diesem Beispiel wird die Verwendung der Firebase Cloud Messaging-Plattform (FCM) veranschaulicht. FCM ist ein Nachfolger von Google Cloud Messaging (GCM). Es sind keine C2D_MESSAGE-Berechtigungen von den App-Benutzern erforderlich.

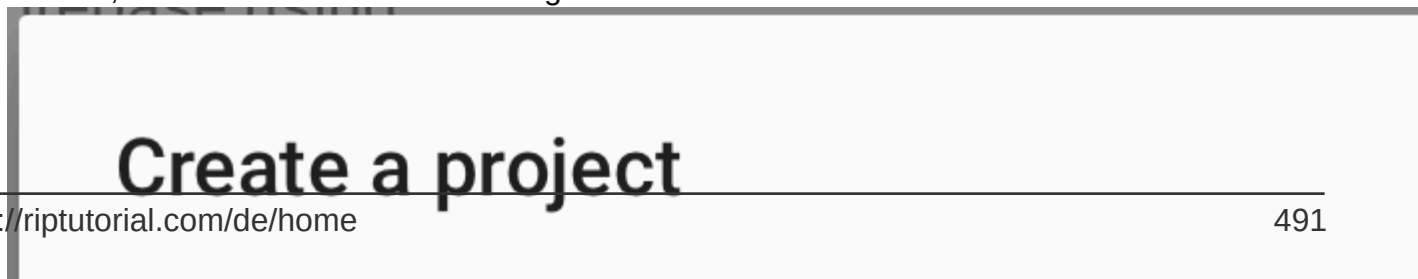
Schritte zur Integration von FCM sind wie folgt.

1. Erstellen eines Beispielprojekts "Hallo Welt" in Android Studio Der Bildschirm Ihres Android-Studios sieht wie folgt aus.


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. Der nächste Schritt ist das Einrichten des Firebase-Projekts. Besuchen Sie https://console.firebase.google.com und erstellen Sie ein Projekt mit einem identischen Namen, damit Sie es leicht nachverfolgen können.

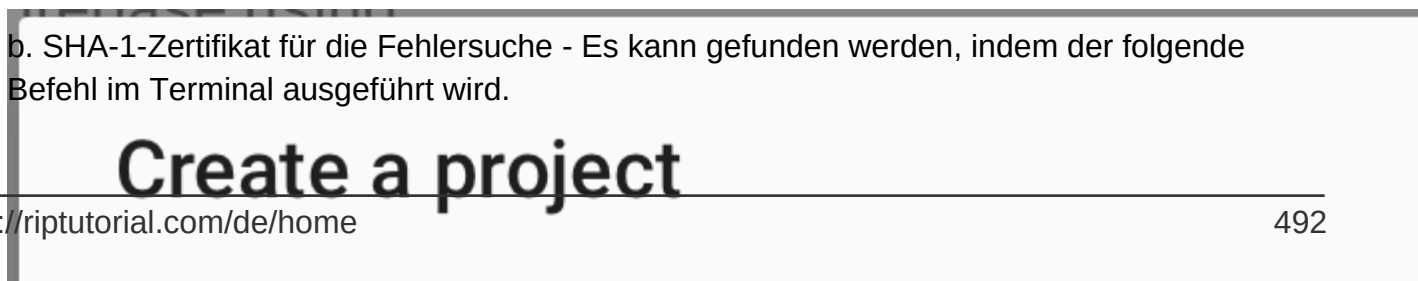
https://console.firebase.google.com und erstellen Sie ein Projekt mit einem identischen Namen, damit Sie es leicht nachverfolgen können.



3. Jetzt ist es an der Zeit, Ihrem soeben erstellten Android-Projekt Firebase hinzuzufügen. Sie benötigen den Paketnamen Ihres Projekts und das Debug-Signaturzertifikat SHA-1 (optional).

   ein. Paketname - Kann aus der Android-Manifest-XML-Datei gefunden werden.

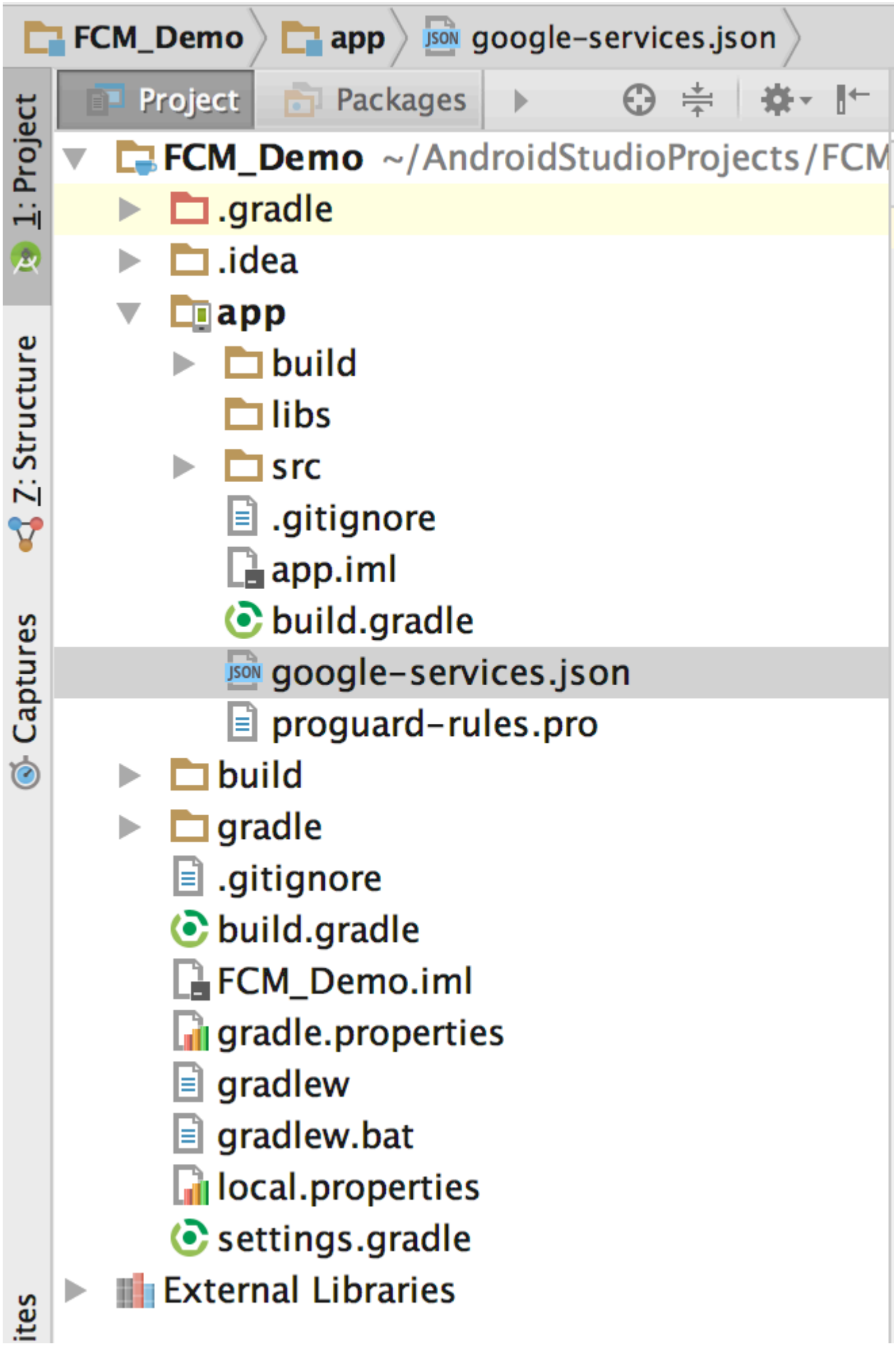
   b. SHA-1-Zertifikat für die Fehlersuche - Es kann gefunden werden, indem der folgende Befehl im Terminal ausgeführt wird.



```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
```

Geben Sie diese Informationen in die Firebase-Konsole ein und fügen Sie die App zum Firebase-Projekt hinzu. Sobald Sie auf die Schaltfläche App hinzufügen klicken, lädt Ihr Browser automatisch eine JSON-Datei mit dem Namen "google-services.json" herunter.

4. Kopieren Sie nun die soeben heruntergeladene Datei google-services.json in das Stammverzeichnis Ihres Android-App-Moduls.


FCM_Demo
app
google-services.json
Project
Packages
1: Project
Z: Structure
Captures
FCM_Demo ~/AndroidStudioProjects/FCM
.gradle
.idea
app
build
libs
src
.gitignore
app.iml
build.gradle
google-services.json
proguard-rules.pro
build
gradle
.gitignore
build.gradle
FCM_Demo.iml
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
External Libraries

5. Folgen Sie den Anweisungen auf der Firebase-Konsole, während Sie fortfahren. ein. Fügen Sie Ihrem Projektlevel build.gradle die folgende Codezeile hinzu

```
dependencies{ classpath 'com.google.gms:google-services:3.1.0' .....
```

b. Fügen Sie am Ende Ihrer Anwendungsebene build.gradle die folgende Codezeile hinzu.

```
    //following are the dependencies to be added
    compile 'com.google.firebase:firebase-messaging:11.0.4'
    compile 'com.android.support:multidex:1.0.1'
}
// this line goes to the end of the file
apply plugin: 'com.google.gms.google-services'
```

c. Android Studio würde Sie bitten, das Projekt zu synchronisieren. Klicken Sie auf Jetzt synchronisieren.

6. Als nächstes müssen Sie zwei Dienste hinzufügen. ein. Ein FirebaseMessagingService wurde wie folgt mit einem Intent-Filter erweitert

```
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
```

b. Ein erweiterter FirebaseInstanceIDService.

```
<intent-filter>
    <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
</intent-filter>
```

7. FirebaseMessagingService-Code sollte so aussehen.

```
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 sollte so aussehen.

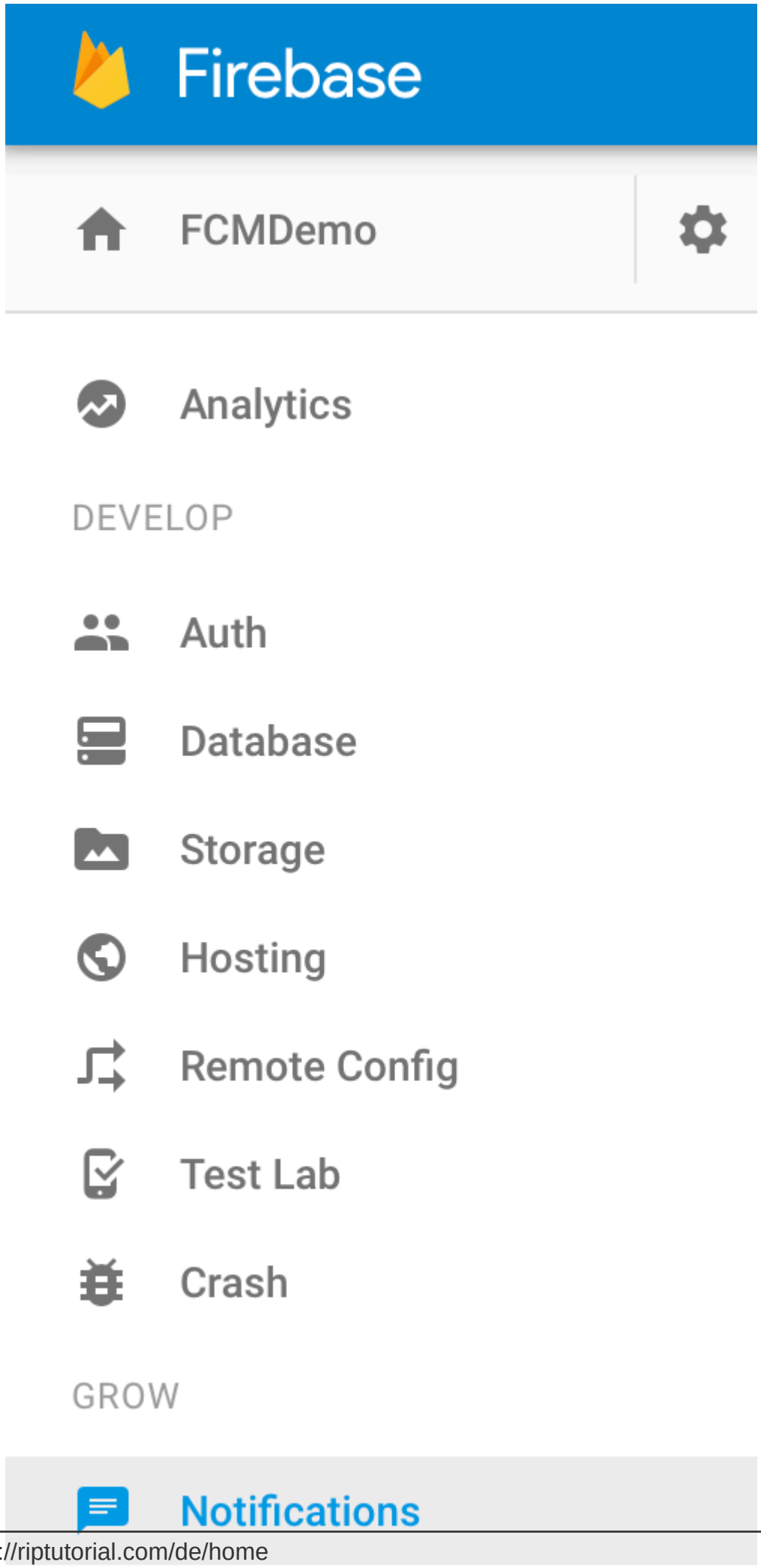
```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

import com.google.firebase.iid.FirebaseInstanceIdService;

public class MyFirebaseInstanceIDService extends FirebaseInstanceIdService {
    public MyFirebaseInstanceIDService() {
    }
```

```
}
```

9. Nun ist es Zeit, das Geräteregistrierungstoken zu erfassen. Fügen Sie der onCreate-Methode von MainActivity die folgende Codezeile hinzu.

```
String token = FirebaseInstanceId.getInstance().getToken();
Log.d("FCMAPP", "Token is "+token);
```

10. Sobald wir das Zugriffstoken haben, können wir die Firebase-Konsole zum Senden der Benachrichtigung verwenden. Führen Sie die App auf Ihrem Android-Handset aus.


Firebase
FCMDemo
Analytics
DEVELOP
Auth
Database
Storage
Hosting
Remote Config
Test Lab
Crash
GROW
Notifications

Kapitel 94: Firebase Cloud Messaging

Einführung

Firebase Cloud Messaging (FCM) ist eine plattformübergreifende Messaging-Lösung, mit der Sie Nachrichten zuverlässig und kostenlos versenden können.

Mit FCM können Sie eine Client-App darüber informieren, dass neue E-Mails oder andere Daten zur Synchronisierung verfügbar sind. Sie können Benachrichtigungsnachrichten senden, um den Wiedereinstieg und die Aufbewahrung von Benutzern zu fördern. Für Anwendungsfälle wie Instant Messaging kann eine Nachricht eine Nutzlast von bis zu `4KB` an eine Client-App übertragen.

Examples

Einrichten einer Firebase Cloud Messaging Client-App auf Android

1. Schließen Sie den Installations- und Setup-Teil ab , um Ihre App mit Firebase zu verbinden. Dadurch wird das Projekt in Firebase erstellt.

2. Fügen Sie der `build.gradle` Datei auf Modulebene die Abhängigkeit für Firebase Cloud Messaging `build.gradle` :

```
dependencies {
     compile 'com.google.firebase:firebase-messaging:10.2.1'
}
```

Jetzt können Sie mit dem FCM in Android arbeiten.

FCM-Clients erfordern Geräte mit `Android 2.3` oder höher, auf denen auch die Google Play Store-App installiert ist, oder einen Emulator, auf dem Android 2.3 mit Google-APIs ausgeführt wird.

Bearbeiten Sie Ihre `AndroidManifest.xml` Datei

```
<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>
```

Registrierungs-Token

Beim ersten Start Ihrer App generiert das FCM-SDK ein Registrierungstoken für die Instanz der Clientanwendung.
Wenn Sie auf einzelne Geräte abzielen oder Gerätegruppen erstellen möchten, müssen Sie auf dieses Token zugreifen, indem Sie `FirebaseInstanceIdService` .

Der Callback von `onTokenRefresh` wird immer dann `onTokenRefresh` wenn ein neues Token generiert wird. Sie können die Methode `FirebaseInstanceID.getToken()` , um das aktuelle Token abzurufen.

Beispiel:

```
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);

    }

}
```

Dieser Code, den ich in meiner App implementiert habe, um Bilder, Nachrichten und auch einen Link zum Öffnen in Ihrem WebView zu verschieben

Dies ist mein FirebaseMessagingService

```
    public class MyFirebaseMessagingService extends FirebaseMessagingService {
Bitmap bitmap;
@Override
public void onMessageReceived(RemoteMessage remoteMessage) {
    String message = remoteMessage.getData().get("message");
    //imageUri will contain URL of the image to be displayed with Notification
    String imageUri = remoteMessage.getData().get("image");
    String link=remoteMessage.getData().get("link");

    //To get a Bitmap image from the URL received
    bitmap = getBitmapfromUrl(imageUri);
    sendNotification(message, bitmap,link);

}


/**
 * Create and show a simple notification containing the received FCM message.
 */
```

```
private void sendNotification(String messageBody, Bitmap image, String link) {
    Intent intent = new Intent(this, NewsListActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    intent.putExtra("LINK",link);
    PendingIntent pendingIntent = PendingIntent.getActivity(this, 0 /* Request code */,
intent,
            PendingIntent.FLAG_ONE_SHOT);
    Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
    NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(this)
            .setLargeIcon(image)/*Notification icon image*/
            .setSmallIcon(R.drawable.hindi)
            .setContentTitle(messageBody)
            .setStyle(new NotificationCompat.BigPictureStyle()
                    .bigPicture(image))/*Notification with Image*/
            .setAutoCancel(true)
            .setSound(defaultSoundUri)
            .setContentIntent(pendingIntent);
    NotificationManager notificationManager =
            (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

    notificationManager.notify(0 /* ID of notification */, notificationBuilder.build());
}
public Bitmap getBitmapfromUrl(String imageUrl) {
    try {
        URL url = new URL(imageUrl);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setDoInput(true);
        connection.connect();
        InputStream input = connection.getInputStream();
        Bitmap bitmap = BitmapFactory.decodeStream(input);
        return bitmap;

    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        return null;

    }
}}
```

Dies ist MainActivity, um den Link in meinem WebView oder einem anderen Browser zu öffnen, der von Ihren Anforderungen abhängt.

```
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();
    }}
```

Nachrichten erhalten

Verwenden Sie zum Empfangen von Nachrichten einen Dienst, der `FirebaseMessagingService` und überschreiben Sie die `onMessageReceived` Methode.

```
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());
        }

        //.....
    }
```

Wenn sich die App im Hintergrund befindet, leitet Android Benachrichtigungen in die Taskleiste. Ein Benutzer, der auf die Benachrichtigung tippt, öffnet standardmäßig das App-Startprogramm.

Dies umfasst Nachrichten, die sowohl Benachrichtigung als auch Datennutzlast enthalten (und alle von der Benachrichtigungskonsole gesendeten Nachrichten). In diesen Fällen wird die Benachrichtigung an die Systemablage des Geräts gesendet, und die Datennutzlast wird in den Extras der Absicht Ihrer Starteraktivität geliefert.

Hier ein kurzer Rückblick:

App-Status	Benachrichtigung	Daten	Beide
Vordergrund	`onMessageReceived`	`onMessageReceived`	`onMessageReceived`
Hintergrund	System Tray	`onMessageReceived`	Benachrichtigung: Taskleiste
			Daten: in Extras der Absicht.

Abonnieren Sie ein Thema

Client-Apps können ein beliebiges vorhandenes Thema abonnieren oder ein neues Thema erstellen. Wenn eine Client-App einen neuen Themennamen abonniert, wird ein neues Thema mit diesem Namen in FCM erstellt, und jeder Client kann ihn anschließend abonnieren.

Verwenden Sie zum `subscribeToTopic()` eines Themas die `subscribeToTopic()` Methode, die den Themennamen angibt:

```
FirebaseMessaging.getInstance().subscribeToTopic("myTopic");
```

Firebase Cloud Messaging online lesen: https://riptutorial.com/de/android/topic/8826/firebase-cloud-messaging

Kapitel 95: Firebase Echtzeitdatenbank

Bemerkungen

Andere verwandte Themen:

- Firebase

Examples

Firebase Realtime DataBase-Ereignishandler

Initialisieren Sie zuerst FirebaseDatabase:

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

Schreiben Sie in Ihre Datenbank:

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Lesen Sie aus Ihrer Datenbank:

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

Daten zu Android-Ereignissen abrufen:

```
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);
```

Schnelle Einrichtung

1. Schließen Sie den Installations- und Setup-Teil ab , um Ihre App mit Firebase zu verbinden. Dadurch wird das Projekt in Firebase erstellt.

2. Fügen Sie der `build.gradle` Datei auf Modulebene die Abhängigkeit für die Firebase-Echtzeitdatenbank `build.gradle` :

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

3. Konfigurieren Sie die Firebase-Datenbankregeln

Jetzt können Sie mit der Echtzeitdatenbank in Android arbeiten.

Beispielsweise schreiben Sie eine `Hello World` Nachricht unter dem `message` in die Datenbank.

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Entwerfen und Verstehen, wie Echtzeitdaten aus der Firebase-Datenbank abgerufen werden

In diesem Beispiel wird davon ausgegangen, dass Sie bereits eine Firebase-Echtzeitdatenbank

eingerichtet haben. Wenn Sie ein Anfänger sind, informieren Sie sich hier, wie Sie Firebase zu Ihrem Android-Projekt hinzufügen.

Fügen Sie zunächst die Abhängigkeit der Firebase-Datenbank zur Datei *build.gradle* auf App-Ebene *hinzu* :

```
compile 'com.google.firebase:firebase-database:9.4.0'
```

Lassen Sie uns nun eine Chat-App erstellen, die Daten in der Firebase-Datenbank speichert.

Schritt 1: Erstellen Sie eine Klasse mit dem Namen Chat

Erstellen Sie einfach eine Klasse mit einigen grundlegenden Variablen, die für den Chat erforderlich sind:

```
public class Chat{
    public String name, message;
}
```

Schritt 2: Erstellen Sie einige JSON-Daten

Um Daten an die Firebase-Datenbank zu senden / abzurufen, müssen Sie JSON verwenden. Nehmen wir an, dass einige Chats bereits auf Stammebene in der Datenbank gespeichert sind. Die Daten dieser Chats könnten wie folgt aussehen:

```
[
    {
        "name":"John Doe",
        "message":"My first Message"
    },
    {
        "name":"John Doe",
        "message":"Second Message"
    },
    {
        "name":"John Doe",
        "message":"Third Message"
    }
]
```

Schritt 3: Hinzufügen der Hörer

Es gibt drei Arten von Zuhörern. Im folgenden Beispiel verwenden wir den `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();
    }
});
```

Schritt 4: Fügen Sie Daten zur Datenbank hinzu

Erstellen Sie einfach ein Chat-Klassenobjekt und fügen Sie die Werte wie folgt hinzu:

```
Chat chat=new Chat();
chat.name="John Doe";
chat.message="First message from android";
```

Rufen Sie nun wie in der Abrufsitzung einen Verweis auf den Chats-Knoten ab:

```
DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference().child("chats");
```

Bevor Sie mit dem Hinzufügen von Daten beginnen, bedenken Sie, dass Sie eine weitere ausführliche Referenz benötigen, da ein Chat-Knoten mehrere weitere Knoten hat. Wenn Sie einen neuen Chat hinzufügen, müssen Sie einen neuen Knoten hinzufügen, der die Chat-Details enthält. Wir können einen neuen und eindeutigen Namen des Knotens mit der Funktion `push()` für das `DatabaseReference` Objekt generieren, das eine andere `DatabaseReference` , die wiederum auf einen neu gebildeten Knoten verweist, um die Chat-Daten einzufügen.

Beispiel

```
// The parameter is the chat object that was newly created a few lines above.
chatDb.push().setValue(chat);
```

Die `setValue()` Funktion stellt sicher, dass alle `onDataChanged` Funktionen der Anwendung aufgerufen werden (einschließlich des gleichen Geräts). `onDataChanged` ist der angehängte Listener des Chats "chats".

Denormalisierung: Flache Datenbankstruktur

Denormalisierung und eine flache Datenbankstruktur sind erforderlich, um separate Aufrufe effizient herunterzuladen. Mit der folgenden Struktur können auch wechselseitige Beziehungen aufrechterhalten werden. Der Nachteil dieses Ansatzes ist, dass Sie die Daten immer an mehreren Stellen aktualisieren müssen.

Stellen Sie sich zum Beispiel eine App vor, mit der der Benutzer Nachrichten für sich selbst speichern kann (Memos).

Gewünschte flache Datenbankstruktur:

```
|--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"
```

Die verwendete Memo-Klasse

```
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;
    }
}
```

Abrufen der Memos eines Benutzers

```
//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) { }
    }
```

Memo erstellen

```
//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);
```

Nach dem Push oder der Datenbank sieht das so aus:

```
|--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"
```

Grundlegendes zur Firebase-JSON-Datenbank

Bevor wir uns die Hände mit Code schmutzig machen, muss man verstehen, wie Daten in Firebase gespeichert werden. Im Gegensatz zu relationalen Datenbanken speichert Firebase Daten im JSON-Format. Stellen Sie sich jede Zeile in einer relationalen Datenbank als JSON-Objekt vor (bei dem es sich im Wesentlichen um ein ungeordnetes Schlüsselwertpaar handelt). Der Spaltenname wird also zu einem Schlüssel und der in dieser Spalte für eine bestimmte Zeile gespeicherte Wert ist der Wert. Auf diese Weise wird die gesamte Zeile als JSON-Objekt dargestellt und eine Liste davon stellt eine gesamte Datenbanktabelle dar. Der unmittelbare Vorteil, den ich dafür sehe, ist, dass die Schemaänderung im Vergleich zu alten RDBMS-Systemen wesentlich kostengünstiger wird. Es ist einfacher, einem JSON ein paar weitere Attribute hinzuzufügen, als eine Tabellenstruktur zu ändern.

Hier ein Beispiel-JSON, um zu zeigen, wie Daten in Firebase gespeichert werden:

```
{
 "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
  }
```

Dies zeigt deutlich, wie Daten, die wir in relationalen Datenbanken gespeichert haben, im JSON-Format gespeichert werden können. Als Nächstes wollen wir sehen, wie diese Daten in Android-Geräten gelesen werden.

Daten von der Firebase abrufen

Ich gehe davon aus, dass Sie bereits wissen, wie Sie in Android Studio gradle Abhängigkeiten hinzufügen können. Wenn Sie nicht einfach der Anleitung von hier folgen. Fügen Sie Ihre App in der Firebase-Konsole hinzu, und synchronisieren Sie das Android Studio, nachdem Sie Abhängigkeiten hinzugefügt haben. Es werden nicht alle Abhängigkeiten benötigt, nur Firebase-Datenbank und Firebase-Authentifizierung.

Nun, da wir wissen, wie Daten gespeichert werden und wie Abstufungsabhängigkeiten hinzugefügt werden, sehen wir uns an, wie Sie mit dem importierten Android-SDB (Firebase) Daten abrufen können.

Erstellen Sie eine Firebase-Datenbankreferenz

```
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)
```

Von hier aus können Sie mehrere child () - Methodenaufrufe verketten, um auf die Daten zu verweisen, an denen Sie interessiert sind. Wenn beispielsweise die Daten wie im vorherigen Abschnitt beschrieben gespeichert werden und Sie auf Bruce Wayne-Benutzer verweisen möchten, können Sie Folgendes verwenden:

```
DatabaseReference bruceWayneRef = userDBRef.child("371298");
// 371298 is key of bruce wayne user in JSON structure (previous section)
```

Oder übergeben Sie einfach die gesamte Referenz an das JSON-Objekt:

```
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..."
```

Da wir nun die Referenz der Daten haben, die wir abrufen möchten, können wir Listener verwenden, um Daten in Android-Apps abzurufen. Im Gegensatz zu herkömmlichen Aufrufen, bei denen REST-API-Aufrufe mithilfe von Retrofit oder Volley ausgelöst werden, ist hier ein einfacher Callback-Listener erforderlich, um die Daten abzurufen. Firebase sdk ruft die Callback-Methoden auf und Sie sind fertig.

Es gibt grundsätzlich zwei Arten von Listenern, die Sie anfügen können, einer ist

ValueEventListener und der andere ist ChildEventListener (im nächsten Abschnitt beschrieben). Bei jeder Änderung der Daten unter dem Knoten, für den wir Verweise und Listener hinzugefügt haben, geben Wertereignis-Listener die gesamte JSON-Struktur zurück, und der untergeordnete Ereignislistener gibt ein bestimmtes untergeordnetes Element zurück, an dem die Änderung stattgefunden hat. Beide sind auf ihre Weise nützlich. Um die Daten von Firebase abzurufen, können wir einer Firebase-Datenbankreferenz einen oder mehrere Listener hinzufügen (list userDBRef, die wir zuvor erstellt haben).

Hier ist ein Beispielcode (Code-Erklärung nach Code):

```
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
    });
```

Haben Sie festgestellt, dass der Klassentyp bestanden wurde? DataSnapshot kann JSON-Daten in unsere definierten POJOs konvertieren, indem Sie einfach den richtigen Klassentyp übergeben.

Wenn Ihr Anwendungsfall nicht jedes Mal die gesamten Daten (in unserem Fall user_base-Tabelle) benötigt, wenn eine kleine Änderung auftritt oder Sie ***die Daten nur einmal abrufen*** möchten, können Sie die ***addListenerForSingleValueEvent () -*** Methode der Datenbankreferenz verwenden. Dadurch wird der Rückruf nur einmal ausgelöst.

```
userDBRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // Do something
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Do something about the error
    });
```

Die obigen Beispiele geben Ihnen den Wert des JSON-Knotens. Um den Schlüssel zu erhalten, rufen Sie einfach an:

```
String myKey = dataSnapshot.getKey();
```

Auf untergeordnete Updates achten

Nutzen Sie einen Anwendungsfall wie eine Chat-App oder eine kollaborative Einkaufslisten-App (die im Wesentlichen eine Liste von Objekten erfordert, die mit den Benutzern synchronisiert werden müssen). Wenn Sie die Firebase-Datenbank verwenden und einen Wertereignis-Listener

zum übergeordneten Knoten des Chat-Bereichs oder zum übergeordneten Knoten der Einkaufsliste hinzufügen, endet die gesamte Chat-Struktur ab dem Beginn der Zeit (ich meinte den Beginn Ihres Chats), wenn ein Chat-Knoten hinzugefügt wird dh jemand sagt hallo). Das wollen wir nicht. Was uns interessiert, ist nur der neue Knoten oder nur der alte Knoten, der gelöscht oder geändert wurde. Die unveränderten sollten nicht zurückgegeben werden.

In diesem Fall können wir ChildEvenListener verwenden . Ohne weitere Hinweise hier ein Codebeispiel (siehe vorherige Abschnitte für JSON-Beispieldaten):

```
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) {
    });
```

Methodennamen sind selbsterklärend. Wie Sie sehen, wenn ein neuer Benutzer hinzugefügt oder eine Eigenschaft eines vorhandenen Benutzers geändert oder ein Benutzer gelöscht oder entfernt wird, wird die entsprechende Rückrufmethode des untergeordneten Ereignislisteners mit relevanten Daten aufgerufen. Wenn Sie also die Benutzeroberfläche zum Beispiel für eine Chat-App auf dem neuesten Stand halten, holen Sie sich die JSON-Analyse von onChildAdded () in POJO und passen Sie sie in Ihre Benutzeroberfläche ein. Denken Sie daran, Ihren Listener zu entfernen, wenn der Benutzer den Bildschirm verlässt.

onChildChanged () gibt den gesamten untergeordneten Wert mit geänderten (neuen) Eigenschaften an.

onChiledRemoved () gibt den entfernten untergeordneten Knoten zurück.

Daten mit Paginierung abrufen

Wenn Sie über eine große JSON-Datenbank verfügen, ist das Hinzufügen eines Wertereignis-Listeners nicht sinnvoll. Die riesige JSON wird zurückgegeben, und das Parsen wäre zeitaufwändig. In solchen Fällen können wir die Paginierung verwenden, um einen Teil der Daten abzurufen und anzuzeigen oder zu verarbeiten. Ein bisschen wie faules Laden oder wie das Abrufen alter Chats, wenn Benutzer auf älteren Chat klicken klicken. In diesem Fall kann Query verwendet werden.

Nehmen wir unser altes Beispiel in den vorherigen Abschnitten. Die Benutzerbasis enthält 3 Benutzer, wenn die Anzahl der Benutzer dreihunderttausend beträgt und Sie die Benutzerliste in Stapel von 50 abrufen möchten:

```
// 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
    });
```

Hier können Wert- oder untergeordnete Ereignisse hinzugefügt und angehört werden. Rufen Sie die Abfrage erneut auf, um die nächsten 50 **abzurufen. Stellen Sie sicher, dass Sie die orderByChild () - Methode hinzufügen. Ohne** diese Funktion wird dies nicht möglich sein. Firebase muss die Reihenfolge kennen, nach der Sie paginieren.

Firebase Echtzeitdatenbank online lesen: https://riptutorial.com/de/android/topic/5511/firebase-echtzeitdatenbank

Kapitel 96: Firebase-Absturzberichterstattung

Examples

So fügen Sie Firebase Crash Reporting zu Ihrer App hinzu

Führen Sie die folgenden Schritte aus, um *Firebase Crash Reporting* zu Ihrer App hinzuzufügen:

- Erstellen Sie eine App auf dem *Firebase Console* hier .
- Kopieren Sie die Datei `google-services.json` aus Ihrem Projekt in Ihr Verzeichnis in `app/` .
- Fügen Sie Ihrer *root.build.gradle* -Datei die folgenden Regeln hinzu, um das `google-services` Plugin aufzunehmen:

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.0.0'
    }
}
```

- Fügen Sie in Ihrer Modul-Gradle-Datei die `apply plugin` Zeile am Ende der Datei hinzu, um das Gradle-Plugin zu aktivieren:

```
apply plugin: 'com.google.gms.google-services'
```

- Fügen Sie der *Build.gradle*- Datei auf App-Ebene die Abhängigkeit für *Crash Reporting hinzu* :

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

- Sie können dann eine benutzerdefinierte Ausnahme aus Ihrer Anwendung auslösen, indem Sie die folgende Zeile verwenden:

```
FirebaseCrash.report(new Exception("Non Fatal Error logging"));
```

  Alle Ihre schwerwiegenden Ausnahmen werden an Ihre *Firebase-Konsole* gemeldet.

- Wenn Sie einer Konsole benutzerdefinierte Protokolle hinzufügen möchten, können Sie den folgenden Code verwenden:

```
FirebaseCrash.log("Level 2 completed.");
```

Für weitere Informationen, besuchen Sie bitte:

- Offizielle Dokumentation
- Thema "Stack Overflow"

Wie melde ich einen Fehler?

Firebase Crash Reporting generiert automatisch Berichte für schwerwiegende Fehler (oder nicht erfasste Ausnahmen).

Sie können Ihren benutzerdefinierten Bericht erstellen mit:

```
FirebaseCrash.report(new Exception("My first Android non-fatal error"));
```

Sie können das Protokoll einchecken, wenn FirebaseCrash das Modul initialisiert hat:

> 07–20 08: 57: 24.442 D / FirebaseCrashApiImpl: **API für FirebaseCrash-Berichterstellung initialisiert** 07–20 08: 57: 24.442 I / FirebaseCrash: **FirebaseCrash-Berichterstellung wird initialisiert.** 57: 24.442 D / FirebaseApp: **Klasse com.google.firebase.crash.FirebaseCrash wurde initialisiert.**

Und dann, wenn die Ausnahme gesendet wurde:

> 07–20 08: 57: 47.052 D / FirebaseCrashApiImpl: **throwable java.lang.Exception: Mein erster nicht schwerwiegender Android-Fehler** 07–20 08: 58: 18.822 D / FirebaseCrashSenderServiceImpl: Antwortcode **: 200** 07–20 08: 58: 18.822 D / FirebaseCrashSenderServiceImpl: **Bericht gesendet**

Sie können Ihrem Bericht benutzerdefinierte Protokolle hinzufügen

```
FirebaseCrash.log("Activity created");
```

Firebase-Absturzberichterstattung online lesen:
https://riptutorial.com/de/android/topic/5965/firebase-absturzberichterstattung

Kapitel 97: FloatingActionButton

Einführung

Die schwebende Aktionsschaltfläche wird für eine spezielle Art von Aktionen verwendet, die standardmäßig als expandierendes Material auf dem Bildschirm angezeigt wird. Das Symbol darin kann animiert sein, auch FAB kann sich aufgrund ihrer relativen Bedeutung anders als andere Elemente der Benutzeroberfläche bewegen. Eine schwebende Aktionsschaltfläche stellt die Hauptaktion in einer Anwendung dar, die einfach eine Aktion auslösen oder irgendwo navigieren kann.

Parameter

Parameter	Detail
`android.support.design:elevation`	Höhenwert für die FAB. Kann eine Referenz auf eine andere Ressource sein, in der Form "@ [+] [package:] type / name" oder ein Designattribut in der Form "? [Package:] type / name".
`android.support.design:fabSize`	Größe für die FAB.
`android.support.design:rippleColor`	Wellenfarbe für die FAB.
`android.support.design:useCompatPadding`	Compat-Auffüllung aktivieren

Bemerkungen

Schwimmende Aktionsschaltflächen werden für eine spezielle Art von Aktionsaktionen verwendet. Sie werden durch ein eingekreistes Symbol über der Benutzeroberfläche unterschieden und weisen ein spezielles Bewegungsverhalten auf, das sich auf das Morphing, das Starten und den Übergabepunkt bezieht.

Es wird nur eine Floating-Aktionstaste pro Bildschirm empfohlen, um die am häufigsten verwendete Aktion darzustellen.

Bevor Sie den `FloatingActionButton` , müssen Sie die Abhängigkeit der Entwurfsunterstützungsbibliothek in der Datei `build.gradle` :

```
dependencies {
    compile 'com.android.support:design:25.1.0'
}
```

Offizielle Dokumentation:

https://developer.android.com/reference/android/support/design/widget/FloatingActionButton.html

Material Design Spezifikationen:

https://material.google.com/components/buttons-floating-action-button.html

Examples

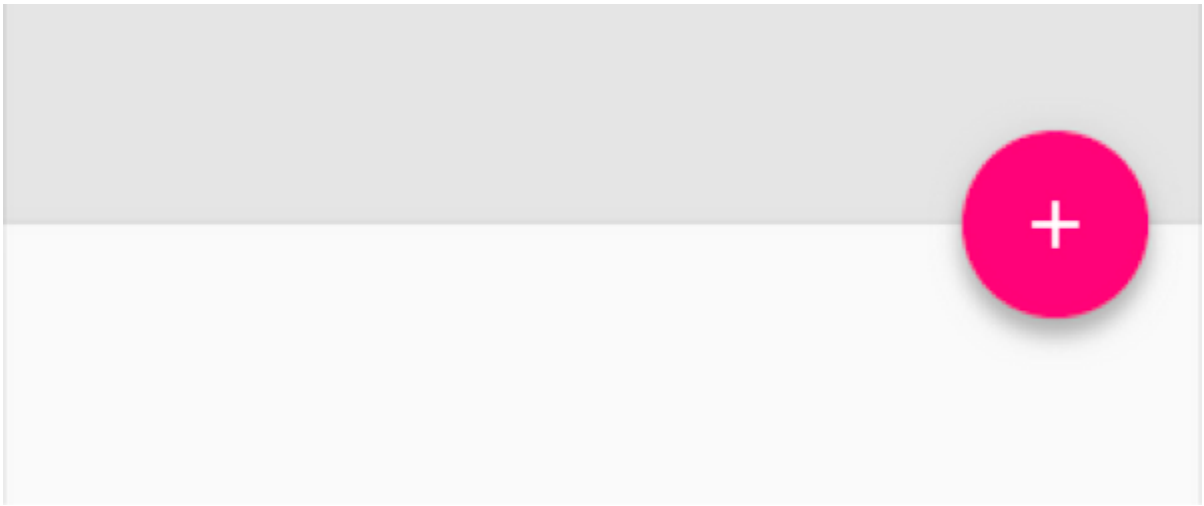
So fügen Sie die FAB zum Layout hinzu

Um einen FloatingActionButton zu verwenden, fügen Sie die Abhängigkeit in der `build.gradle` Datei hinzu, wie im Abschnitt "Anmerkungen" beschrieben.

Dann fügen Sie das Layout hinzu:

```
<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" />
```

Ein Beispiel:



Farbe

Die Hintergrundfarbe dieser Ansicht ist standardmäßig der colorAccent Ihres Themas.

Wenn das `src` im obigen Bild nur auf das Symbol + (standardmäßig 24x24 dp) zeigt, können Sie die *Hintergrundfarbe* des Vollkreises verwenden, `app:backgroundTint="@color/your_colour"` Sie `app:backgroundTint="@color/your_colour"`

Wenn Sie die Farbe im Code ändern möchten, können Sie

```
myFab.setBackgroundTintList(ColorStateList.valueOf(your color in int));
```

Wenn Sie die Farbe von FAB im gepressten Zustand ändern möchten, verwenden Sie

```
mFab.setRippleColor(your color in int);
```

Positionierung

Es wird empfohlen, mindestens 16 dB vom Rand des Mobilgeräts und mindestens 24 dB vom Tablet / Desktop aus zu platzieren.

Hinweis : Wenn Sie einen SRC eingestellt haben, mit Ausnahme, dass der gesamte Bereich von `FloatingActionButton` abgedeckt wird, stellen Sie sicher, dass Sie die richtige Größe des Bildes haben, um das beste Ergebnis zu erzielen.

Die Standardkreisgröße beträgt 56 x 56 dB

Minikreisgröße: 40 x 40 dp

Wenn Sie nur das Innensymbol ändern möchten, verwenden Sie ein 24 x 24-dB-Symbol für die Standardgröße

FloatingActionButton beim Swipe anzeigen und ausblenden

Um einen `FloatingActionButton` mit der Standardanimation ein- und auszublenden, rufen Sie einfach die Methoden `show()` und `hide()` . Es empfiehlt sich, einen `FloatingActionButton` im Aktivitätslayout zu belassen, anstatt ihn in ein Fragment zu setzen. Dadurch können die Standardanimationen beim Anzeigen und Ausblenden funktionieren.

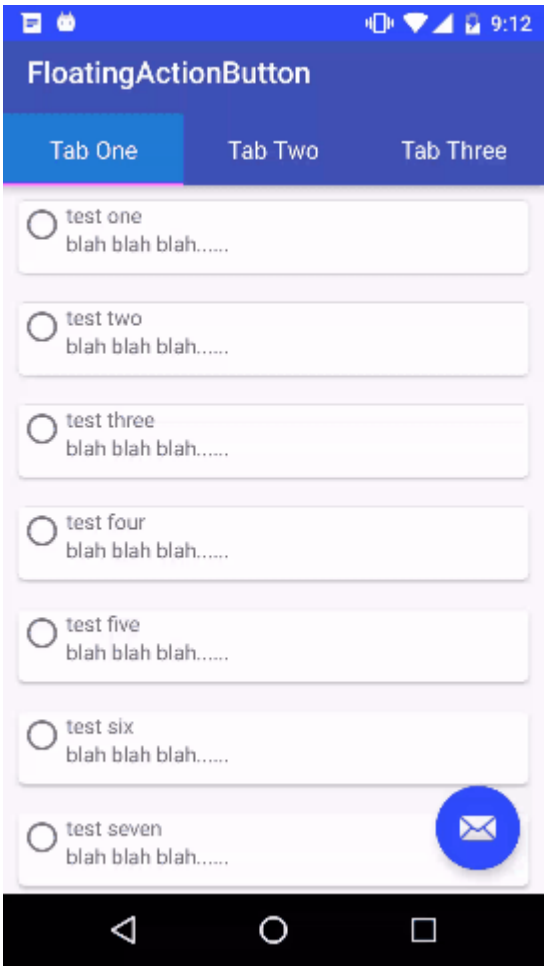
Hier ist ein Beispiel mit einem `ViewPager` :

- Drei Registerkarten
- `FloatingActionButton` für die erste und dritte Registerkarte anzeigen
- Blenden Sie den `FloatingActionButton` auf der mittleren Registerkarte aus

```
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();
                }
            }
        });

    }
}
```

Ergebnis:



FloatingActionButton beim Blättern anzeigen und ausblenden

Beginnend mit der Support Library Version 22.2.1 ist es möglich, ein FloatingActionButton-Verhalten mithilfe einer `FloatingActionButton.Behavior` Unterklasse, die die Methoden `show()` und `hide()` nutzt, vor dem Bildlaufverhalten anzuzeigen und `hide()` .

Beachten Sie, dass dies nur mit einem CoordinatorLayout in Verbindung mit inneren Ansichten funktioniert, die Nested Scrolling unterstützen, wie z. B. RecyclerView und NestedScrollView .

Diese `ScrollAwareFABBehavior` Klasse stammt aus den Android-Guides für Codepath (cc-wiki mit Attribution erforderlich).

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

`app:layout_behavior` in der FloatingActionButton-Layout-XML die `app:layout_behavior` mit dem vollständig qualifizierten Klassennamen von `ScrollAwareFABBehavior` :

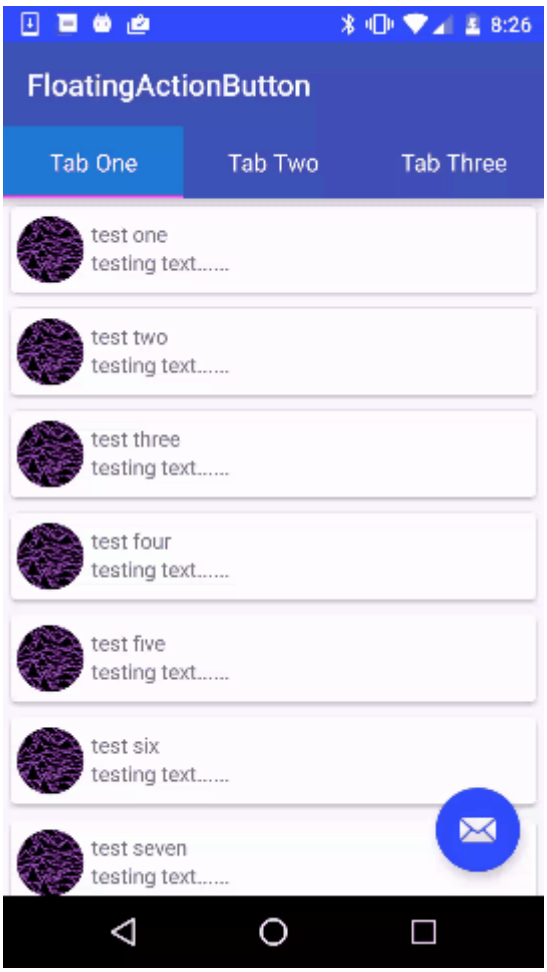
```
app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
```

Zum Beispiel mit diesem Layout:

```
<android.support.design.widget.CoordinatorLayout
    android:id="@+id/main_layout"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <android.support.design.widget.AppBarLayout
        android:id="@+id/appBarLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:elevation="6dp">
        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:background="?attr/colorPrimary"
            android:minHeight="?attr/actionBarSize"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
            app:elevation="0dp"
            app:layout_scrollFlags="scroll|enterAlways"
            />

        <android.support.design.widget.TabLayout
            android:id="@+id/tab_layout"
            app:tabMode="fixed"
            android:layout_below="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
            android:background="?attr/colorPrimary"
            app:elevation="0dp"
            app:tabTextColor="#d3d3d3"
            android:minHeight="?attr/actionBarSize"
            />

    </android.support.design.widget.AppBarLayout>

    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager"
        android:layout_below="@+id/tab_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
        android:layout_margin="@dimen/fab_margin"
        android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

Hier ist das Ergebnis:



Einstellungsverhalten von FloatingActionButton

Sie können das Verhalten der FAB in XML festlegen.

Zum Beispiel:

```
<android.support.design.widget.FloatingActionButton
   app:layout_behavior=".MyBehavior" />
```

Oder Sie können programmgesteuert einstellen mit:

```
CoordinatorLayout.LayoutParams p = (CoordinatorLayout.LayoutParams) fab.getLayoutParams();
p.setBehavior(xxxx);
fab.setLayoutParams(p);
```

FloatingActionButton online lesen: https://riptutorial.com/de/android/topic/2979/floatingactionbutton

Kapitel 98: Fortschrittsanzeige

Bemerkungen

Offizielle Dokumentation: ProgressBar

Examples

Unbestimmte Fortschrittsleiste

Eine unbestimmte Fortschrittsleiste zeigt eine zyklische Animation ohne Fortschrittsanzeige.

Grundlegende unbestimmte Fortschrittsleiste (Spinnrad)

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Horizontale unbestimmte Fortschrittsleiste (flache Leiste)

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Andere integrierte ProgressBar-Stile

```
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"
```

So verwenden Sie die unbestimmte Fortschrittsleiste in einer Aktivität

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setVisibility(View.VISIBLE);
progressBar.setVisibility(View.GONE);
```

Ermitteln Sie die Fortschrittsleiste

Eine bestimmte Fortschrittsleiste zeigt den aktuellen Fortschritt hin zu einem bestimmten Maximalwert.

Horizontale bestimmte Fortschrittsleiste

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="10dp"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Vertikal bestimmte Fortschrittsleiste

```
<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>
```

Ring bestimmt 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>
```

So verwenden Sie die bestimmte Fortschrittsleiste in einer Aktivität.

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setSecondaryProgress(100);
progressBar.setProgress(10);
progressBar.setMax(100);
```

Angepasste Fortschrittsleiste

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 :

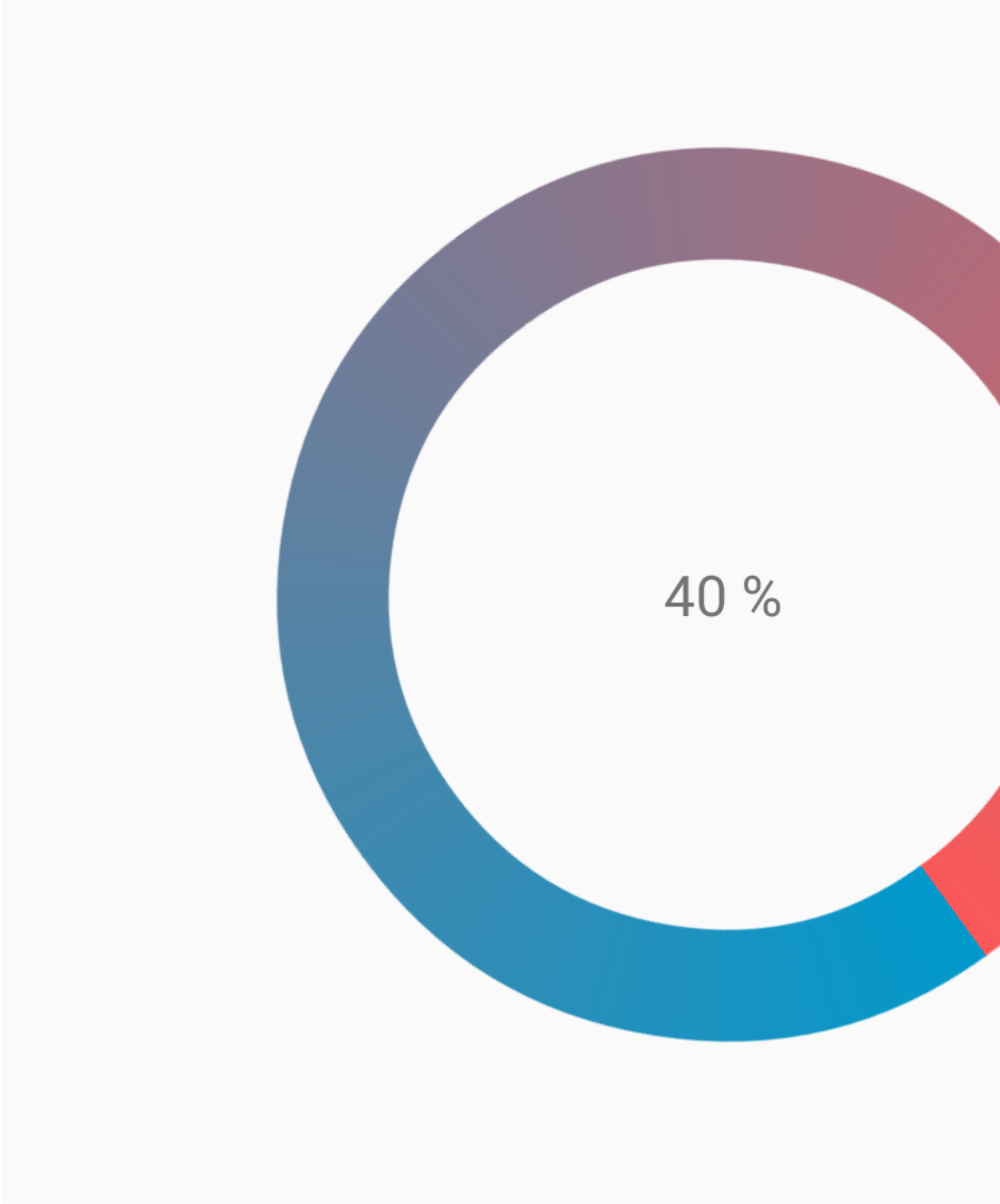
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    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>
```

Referenz-Screenshot:



Fortschrittsleiste tönen

Wenn Sie ein AppCompat-Design verwenden, wird die Farbe der `ProgressBar` das `colorAccent`

Ihnen definierte `colorAccent` sein.

5,0

Um die `ProgressBar` Farbe zu ändern, ohne die Akzentfarbe zu ändern, können Sie das Attribut `android:theme` , das die Akzentfarbe überschreibt:

```
<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>
```

Um die `ProgressBar` tönen, können Sie in der XML-Datei die Attribute `android:indeterminateTintMode` und `android:indeterminateTint`

```
<ProgressBar
    android:indeterminateTintMode="src_in"
    android:indeterminateTint="@color/my_color"
/>
```

Material Linear ProgressBar

Laut Materialdokumentation :

> Eine lineare Fortschrittsanzeige sollte immer von 0% bis 100% gefüllt sein und niemals an Wert verlieren.
> Es sollte durch Balken am Rand einer Kopfzeile oder eines Blattes dargestellt werden, die erscheinen und verschwinden.

Um ein lineares ProgressBar-Material zu verwenden, verwenden Sie es einfach in Ihrer XML-Datei:

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Unbestimmt

Um eine **unbestimmte** ProgressBar zu erstellen, setzen Sie das Attribut `android:indeterminate` auf `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"/>
```

Bestimmen

Um ein **bestimmtes** ProgressBar- `android:indeterminate` zu erstellen **,** setzen Sie das Attribut `android:indeterminate` auf `false` und verwenden Sie das Attribut `android:max` und das Attribut `android:progress` :

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"/>
```

Verwenden Sie einfach diesen Code, um den Wert zu aktualisieren:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
```

Puffer

Um einen **Puffereffekt** mit der ProgressBar zu erstellen, setzen Sie das Attribut `android:indeterminate` auf `false` und verwenden Sie die Attribute `android:max` , `android:progress` und `android:secondaryProgress` :

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"
    android:secondaryProgress="25"/>
```

Der Pufferwert wird vom Attribut `android:secondaryProgress` definiert.
Verwenden Sie einfach diesen Code, um die Werte zu aktualisieren:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
progressBar.setSecondaryProgress(50);
```

Unbestimmt und bestimmt

Um diese Art von ProgressBar zu erhalten, verwenden Sie einfach eine unbestimmte ProgressBar mit dem Attribut `android:indeterminate` auf true.

```
<ProgressBar
    android:id="@+id/progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="true"/>
```

Wenn Sie dann von einem unbestimmten zu einem bestimmten Fortschritt wechseln müssen, verwenden Sie die `setIndeterminate()` -Methode.

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setIndeterminate(false);
```

Dialogfeld "Benutzerdefinierter Fortschritt" erstellen

Durch Erstellen einer benutzerdefinierten Fortschrittsdialogklasse kann der Dialog in der Benutzeroberflächeninstanz angezeigt werden, ohne dass der Dialog neu erstellt werden muss.

Erstellen Sie zunächst eine benutzerdefinierte Fortschrittsdialogklasse.

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;
       }
```

```
   }
}
```

Erstellen Sie nun das benutzerdefinierte Fortschrittslayout

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>
```

Das ist es. Nun zum Aufruf des Dialogs im Code

```
CustomProgress customProgress = CustomProgress.getInstance();

// now you have the instance of CustomProgres
// for showing the ProgressBar

customProgress.showProgress(#Context, getString(#StringId), #boolean);

// for hiding the ProgressBar

customProgress.hideProgress();
```

Fortschrittsanzeige online lesen: https://riptutorial.com/de/android/topic/3353/fortschrittsanzeige

Kapitel 99: Fragmente

Einführung

Einführung in Fragmente und deren Interkommunikationsmechanismus

Syntax

- void onActivityCreated (Bundle savedInstanceState) // Wird aufgerufen, wenn die Aktivität des Fragments erstellt und die Ansichtshierarchie dieses Fragments instanziiert wurde.
- void onActivityResult (int requestCode, int resultCode, Intent data) // Empfangen Sie das Ergebnis eines vorherigen Aufrufs an startActivityForResult (Intent, int).
- void onAttach (Activity activity) // Diese Methode wurde auf API-Ebene 23 nicht mehr unterstützt. Verwenden Sie stattdessen onAttach (Context).
- void onAttach (Kontextkontext) // Wird aufgerufen, wenn ein Fragment zum ersten Mal mit seinem Kontext verbunden wird.
- void onAttachFragment (Fragment childFragment) // Wird aufgerufen, wenn ein Fragment als untergeordnetes Element dieses Fragments angehängt ist.
- void onConfigurationChanged (Configuration newConfig) // Wird vom System aufgerufen, wenn sich die Gerätekonfiguration ändert, während Ihre Komponente ausgeführt wird.
- void onCreate (Bundle savedInstanceState) // Wird aufgerufen, um ein Fragment zu erstellen.
- View onCreateView (LayoutInflater-Inflater, ViewGroup-Container, Bundle savedInstanceState) // Wird aufgerufen, damit das Fragment seine Benutzeroberflächenansicht instanziiert.
- void onDestroy () // Wird aufgerufen, wenn das Fragment nicht mehr verwendet wird.
- void onDestroyView () // Wird aufgerufen, wenn die zuvor von onCreateView (LayoutInflater, ViewGroup, Bundle) erstellte Ansicht vom Fragment getrennt wurde.
- void onDetach () // Wird aufgerufen, wenn das Fragment nicht mehr an seine Aktivität angehängt ist.
- void onInflate (Aktivitätsaktivität, AttributeSet-Attribute, Bundle savedInstanceState) // Diese Methode wurde in API-Ebene 23 nicht mehr unterstützt. Verwenden Sie stattdessen onInflate (Context, AttributeSet, Bundle).
- void onInflate (Context-Kontext, AttributeSet-Attribute, Bundle savedInstanceState) // Wird aufgerufen, wenn ein Fragment als Teil einer Inflation des Ansichtslayouts erstellt wird,

normalerweise durch Festlegen der Inhaltsansicht einer Aktivität.

- void onPause () // Wird aufgerufen, wenn das Fragment nicht mehr fortgesetzt wird.
- void onResume () // Wird aufgerufen, wenn das Fragment für den Benutzer sichtbar ist und aktiv ausgeführt wird.
- void onSaveInstanceState (Bundle outState) // Wird aufgerufen, um das Fragment zu bitten, seinen aktuellen dynamischen Status zu speichern, damit es später wiederhergestellt werden kann, wenn eine neue Instanz seines Prozesses neu gestartet wird.
- void onStart () // Wird aufgerufen, wenn das Fragment für den Benutzer sichtbar ist.
- void onStop () // Wird aufgerufen, wenn das Fragment nicht mehr gestartet wird.
- void onViewStateRestored (Bundle savedInstanceState) // Wird aufgerufen, wenn der gesamte gespeicherte Status in der Ansichtshierarchie des Fragments wiederhergestellt wurde.

Bemerkungen

Ein Fragment repräsentiert ein Verhalten oder einen Teil einer Benutzeroberfläche in einer Aktivität. Sie können mehrere Fragmente in einer einzigen Aktivität kombinieren, um eine Benutzeroberfläche mit mehreren Bereichen zu erstellen und ein Fragment in mehreren Aktivitäten wiederzuverwenden. Sie können sich ein Fragment als modularen Abschnitt einer Aktivität vorstellen, der seinen eigenen Lebenszyklus hat, seine eigenen Eingabeereignisse empfängt und die Sie hinzufügen oder entfernen können, während die Aktivität ausgeführt wird (eine Art "Unteraktivität", die Sie können Wiederverwendung in verschiedenen Aktivitäten).

Konstrukteur

Jedes Fragment muss über einen **leeren Konstruktor** verfügen, sodass es beim Wiederherstellen des Aktivitätszustands instanziiert werden kann. Es wird dringend empfohlen, dass Unterklassen keine anderen Konstruktoren mit Parametern haben, da diese Konstruktoren nicht aufgerufen werden, wenn das Fragment erneut instanziiert wird. Stattdessen können Argumente vom Aufrufer mit setArguments (Bundle) bereitgestellt und später vom Fragment mit getArguments () abgerufen werden.

Examples

Das newInstance () -Muster

Obwohl es möglich ist, einen Fragmentkonstruktor mit Parametern zu erstellen, ruft Android intern den Zero-Argument-Konstruktor auf, wenn Fragmente neu erstellt werden (z. B. wenn sie wiederhergestellt werden, nachdem sie aus eigenen Gründen von Android beendet wurden). Aus diesem Grund ist es nicht ratsam, sich auf einen Konstruktor mit Parametern zu verlassen.

Um sicherzustellen, dass die erwarteten Fragmentargumente immer vorhanden sind, können Sie das Fragment mit einer statischen `newInstance()` Methode erstellen und die gewünschten Parameter in ein Bundle `newInstance()` , das beim Erstellen einer neuen Instanz verfügbar ist.

```
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);
    }
  }
}
```

Jetzt in der Aktivität:

```
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();
```

Dieses Muster ist eine bewährte Methode, um sicherzustellen, dass alle erforderlichen Argumente

bei der Erstellung an Fragmente übergeben werden. Wenn das System das Fragment zerstört und es später erneut erstellt, wird der Status automatisch wiederhergestellt. Sie müssen es jedoch mit einer `onSaveInstanceState(Bundle)` Implementierung `onSaveInstanceState(Bundle)` .

Navigation zwischen Fragmenten mit Backstack und statischem Gewebemuster

Zuallererst müssen wir unser erstes `Fragment` am Anfang hinzufügen, wir sollten es in der `onCreate()` -Methode unserer Activity tun:

```
if (null == savedInstanceState) {
    getSupportFragmentManager().beginTransaction()
      .addToBackStack("fragmentA")
      .replace(R.id.container, FragmentA.newInstance(), "fragmentA")
      .commit();
}
```

Als Nächstes müssen wir unseren Backstack verwalten. Am einfachsten verwenden Sie eine in unserer Aktivität hinzugefügte Funktion, die für alle FragmentTransactions verwendet wird.

```
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();
}
```

Schließlich sollten wir `onBackPressed()` überschreiben, um die Anwendung zu `onBackPressed()` wenn Sie vom letzten im Backstack verfügbaren Fragment zurückkehren.

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

Ausführung in Tätigkeit:

```
replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Ausführung außerhalb der Aktivität (unter der Annahme, dass `MainActivity` unsere Aktivität ist):

```
((MainActivity) getActivity()).replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Übergeben Sie Daten mithilfe von Bundle von Activity an Fragment

Alle Fragmente sollten einen leeren Konstruktor haben (dh eine Konstruktormethode ohne Eingabeargumente). Um Ihre Daten an das erstellte Fragment zu übergeben, sollten Sie daher die Methode `setArguments()` verwenden. Diese Methode erhält ein Bündel, in dem Sie Ihre Daten speichern, und speichert das Bündel in den Argumenten. Anschließend kann dieses Bundle in `onCreate()` und `onCreateView()` Rückrufen des Fragments abgerufen werden.

Aktivität:

```
 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();
```

Fragment:

```
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    String myValue = this.getArguments().getString("message");
    ...
 }
```

Ereignisse zurück an eine Aktivität mit Callback-Schnittstelle senden

Wenn Sie Ereignisse vom Fragment an die Aktivität senden müssen, besteht eine der möglichen Lösungen darin, die Callback-Schnittstelle zu definieren und von der Hostaktivität zu implementieren.

Beispiel

Senden Sie einen Rückruf an eine Aktivität, wenn Sie auf die

Schaltfläche des Fragments klicken

Definieren Sie zunächst die Callback-Schnittstelle:

```
public interface SampleCallback {
    void onButtonClicked();
}
```

Der nächste Schritt besteht darin, diesen Rückruf in Fragment zuzuweisen:

```
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;
    }
}
```

Und schließlich implementiere Rückruf in Aktivität:

```
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
    }
}
```

Animieren Sie den Übergang zwischen Fragmenten

Um den Übergang zwischen Fragmenten oder das Anzeigen oder Ausblenden eines Fragments zu animieren, verwenden Sie den `FragmentManager` , um eine `FragmentTransaction` zu erstellen.

Für eine einzelne `FragmentTransaction` gibt es zwei verschiedene Arten, Animationen auszuführen: Sie können eine Standardanimation verwenden oder eigene benutzerdefinierte Animationen bereitstellen.

Standardanimationen werden durch Aufrufen von `FragmentTransaction.setTransition(int transit)` und Verwenden einer der vordefinierten Konstanten in der `FragmentTransaction` Klasse angegeben. Zum Zeitpunkt des Schreibens sind diese Konstanten:

```
FragmentTransaction.TRANSIT_NONE
FragmentTransaction.TRANSIT_FRAGMENT_OPEN
FragmentTransaction.TRANSIT_FRAGMENT_CLOSE
FragmentTransaction.TRANSIT_FRAGMENT_FADE
```

Die vollständige Transaktion könnte ungefähr so aussehen:

```
getSupportFragmentManager()
    .beginTransaction()
    .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

Benutzerdefinierte Animationen werden durch Aufrufen von `FragmentTransaction.setCustomAnimations(int enter, int exit)` oder `FragmentTransaction.setCustomAnimations(int enter, int exit, int popEnter, int popExit)` .

Die `enter` und `exit` Animationen für abgespielt `FragmentTransaction` s, die nicht mit Fragmenten aus den hinteren Stapeln knallen. Die `popEnter` und `popExit` Animationen werden abgespielt, wenn Sie ein Fragment aus dem Back-Stack ziehen.

Der folgende Code zeigt, wie Sie ein Fragment ersetzen, indem Sie ein Fragment herausschieben und das andere an seiner Stelle verschieben.

```
getSupportFragmentManager()
    .beginTransaction()
    .setCustomAnimations(R.anim.slide_in_left, R.anim.slide_out_right)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

Die XML-Animationsdefinitionen verwenden das `objectAnimator` Tag. Ein Beispiel für **slide_in_left.xml** könnte **ungefähr so** aussehen:

```
<?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>
```

Kommunikation zwischen Fragmenten

Alle Kommunikationen zwischen Fragmenten müssen über eine Aktivität erfolgen. Fragmente können **NICHT** ohne Aktivität miteinander kommunizieren.

Zusätzliche Ressourcen

- So implementieren Sie OnFragmentInteractionListener
- Android | Kommunikation mit anderen Fragmenten

In diesem Beispiel haben wir eine `MainActivity` , die zwei Fragmente, `SenderFragment` und `ReceiverFragment` , zum Senden und Empfangen einer `message` (in diesem Fall ein einfacher String) hostet.

Ein Button in `SenderFragment` initiiert das Senden der Nachricht. Eine TextView im `ReceiverFragment` wird aktualisiert, wenn die Nachricht von ihr empfangen wird.

Es folgt ein Ausschnitt für die MainActivity mit Kommentaren zu den wichtigen Codezeilen:

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

Die Layoutdatei für `MainActivity` zwei Fragmente in einem LinearLayout:

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

<fragment
    android:id="@+id/fragment_sender"
    android:name="com.naru.fragmentcommunication.SenderFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    tools:layout="@layout/fragment_sender" />

<fragment
    android:id="@+id/fragment_receiver"
    android:name="com.naru.fragmentcommunication.ReceiverFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    tools:layout="@layout/fragment_receiver" />
</LinearLayout>
```

Das `SenderFragment` macht eine Schnittstelle `SendMessageListener` , mit der die `MainActivity` , wann auf die `SenderFragment` im `SenderFragment` geklickt wurde.

Das folgende Codefragment für `SenderFragment` die wichtigsten Codezeilen:

```
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;
}
```

```
}
```

Die Layoutdatei für das `SenderFragment` :

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

<Button
    android:id="@+id/bSend"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="SEND"
    android:layout_gravity="center_horizontal" />
</LinearLayout>
```

`ReceiverFragment` ist einfach und stellt eine einfache öffentliche Methode zum Aktualisieren seiner TextView bereit. Wenn die `MainActivity` die Nachricht von dem erhält `SenderFragment` es nennt diese öffentliche Methode des `ReceiverFragment`

Im Folgenden finden Sie das Code-Snippet für das `ReceiverFragment` mit Kommentaren zu den wichtigen Codezeilen:

```
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);
}
}
```

Die Layoutdatei für das `ReceiverFragment` :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">
<TextView
    android:id="@+id/tvReceivedMessage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Waiting for message!" />
</LinearLayout>
```

Fragmente online lesen: https://riptutorial.com/de/android/topic/1396/fragmente

Kapitel 100: Freigegebene Elementübergänge

Einführung

Hier finden Sie Beispiele für den Übergang zwischen `Activities` oder `Fragments` mit einem gemeinsamen Element. Ein Beispiel für dieses Verhalten ist die Google Play Store App, die das Symbol einer App aus der Liste in die Detailansicht der App übersetzt.

Syntax

- transaction.addSharedElement (sharedElementView, "targetTransitionName");
- fragment.setSharedElementEnterTransition (new CustomTransaction ());

Examples

Gemeinsamer Elementübergang zwischen zwei Fragmenten

In diesem Beispiel sollte eine von zwei verschiedenen `ImageViews` vom `ChooserFragment` in das `DetailFragment` .

Im `ChooserFragment` Layout benötigen wir die eindeutigen `transitionName` Attribute:

```
<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" />
```

In der `ChooserFragments` Klasse müssen wir die angeklickte `View` und eine ID an die übergeordnete `Activity` , die den Austausch der Fragmente `DetailFragment` (wir benötigen die ID, um zu wissen, welche `DetailFragment` im `DetailFragment`). Wie Sie Informationen an eine übergeordnete Aktivität im Detail weitergeben, wird in einer anderen Dokumentation beschrieben.

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

In `DetailFragment` die `ImageView` des gemeinsam genutzten Elements auch das eindeutige Attribut `transitionName` .

```
<ImageView
    android:id="@+id/image_shared"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:transitionName="sharedImage" />
```

In der `onCreateView()` -Methode von `DetailFragment` müssen wir entscheiden, welche `DetailFragment` angezeigt werden soll (wenn dies nicht der `DetailFragment` ist, wird das gemeinsam genutzte Element nach dem Übergang nicht mehr angezeigt).

```
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;
}
```

Die übergeordnete `Activity` empfängt die Rückrufe und übernimmt das Ersetzen der Fragmente.

```
@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();
}
```

Nicht zu vergessen - der `Transition` selbst. In diesem Beispiel wird das gemeinsam genutzte Element verschoben und skaliert.

```
@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class DetailsTransition extends TransitionSet {

    public DetailsTransition() {
        setOrdering(ORDERING_TOGETHER);
        addTransition(new ChangeBounds()).
            addTransition(new ChangeTransform()).
            addTransition(new ChangeImageTransform());
    }

}
```

Freigegebene Elementübergänge online lesen:
https://riptutorial.com/de/android/topic/8933/freigegebene-elementubergange

Kapitel 101: Fresko

Einführung

Fresco ist ein leistungsfähiges System zum Anzeigen von Bildern in Android-Anwendungen.

In Android 4.x und niedriger legt Fresco Bilder in einem speziellen Bereich des **Android-Speichers** (Ashmem) ab. Dadurch läuft Ihre Anwendung schneller - und erleidet den gefürchteten OutOfMemoryError seltener.

Fresco unterstützt auch das Streaming von JPEGs.

Bemerkungen

So richten Sie Abhängigkeiten in der Build.gradle-Datei auf App-Ebene ein:

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

Weitere Informationen finden Sie hier .

Examples

Erste Schritte mit Fresco

`build.gradle` Sie zunächst Fresco zu Ihrem `build.gradle` wie im Abschnitt "Anmerkungen" gezeigt:

Wenn Sie zusätzliche Funktionen benötigen, beispielsweise animierte GIF- oder WebP-Unterstützung, müssen Sie auch die entsprechenden Fresco-Artefakte hinzufügen.

Fresko muss initialisiert werden. Sie sollten dies nur einmal tun, daher ist es eine gute Idee, die Initialisierung in Ihrer `Application` . Ein Beispiel dafür wäre:

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

Wenn Sie Remote-Images von einem Server laden möchten, benötigt Ihre App die Internt-Berechtigung. `AndroidManifest.xml` Sie es einfach zu Ihrem `AndroidManifest.xml` :

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

`SimpleDraweeView` dann Ihrem XML-Layout eine `SimpleDraweeView` . Fresco unterstützt `wrap_content` für Bildmaße, da Sie möglicherweise mehrere Bilder mit unterschiedlichen Abmessungen (Platzhalterbild, `wrap_content` , aktuelles Bild, ...) haben.

Sie können also entweder eine `SimpleDraweeView` mit festen Abmessungen hinzufügen (oder `match_parent`):

```
<com.facebook.drawee.view.SimpleDraweeView
    android:id="@+id/my_image_view"
    android:layout_width="120dp"
    android:layout_height="120dp"
    fresco:placeholderImage="@drawable/placeholder" />
```

Oder geben Sie ein *Bildverhältnis* für Ihr Bild an:

```
<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" />
```

Zum Schluss können Sie Ihren Image-URI in Java festlegen:

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

Das ist es! Sie sollten Ihren Platzhalter so lange zeichnen, bis das Netzwerkimage abgerufen wurde.

OkHttp 3 mit Fresco verwenden

Zuerst müssen Sie zusätzlich zu der normalen Abhängigkeit von Fresco Gradle die Abhängigkeit von OkHttp 3 zu Ihrem `build.gradle` :

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

Wenn Sie Fresco initialisieren (normalerweise in Ihrer benutzerdefinierten `Application`), können Sie jetzt Ihren OkHttp-Client angeben:

```
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);
```

JPEG-Streaming mit Fresco mit DraweeController

In diesem Beispiel wird davon ausgegangen, dass Sie Ihrer App bereits Fresco hinzugefügt haben (siehe folgendes Beispiel):

```
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.
```

Fresko online lesen: https://riptutorial.com/de/android/topic/5217/fresko

Kapitel 102: FuseView zu einem Android-Projekt hinzufügen

Einführung

Exportieren Sie eine Fuse.View von fusetools und verwenden Sie sie in einem vorhandenen Android-Projekt.

Unser Ziel ist es, die gesamte Hikr-Beispiel-App zu exportieren und innerhalb einer `Activity` .

Die letzte Arbeit findet sich unter lucamtudor / hikr-fuse-view

Examples

hikr app, nur eine andere android.view.View

Voraussetzungen

- Sie sollten eine Sicherung installiert haben (https://www.fusetools.com/downloads)
- Sie sollten das Einführungstutorial durchgeführt haben
- in terminal: `fuse install android`
- im terminal: `uno install Fuse.Views`

Schritt 1

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

Schritt 2 : Fügen Sie der `Fuse.Views`

Suchen `hikr.unoproj` Datei `hikr.unoproj` im `hikr.unoproj` des Projekts und fügen `"Fuse.Views"` dem Array `"Packages" "Fuse.Views"` .

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

Schritt 3 : `HikrApp` Sie eine `HikrApp` Komponente für die gesamte App

3.1 `HikrApp.ux` im Projektordner eine neue Datei mit dem Namen `HikrApp.ux` und fügen Sie den Inhalt von `MainView.ux` .

HikrApp.ux

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

    <Router ux:Name="router" />

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

3.2 In `HikrApp.ux`

- Ersetzen Sie die `<App>` -Tags durch `<Page>`
- add `ux:Class="HikrApp"` zur Eröffnung `<Page>`
- `<ClientPanel>` entfernen, müssen Sie sich nicht mehr um die Statusleiste oder die unteren `<ClientPanel>` kümmern

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 Verwenden Sie die neu erstellte `HikrApp` Komponente in `MainView.ux`

Ersetzen Sie den Inhalt der Datei `MainView.ux` durch:

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

Unsere App ist wieder normal, aber wir haben sie jetzt in eine separate Komponente namens `HikrApp`

Schritt 4 Ersetzen Sie in `MainView.ux` die `<App>` -Tags durch `<ExportedViews>` und fügen Sie `ux:Template="HikrAppView"` zu `<HikrApp />`

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

Denken Sie an die Vorlage `HikrAppView` , denn wir benötigen sie, um einen Verweis auf unsere Ansicht von Java zu erhalten.

Hinweis Aus den Sicherungsdokumenten:

> `ExportedViews` verhält sich wie eine `App` wenn eine normale `fuse preview` und ein `uno build`

Nicht wahr. Diese Fehlermeldung wird bei der Vorschau von Fuse Studio angezeigt:

> Fehler: In keiner der enthaltenen UX-Dateien konnte ein App-Tag gefunden werden. Haben Sie vergessen, die UX-Datei mit dem App-Tag einzuschließen?

Schritt 5 `SplashPage.ux` das `<DockPanel>` in eine `<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>
```

Schritt 6 Exportieren Sie das Sicherungsprojekt als AAR-Bibliothek

- im Terminal im Stammprojektordner: `uno clean`
- im Terminal im `uno build -t=android -DLIBRARY` : `uno build -t=android -DLIBRARY`

Schritt 7 Bereiten Sie Ihr Android-Projekt vor

- Kopieren Sie das Archiv von `.../rootHikeProject/build/Android/Debug/app/build/outputs/aar/app-debug.aar` nach `.../androidRootProject/app/libs`
- `flatDir { dirs 'libs' }` Sie `flatDir { dirs 'libs' }` der root-Datei `build.gradle`

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

buildscript { ... }

...

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

...
```

- Fügen Sie `compile(name: 'app-debug', ext: 'aar')` den Abhängigkeiten in `app/build.gradle`

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

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

```
    }
}

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

- `AndroidManifest.xml` der Aktivität in `AndroidManifest.xml` die folgenden Eigenschaften `AndroidManifest.xml`

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

Ihre `AndroidManifest.xml` wird folgendermaßen aussehen:

```
<?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>
```

Schritt 8 : Zeigen Sie die `Fuse.View HikrAppView` in Ihrer `Activity`

- Beachten Sie, dass Ihre `Activity FuseViewsActivity` erben `FuseViewsActivity`

```
public class MainActivity extends FuseViewsActivity {

    @Override
```

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

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

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

activity_main.xml

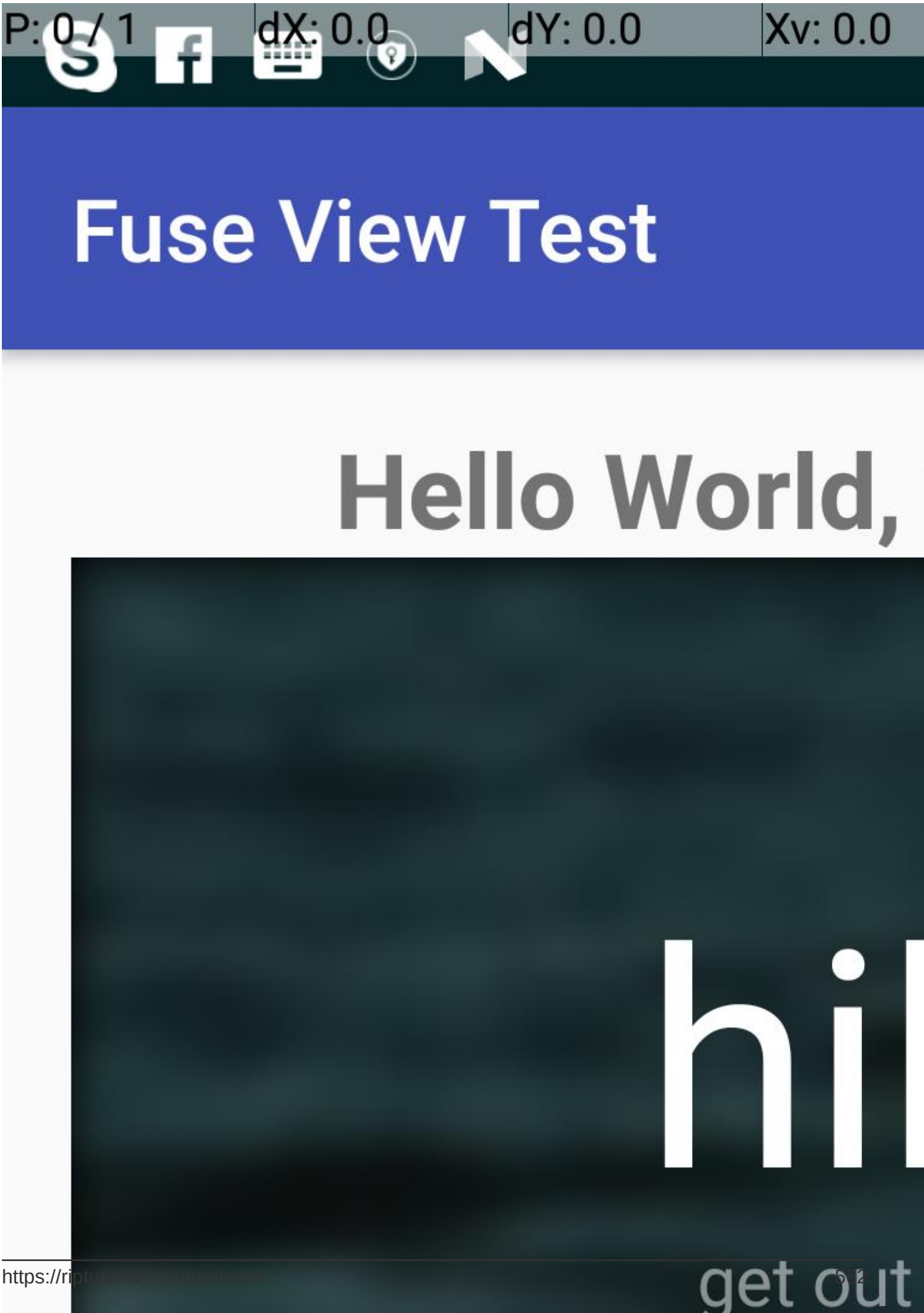
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"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

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

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

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

    </FrameLayout>

</LinearLayout>
```

Hinweis
Wenn Sie bei Android die Zurück-Taste drücken, stürzt die App ab. Sie können das Problem im

Sicherungsforum verfolgen.

```
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
```

Und das Endergebnis ist so etwas. Sie können auch einen kurzen Clip auf Github finden .


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

https://riptutorial.com/de/android/topic/10052/fuseview-zu-einem-android-projekt-hinzufugen

Kapitel 103: Fusselwarnungen

Bemerkungen

Das **Lint-Tool** überprüft die Quelldateien Ihres Android-Projekts auf potenzielle Fehler und Optimierungsverbesserungen auf Korrektheit, Sicherheit, Leistung, Benutzerfreundlichkeit, Zugänglichkeit und Internationalisierung. Sie können Lint über die Befehlszeile oder über Android Studio ausführen.

Offizielle Dokumentation:

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

Examples

Verwenden von Tools: In XML-Dateien ignorieren

Die Attributtools `tools:ignore` können in XML-Dateien verwendet werden, um Flusenwarnungen zu löschen.

ABER das Entfernen von Fusselwarnungen mit dieser Technik ist meistens der falsche Weg.

Eine Fusselwarnung muss verstanden und behoben werden. Sie kann nur dann ignoriert werden, wenn Sie deren Bedeutung vollständig verstanden haben und einen guten Grund haben, sie zu ignorieren.

Hier ist ein Anwendungsfall, in dem es legitim ist, eine Fusselwarnung zu ignorieren:

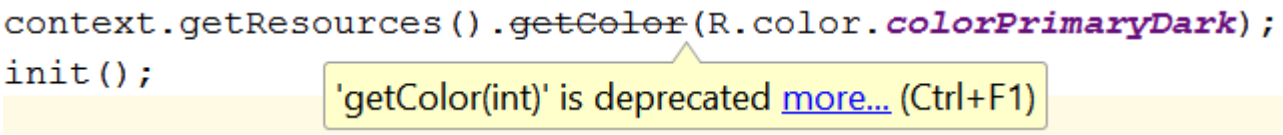
- Sie entwickeln eine System-App (signiert mit dem Geräteherstellerschlüssel)
- Ihre App muss das Gerätedatum (oder eine andere geschützte Aktion) ändern.

Dann können Sie dies in Ihrem Manifest tun: (dh die geschützte Berechtigung wird angefordert und die Fusselwarnung wird ignoriert, weil Sie wissen, dass in Ihrem Fall die Berechtigung erteilt wird).

```
<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"/>
```

Importieren von Ressourcen ohne Fehler "Veraltet"

Bei Verwendung der Android-API 23 oder höher ist eine solche Situation häufig zu sehen:



Diese Situation wird durch die strukturelle Änderung der Android-API in Bezug auf das Abrufen der Ressourcen verursacht. Nun die Funktion:

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

sollte benutzt werden. Die android.support.v4-Bibliothek hat jedoch eine andere Lösung.

Fügen Sie der Datei build.gradle die folgende Abhängigkeit hinzu:

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

Dann stehen alle Methoden aus der Support Library zur Verfügung:

```
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));
```

Darüber hinaus können weitere Methoden aus der Support-Bibliothek verwendet werden:

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

Konfigurieren Sie LintOptions mit Gradle

Sie können Flusen konfigurieren, indem `lintOptions` in der Datei `lintOptions` Abschnitt `build.gradle` :

```
android {

    //.....

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

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

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

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

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

```
        abortOnError false

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

Sie können Flusen für eine bestimmte Variante (siehe unten) `./gradlew lintRelease` , z. B. `./gradlew lintRelease` , oder für alle Varianten (`./gradlew lint`). In diesem Fall wird ein Bericht erstellt, in dem beschrieben wird, für welche bestimmten Varianten ein bestimmtes Problem gilt.

Hier finden Sie die DSL-Referenz für alle verfügbaren Optionen .

So konfigurieren Sie die Datei lint.xml

In der Datei `lint.xml` können Sie Ihre Voreinstellungen für die Lint-Prüfung `lint.xml` . Wenn Sie diese Datei manuell erstellen, platzieren Sie sie im Stammverzeichnis Ihres Android-Projekts. Wenn Sie die Lint-Einstellungen in Android Studio konfigurieren, wird die Datei lint.xml automatisch erstellt und Ihrem Android-Projekt für Sie hinzugefügt.

Beispiel:

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

Durch Festlegen des Attributs für den Schweregrad im Tag können Sie die Lint-Prüfung für ein Problem deaktivieren oder den Schweregrad für ein Problem ändern.

Das folgende Beispiel zeigt den Inhalt einer Datei `lint.xml` .

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

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

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

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

Konfigurieren der Fusselprüfung in Java- und XML-Quelldateien

Sie können die Lint-Prüfung in Ihren Java- und XML-Quelldateien deaktivieren.

Fusselprüfung in Java konfigurieren

Um die Lint-Überprüfung speziell für eine **Java-Klasse** oder -Methode in Ihrem Android-Projekt zu deaktivieren, fügen `@SuppressLint` diesem Java-Code die **Annotation** `@SuppressLint` .

Beispiel:

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

So deaktivieren Sie die Überprüfung für alle Lint-Probleme:

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

Konfigurieren der Fusselprüfung in XML

Sie können das `tools:ignore` Attribut verwenden, um die Lint-Prüfung für bestimmte Abschnitte Ihrer **XML-Dateien** zu deaktivieren.

Zum Beispiel:

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

Verwenden Sie, um die Überprüfung auf alle Lint-Probleme im XML-Element zu unterdrücken

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

Mark Unterdrückung von Warnungen

Es empfiehlt sich, einige Warnungen in Ihrem Code zu markieren. Beispielsweise sind einige veraltete Methoden für Ihre Tests oder die alte Supportversion erforderlich. Bei der Flusenüberprüfung wird der Code jedoch mit Warnungen markiert. Um dieses Problem zu vermeiden, müssen Sie die Annotation @SuppressWarnings verwenden.

Fügen Sie zum Beispiel Ignorieren zu Warnungen zu veralteten Methoden hinzu. Sie müssen auch die Beschreibung der Warnungen in die Anmerkung einfügen:

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

Mit dieser Anmerkung können Sie alle Warnungen, einschließlich Lint, Android und andere, ignorieren. Die Verwendung von Unterdrückungswarnungen hilft, Code richtig zu verstehen!

Fusselwarnungen online lesen: https://riptutorial.com/de/android/topic/129/fusselwarnungen

Kapitel 104: Gemeinsame Einstellungen

Einführung

SharedPreferences bieten eine Möglichkeit, Daten in Form von **Schlüsselwertpaaren** auf der Festplatte zu speichern.

Syntax

- **Kontextmethode**
  - public SharedPreferences getSharedPreferences (Stringname, int-Modus)
- **Aktivitätsmethode**
  - public SharedPreferences getPreferences ()
- **SharedPreferences-Methoden**
  - public SharedPreferences.Editor edit ()
  - public boolean enthält ()
  - public Map <String,?> getAll ()
  - public boolean getBoolean (String-Schlüssel, boolean defValue)
  - public float getFloat (String-Schlüssel, float defValue)
  - public int getInt (String-Schlüssel, int defValue)
  - public long getLong (String key, long defValue)
  - public String getString (String-Schlüssel, String defValue)
  - public Set getStringSet (String-Schlüssel, DefValues setzen)
  - public void registerOnSharedPreferenceChangeListener (SharedPreferences.OnSharedPreferenceChangeListener-Listener)
  - public void unregisterOnSharedPreferenceChangeListener (SharedPreferences.OnSharedPreferenceChangeListener-Listener)
- **SharedPreferences.Editor-Methoden**
  - öffentlich ungültig gelten ()
  - Öffentliches boolesches Commit ()
  - public SharedPreferences.Editor clear ()
  - public SharedPreferences.Editor putBoolean (String-Schlüssel, boolescher Wert)
  - public SharedPreferences.Editor putFloat (String-Schlüssel, Gleitkommawert)
  - public SharedPreferences.Editor putInt (String-Schlüssel, int-Wert)
  - public SharedPreferences.Editor putLong (String-Schlüssel, langer Wert)
  - public SharedPreferences.Editor putString (String-Schlüssel, String-Wert)
  - public SharedPreferences.Editor putStringSet (String-Schlüssel, Werte setzen)
  - public SharedPreferences.Editor remove (String-Schlüssel)

Parameter

Parameter	Einzelheiten
Schlüssel	Eine nicht leere `String` , die den Parameter identifiziert. Es kann Leerzeichen oder nicht druckbare Dateien enthalten. Dies wird nur in Ihrer App (und in der XML-Datei) verwendet und muss daher nicht mit einem Namensbereich versehen werden. Es ist jedoch eine gute Idee, es als Konstante im Quellcode zu haben. Lokalisieren Sie es nicht.
defValue	Alle get-Funktionen erhalten einen Standardwert, der zurückgegeben wird, wenn der angegebene Schlüssel nicht in den `SharedPreferences` . Es wird nicht zurückgegeben, wenn der Schlüssel vorhanden ist, aber der Wert hat den falschen Typ: In diesem Fall erhalten Sie eine `ClassCastException` .

Bemerkungen

- `SharedPreferences` sollten nicht zum Speichern großer Datenmengen verwendet werden. Für solche Zwecke ist es viel besser, `SQLiteDatabase` zu verwenden.
- `SharedPreferences` sind nur ein Prozess, es sei denn, Sie verwenden den veralteten Modus `MODE_MULTI_PROCESS` . Wenn Ihre App also über mehrere Prozesse verfügt, können Sie die `SharedPreferences` des `SharedPreferences` in einem anderen Prozess lesen. In solchen Fällen sollten Sie einen anderen Mechanismus zum `MODE_MULTI_PROCESS` Daten zwischen Prozessen verwenden, `MODE_MULTI_PROCESS` jedoch nicht verwenden, da dies nicht zuverlässig und nicht mehr veraltet ist.
- Es ist besser , verwenden `SharedPreferences` Instanz in `Singleton` Klasse überall auf die Anwendung zugreifen `context` . Wenn Sie es nur für bestimmte Aktivitäten verwenden möchten, gehen Sie zu `getPreferences()` .
- Speichern Sie vertrauliche Informationen bei der Verwendung von `SharedPreferences` im Klartext, da sie leicht lesbar sind.

Offizielle Dokumentation

https://developer.android.com/reference/android/content/SharedPreferences.html

Examples

Lesen und Schreiben von Werten in SharedPreferences

```
public class MyActivity extends Activity {

    private static final String PREFS_FILE = "NameOfYourPrefrenceFile";
```

```
    // PREFS_MODE defines which apps can access the file
    private static final int PREFS_MODE = Context.MODE_PRIVATE;
    // you can use live template "key" for quickly creating keys
    private static final String KEY_BOOLEAN = "KEY_FOR_YOUR_BOOLEAN";
    private static final String KEY_STRING = "KEY_FOR_YOUR_STRING";
    private static final String KEY_FLOAT = "KEY_FOR_YOUR_FLOAT";
    private static final String KEY_INT = "KEY_FOR_YOUR_INT";
    private static final String KEY_LONG = "KEY_FOR_YOUR_LONG";

    @Override
    protected void onStart() {
        super.onStart();

        // Get the saved flag (or default value if it hasn't been saved yet)
        SharedPreferences settings = getSharedPreferences(PREFS_FILE, PREFS_MODE);
        // read a boolean value (default false)
        boolean booleanVal = settings.getBoolean(KEY_BOOLEAN, false);
        // read an int value (Default 0)
        int intVal = settings.getInt(KEY_INT, 0);
        // read a string value (default "my string")
        String str = settings.getString(KEY_STRING, "my string");
        // read a long value (default 123456)
        long longVal = settings.getLong(KEY_LONG, 123456);
        // read a float value (default 3.14f)
        float floatVal = settings.getFloat(KEY_FLOAT, 3.14f);
    }

    @Override
    protected void onStop() {
        super.onStop();

        // Save the flag
        SharedPreferences settings = getSharedPreferences(PREFS_FILE, PREFS_MODE);
        SharedPreferences.Editor editor = settings.edit();
        // write a boolean value
        editor.putBoolean(KEY_BOOLEAN, true);
        // write an integer value
        editor.putInt(KEY_INT, 123);
        // write a string
        editor.putString(KEY_STRING, "string value");
        // write a long value
        editor.putLong(KEY_LONG, 456876451);
        // write a float value
        editor.putFloat(KEY_FLOAT, 1.51f);
        editor.apply();
    }
}
```

`getSharedPreferences()` ist eine Methode aus der `Context` Klasse, die `Activity` erweitert. Wenn Sie die zugreifen müssen `getSharedPreferences()` Methode aus anderen Klassen, können Sie `context.getSharedPreferences()` mit einem `Context` Objektverweis von einer `Activity` , `View` oder `Application` .

Schlüssel entfernen

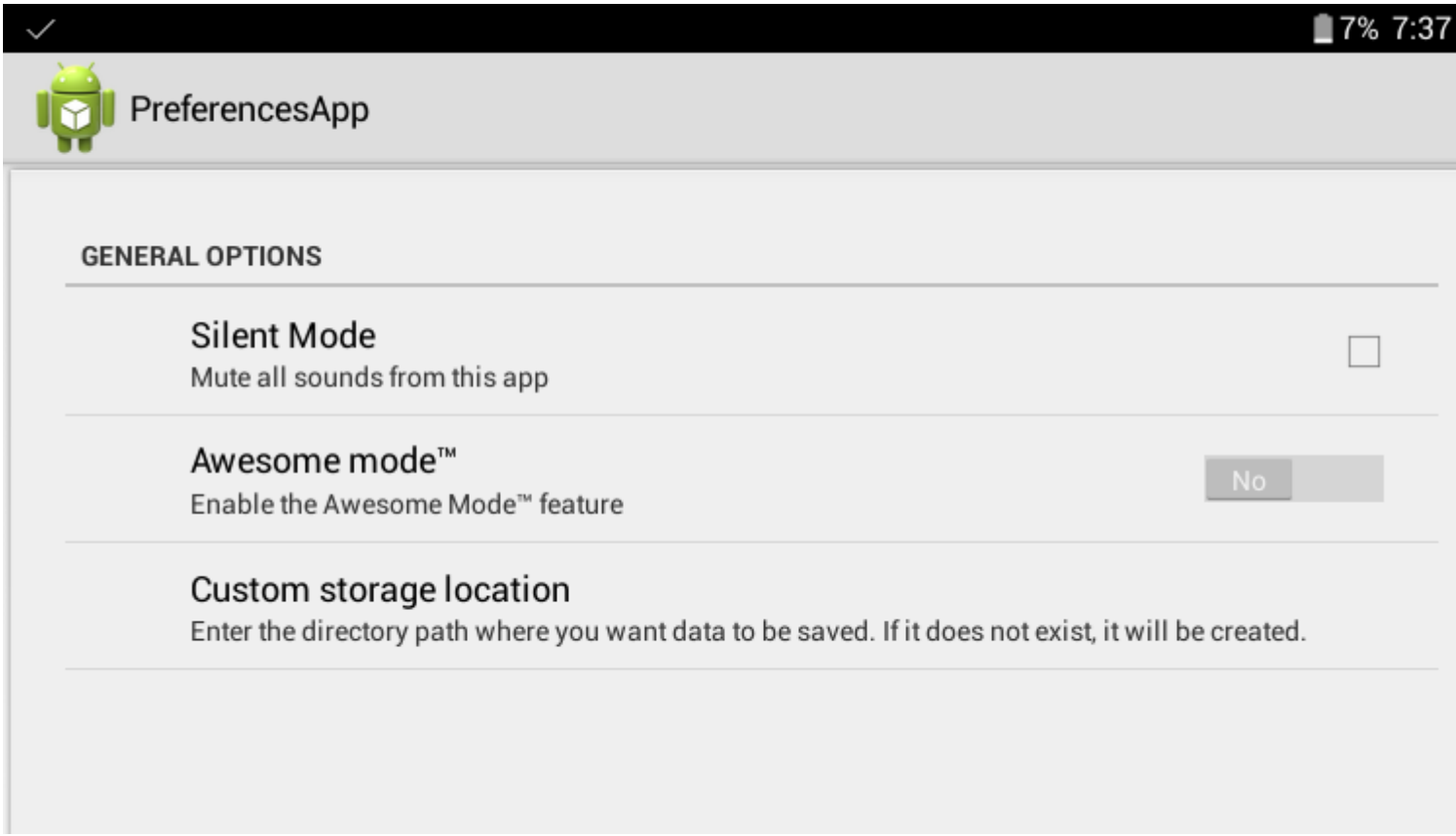
```
private static final String MY_PREF = "MyPref";

// ...
```

```
SharedPreferences prefs = ...;

// ...

SharedPreferences.Editor editor = prefs.edit();
editor.putString(MY_PREF, "value");
editor.remove(MY_PREF);
editor.apply();
```

Nach dem `apply()` enthält `prefs` zusätzlich zu dem bereits enthaltenen `prefs` "key" -> "value". Obwohl es so aussieht, als hätte ich "Schlüssel" hinzugefügt und dann entfernt, geschieht das Entfernen tatsächlich zuerst. Die Änderungen im `Editor` werden alle auf einmal angewendet, nicht in der Reihenfolge, in der Sie sie hinzugefügt haben. Alle Entfernungen passieren vor allen Puts.

Einstellungsbildschirm mit SharedPreferences implementieren

`SharedPreferences` zum `SharedPreferences` eines Bildschirms "Einstellungen" in Ihrer App verwenden, in dem der Benutzer seine Einstellungen / Optionen festlegen kann. So was:



Ein PreferenceScreen speichert Benutzervorgaben in `SharedPreferences` . Um einen PreferenceScreen zu erstellen, benötigen Sie einige Dinge:

Eine XML-Datei zum Definieren der verfügbaren Optionen:

Dies gilt für `/res/xml/preferences.xml` . Für den obigen Einstellungsbildschirm sieht das so aus:

```
<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>
```

Dies definiert die verfügbaren Optionen im Einstellungsbildschirm. Es gibt viele andere `Preference` die in der Android Developers-Dokumentation in der Präferenzklasse aufgeführt sind .

Als Nächstes benötigen wir **eine Aktivität zum Hosten unserer** Benutzerschnittstelle für **Einstellungen** . In diesem Fall ist es ziemlich kurz und sieht so aus:

```
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);
    }
}
```

Es erweitert `PreferenceActivity` und stellt die Benutzeroberfläche für den `PreferenceActivity` bereit. Es kann wie eine normale Aktivität gestartet werden, in diesem Fall mit:

```
Intent i = new Intent(this, PreferencesActivity.class);
startActivity(i);
```

Vergessen Sie nicht, `PreferencesActivity` zu Ihrer `AndroidManifest.xml` .

Die Werte der Voreinstellungen in Ihrer App zu erhalten, ist ziemlich einfach. Rufen Sie zunächst `setDefaultValues()` , um die in Ihrer XML definierten Standardwerte festzulegen, und `SharedPreferences` dann die Standard- `SharedPreferences` . Ein Beispiel:

```
//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", "");
```

Rufen Sie alle gespeicherten Einträge aus einer bestimmten SharedPreferences-Datei ab

Die `getAll()` -Methode ruft alle Werte aus den Voreinstellungen ab. Wir können damit beispielsweise den aktuellen Inhalt der `SharedPreferences` :

```
private static final String PREFS_FILE = "MyPrefs";

public static void logSharedPreferences(final Context context) {
    SharedPreferences sharedPreferences = context.getSharedPreferences(PREFS_FILE,
Context.MODE_PRIVATE);
    Map<String, ?> allEntries = sharedPreferences.getAll();
    for (Map.Entry<String, ?> entry : allEntries.entrySet()) {
        final String key = entry.getKey();
        final Object value = entry.getValue();
        Log.d("map values", key + ": " + value);
    }
}
```

Die Dokumentation warnt Sie vor der Modifizierung `Collection` von zurück `getAll` :

> Beachten Sie, dass Sie die von dieser Methode zurückgegebene Auflistung nicht ändern oder deren Inhalt ändern dürfen. Die Konsistenz Ihrer gespeicherten Daten wird in diesem Fall nicht garantiert.

Auf Änderungen von SharedPreferences warten

```
SharedPreferences sharedPreferences = ...;
sharedPreferences.registerOnSharedPreferenceChangeListener(mOnSharedPreferenceChangeListener);


private final SharedPreferences.OnSharedPreferenceChangeListener
mOnSharedPreferenceChangeListener = new SharedPreferences.OnSharedPreferenceChangeListener() {
    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
        //TODO
    }
}
```

Bitte beachten Sie:

- Der Listener wird nur ausgelöst, wenn ein Wert hinzugefügt oder geändert wurde. Wenn Sie denselben Wert festlegen, wird er nicht aufgerufen.

- Der Listener muss in einer Membervariablen und **NICHT** mit einer anonymen Klasse `registerOnSharedPreferenceChangeListener` werden, da sie von `registerOnSharedPreferenceChangeListener` mit einem schwachen Verweis `registerOnSharedPreferenceChangeListener` .
- Anstatt eine Member-Variable zu verwenden, kann sie auch direkt von der Klasse implementiert werden und dann `registerOnSharedPreferenceChangeListener(this);` aufrufen `registerOnSharedPreferenceChangeListener(this);`
- Denken Sie daran, den Listener aufzuheben, wenn er nicht mehr mit `unregisterOnSharedPreferenceChangeListener` benötigt wird.

Lesen und Schreiben von Daten in SharedPreferences mit Singleton

SharedPreferences Manager (Singleton) -Klasse zum Lesen und Schreiben aller Arten von Daten.

```
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);
    }
}
```

`Model interface` die von Klassen implementiert wird, die zu `Gson` , um eine Verschleierung durch `Gson` zu vermeiden.

```
public interface Model {

}
```

Proguard-Regeln für die `Model` Schnittstelle:

```
-keep interface com.example.app.Model
-keep class * implements com.example.app.Model { *;}
```

Verschiedene Möglichkeiten, ein Objekt von SharedPreferences zu instantiieren

Sie können auf SharedPreferences auf verschiedene Arten zugreifen:

Rufen Sie die standardmäßige SharedPreferences-Datei ab:

```
import android.preference.PreferenceManager;
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
```

Holen Sie sich eine bestimmte SharedPreferences-Datei:

```
public static final String PREF_FILE_NAME = "PrefFile";
SharedPreferences prefs = getSharedPreferences(PREF_FILE_NAME, MODE_PRIVATE);
```

Holen Sie sich SharedPreferences von einer anderen App:

```
// Note that the other app must declare prefs as MODE_WORLD_WRITEABLE
final ArrayList<HashMap<String,String>> LIST = new ArrayList<HashMap<String,String>>();
```

```
Context contextOtherApp = createPackageContext("com.otherapp", Context.MODE_WORLD_WRITEABLE);
SharedPreferences prefs = contextOtherApp.getSharedPreferences("pref_file_name",
Context.MODE_WORLD_READABLE);
```

getPreferences (int) VS getSharedPreferences (String, int)

`getPreferences(int)`

gibt die Einstellungen zurück, die unter `Activity's class name` von `Activity's class name` gespeichert wurden, wie in den Dokumenten beschrieben :

> Rufen Sie ein SharedPreferences-Objekt ab, um auf Einstellungen zuzugreifen, die für diese Aktivität privat sind. Dadurch wird einfach die zugrunde liegende Methode getSharedPreferences (String, int) aufgerufen, indem der Klassenname dieser Aktivität als Präferenzname übergeben wird.

Bei Verwendung der getSharedPreferences-Methode (String-Name, int-Modus) gibt die Methode die unter dem angegebenen `name` gespeicherten Präferenzen zurück. Wie in den Dokumenten:

> Rufen Sie den Inhalt der Voreinstellungsdatei 'name' ab und geben Sie diese ein, und geben Sie SharedPreferences zurück, über die Sie die Werte abrufen und ändern können.

Wenn der Wert, der in den `SharedPreferences` gespeichert wird, in der App verwendet werden muss, sollte `getSharedPreferences (String name, int mode)` mit einem festen Namen verwendet werden. Mit `getPreferences(int)` die Voreinstellungen der `Activity` die sie anruft, zurückgegeben / `getPreferences(int)` .

Commit vs. Apply

Die `editor.apply()` -Methode ist **asynchron** , während `editor.commit()` **synchron ist** .

Natürlich sollten Sie entweder `apply()` oder `commit()` aufrufen.

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()` wurde in 2.3 (API 9) hinzugefügt. Es wird ein Commit ausgeführt, ohne dass ein boolescher Wert zurückgegeben wird, der den Erfolg oder Misserfolg angibt.

`commit()` gibt true zurück, wenn das Speichern funktioniert, andernfalls false.

`apply()` wurde hinzugefügt, da das Android-Entwicklerteam feststellte, dass fast niemand den Rückgabewert zur Kenntnis genommen hat. Daher ist apply schneller, da es asynchron ist.

Im Gegensatz zu `commit()` , das seine Voreinstellungen synchron in den permanenten Speicher schreibt, werden die Änderungen durch `apply()` sofort in den `SharedPreferences` im Arbeitsspeicher `SharedPreferences` jedoch ein asynchroner Commit auf die Festplatte `SharedPreferences` , und es werden keine Fehler gemeldet. Wenn ein anderer Editor für diese `SharedPreferences` ein reguläres `commit()` `SharedPreferences` `commit()` während ein `apply()` noch aussteht, wird das `commit()` so lange blockiert, bis alle asynchronen Commits (apply) abgeschlossen sind, sowie alle anderen ausstehenden Sync-Commits.

Unterstützte Datentypen in SharedPreferences

`SharedPreferences` können Sie nur primitive Datentypen speichern (`boolean` , `float` , `long` , `int` , `String` und `string set`). Sie können keine komplexeren Objekte in `SharedPreferences` , und als solche ist wirklich ein Ort zum Speichern von Benutzereinstellungen oder Ähnlichem gedacht. Es ist nicht als Datenbank gedacht, in der Benutzerdaten gespeichert werden (z. B. Speichern einer Aufgabenliste, die ein Benutzer erstellt hat).

Um etwas in `SharedPreferences` zu speichern, `SharedPreferences` Sie einen Schlüssel und einen Wert. Mit dem Schlüssel können Sie nachschlagen, was Sie später gespeichert haben, und auf die Wertdaten, die Sie speichern möchten.

```
    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();
```

Speichern, Abrufen, Entfernen und Löschen von Daten aus SharedPreferences

Erstellen Sie SharedPreferences BuyyaPref

```
SharedPreferences pref = getApplicationContext().getSharedPreferences("BuyyaPref",
MODE_PRIVATE);
Editor editor = pref.edit();
```

Speichern von Daten als KEY / VALUE-Paar

```
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
```

Holen Sie sich SharedPreferences-Daten

Wenn der Wert für den Schlüssel nicht vorhanden ist, geben Sie den zweiten Parameterwert zurück.

```
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
```

Schlüsselwert aus SharedPreferences löschen

```
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
```

Löschen Sie alle Daten aus SharedPreferences

```
 editor.clear();
 editor.commit(); // commit changes
```

Unterstützt Pre-Honeycomb mit StringSet

Hier ist die Utility-Klasse:

```
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() {}
}
```

Ein Beispiel zum Speichern von Einstellungen als StringSet-Datentyp ist:

```
Set<String> sets = new HashSet<>();
sets.add("John");
sets.add("Nicko");
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
SharedPreferencesCompat.putStringSet(preferences.edit(), "pref_people", sets);
```

Um sie zurückzuholen:

```
Set<String> people = SharedPreferencesCompat.getStringSet(preferences, "pref_people", new
HashSet<String>());
```

Referenz: Android Support-Voreinstellung

Fügen Sie einen Filter für EditTextPreference hinzu

Diese Klasse erstellen:

```
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;
    }
}
```

Benutzen :

```
EditText compressPic = ((EditTextPreference)
findPreference(getString("pref_key_compress_pic"))).getEditText();
compressPic.setFilters(new InputFilter[]{ new InputFilterMinMax(1, 100) });
```

Gemeinsame Einstellungen online lesen: https://riptutorial.com/de/android/topic/119/gemeinsame-einstellungen

Kapitel 105: Genymotion für Android

Einführung

Genymotion ist ein schneller Emulator von Drittanbietern, der anstelle des Standard-Emulators von Android verwendet werden kann. In manchen Fällen ist es genauso gut oder besser als die Entwicklung auf tatsächlichen Geräten!

Examples

Installation von Genymotion, der kostenlosen Version

Schritt 1 - `VirtualBox` installieren

Laden Sie VirtualBox entsprechend Ihrem Betriebssystem herunter und installieren Sie sie. , es ist erforderlich, um `Genymotion` .

Schritt 2 - `Genymotion` herunterladen

Gehen Sie zur Download-Seite von Genymotion und laden Sie `Genymotion` entsprechend Ihrem Betriebssystem herunter.

Hinweis: Sie müssen ein neues Konto erstellen ODER sich mit Ihrem Konto anmelden.

Schritt 3 - Installation von `Genymotion`

Wenn Sie sich unter `Linux` auf diese Antwort beziehen, können Sie eine `.bin` Datei installieren und ausführen.

Schritt 4 - Installieren der Emulatoren von `Genymotion`

- `Genymotion`
- Klicken Sie auf die Schaltfläche Hinzufügen (in der oberen Leiste).
- Melden Sie sich mit Ihrem Konto an und Sie können die verfügbaren Emulatoren durchsuchen.
- Wählen Sie aus und installieren Sie, was Sie benötigen.

Schritt 5 - Integration von `genymotion` in `Android Studio`

`Genymotion` können mit eingebunden werden `Android Studio` über ein Plugin, hier sind die Schritte , die sie in installieren `Android Studio`

- Gehen Sie zu Datei / Einstellungen (für Windows und Linux) oder zu Android Studio / Voreinstellungen (für Mac OS X).
- Wählen Sie Plugins aus und klicken Sie auf Repositorys durchsuchen.
- Klicken Sie mit der rechten Maustaste auf Genymotion und klicken Sie auf Herunterladen und installieren.

Sie sollten jetzt das Plugin-Symbol sehen können, siehe dieses Bild

Beachten Sie, dass Sie die Symbolleiste anzeigen möchten, indem Sie auf Ansicht> Symbolleiste klicken.

Schritt 6 - `Genymotion` von `Android Studio Genymotion`

- Gehen Sie zu Datei / Einstellungen (für Windows und Linux) oder zu Android Studio / Voreinstellungen (für Mac OS X).
- Gehen Sie zu Other Settings / Genymotion, fügen Sie den Pfad des `Genymotion's` Ordners hinzu und übernehmen Sie Ihre Änderungen.

Jetzt sollten Sie `Genymotion's` Emulator ausführen `Genymotion's` indem Sie auf das Plugin-Symbol `Genymotion's` und einen installierten Emulator auswählen und dann die Start-Taste drücken!

Google Framework auf Genymotion

Wenn Entwickler Google Maps oder andere Google-Dienste wie Google Mail, Youtube, Google Drive usw. testen möchten, müssen sie zunächst das Google-Framework auf Genymotion installieren. Hier sind die Schritte: -

4.4 Kitkat
5,0 Lutscher
5.1 Lutscher
6,0 Eibisch
7,0 Nougat
7.1 Nougat (Webview-Patch)

1. Download vom obigen Link
2. Ziehen Sie die heruntergeladene ZIP-Datei einfach per Drag & Drop in genymotion und starten Sie sie erneut
3. Fügen Sie ein Google-Konto hinzu und laden Sie "Google Play Music" herunter.

Referenz:-
Stapelüberlauffrage zu diesem Thema

Genymotion für Android online lesen: https://riptutorial.com/de/android/topic/9245/genymotion-fur-

android

Kapitel 106: Geräteanzeige-Metriken

Examples

Rufen Sie die Pixelabmessungen der Bildschirme ab

Um die Breite und Höhe der Bildschirme in Pixel zu erhalten, können Sie die WindowManagers-Anzeigemetriken verwenden.

```
// Get display metrics
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
```

Diese DisplayMetrics enthalten eine Reihe von Informationen über den Bildschirm des Geräts, wie Dichte oder Größe:

```
// Get width and height in pixel
Integer heightPixels = metrics.heightPixels;
Integer widthPixels = metrics.widthPixels;
```

Holen Sie sich die Bildschirmdichte

Um die Bildschirmdichte zu erhalten, können wir auch die Windowmanagers DisplayMetrics verwenden . Dies ist ein schnelles Beispiel:

```
// Get density in dpi
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
int densityInDpi =  metrics.densityDpi;
```

Formel px zu dp, dp zu px Unterhaltung

DP zu Pixel:

```
private int dpToPx(int dp)
{
    return (int) (dp * Resources.getSystem().getDisplayMetrics().density);
}
```

Pixel zu DP:

```
private int pxToDp(int px)
{
    return (int) (px / Resources.getSystem().getDisplayMetrics().density);
}
```

Geräteanzeige-Metriken online lesen: https://riptutorial.com/de/android/topic/4207/gerateanzeige-

metriken

Kapitel 107: Gestenerkennung

Bemerkungen

Offizielle Dokumentation: Gemeinsame Gesten erkennen

Examples

Swipe-Erkennung

```
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() {
    }

}
```

Auf einen Blick angewendet ...

```
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");
    }

});
```

Grundgestenerkennung

```
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;
    }

}
```

Gestenerkennung online lesen: https://riptutorial.com/de/android/topic/4711/gestenerkennung

Kapitel 108: Geteilter Bildschirm / Multi-Screen-Aktivitäten

Examples

Split Screen wurde in Android Nougat eingeführt.

Legen Sie dieses Attribut in Ihrem Manifest oder Element fest, um die Mehrfensteranzeige zu aktivieren oder zu deaktivieren:

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

Wenn dieses Attribut auf true gesetzt ist, kann die Aktivität im Split-Screen- und im Freeform-Modus gestartet werden. Wenn das Attribut auf "false" gesetzt ist, unterstützt die Aktivität den Mehrfenstermodus nicht. Wenn dieser Wert falsch ist und der Benutzer versucht, die Aktivität im Mehrfenstermodus zu starten, wird die Aktivität auf dem gesamten Bildschirm angezeigt.

Wenn Ihre App auf API-Ebene 24 abzielt, Sie jedoch keinen Wert für dieses Attribut angeben, wird der Wert des Attributs standardmäßig auf "true" gesetzt.

Der folgende Code zeigt, wie Sie die Standardgröße und den Standardort einer Aktivität sowie die Mindestgröße angeben, wenn die Aktivität im Freiformmodus angezeigt wird:

```
<--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>
```

Deaktivierte Funktionen im Mehrfenstermodus

Bestimmte Funktionen werden deaktiviert oder ignoriert, wenn sich ein Gerät im Mehrfenstermodus befindet, da sie für eine Aktivität, bei der der Bildschirm des Geräts möglicherweise für andere Aktivitäten oder Apps freigegeben wird, nicht sinnvoll sind. Zu diesen Funktionen gehören:

1. Einige Anpassungsoptionen für die Systemoberfläche sind deaktiviert. Beispielsweise können Apps die Statusleiste nicht ausblenden, wenn sie nicht im Vollbildmodus ausgeführt werden.

2. Das System ignoriert Änderungen am Attribut **android: screenOrientation** .

Wenn Ihre App auf API Level 23 oder niedriger abzielt

Wenn Ihre App auf API-Level 23 oder niedriger abzielt und der Benutzer versucht, die App im Mehrfenstermodus zu verwenden, ändert das System die App zwangsweise, es sei denn, die App gibt eine feste Ausrichtung vor.

Wenn Ihre App keine feste Ausrichtung angibt, sollten Sie Ihre App auf einem Gerät mit Android 7.0 oder höher starten und versuchen, die App in den Split-Screen-Modus zu versetzen. Stellen Sie sicher, dass die Benutzererfahrung akzeptabel ist, wenn die Größe der App zwangsweise geändert wird.

Wenn die App eine feste Ausrichtung deklariert, sollten Sie versuchen, die App in den Mehrfenstermodus zu schalten. Stellen Sie sicher, dass die App im Vollbildmodus bleibt, wenn Sie dies tun.

Geteilter Bildschirm / Multi-Screen-Aktivitäten online lesen:
https://riptutorial.com/de/android/topic/7130/geteilter-bildschirm---multi-screen-aktivitaten

Kapitel 109: Gleiten

Einführung

**** WARNUNG Diese Dokumentation ist nicht gepflegt und häufig ungenau. ****

Die offizielle Dokumentation von Glide ist eine viel bessere Quelle:

Für Glide v4 siehe http://bumptech.github.io/glide/ . Für Glide v3 siehe https://github.com/bumptech/glide/wiki .

Bemerkungen

Glide ist ein schnelles und effizientes Open-Source-Media-Management- und Image-Load-Framework für Android, das Media-Dekodierung, Speicher- und Datenträger-Caching und Ressourcen-Pooling in einer einfachen und benutzerfreundlichen Oberfläche vereint.

Glide unterstützt das Abrufen, Dekodieren und Anzeigen von Videobildern, Bildern und animierten GIFs. Glide enthält eine flexible API, mit der sich Entwickler an fast jeden Netzwerkstack anschließen können.
Glide verwendet standardmäßig einen benutzerdefinierten HttpUrlConnection- basierten Stack, enthält jedoch auch Plug -Ins für Hilfsprogrammbibliotheken in das Volley- Projekt von Google oder die OkHttp-Bibliothek von Square .

Das Hauptaugenmerk von Glide liegt darauf, das Scrollen aller Arten von Bildern so reibungslos und schnell wie möglich zu gestalten. Glide ist jedoch in fast allen Fällen effektiv, in denen ein Remote-Bild abgerufen, in der Größe verändert und angezeigt werden muss.

Quellcode und weitere Dokumentation finden Sie auf GitHub: https://github.com/bumptech/glide

Examples

Fügen Sie Ihrem Projekt Glide hinzu

Aus der offiziellen Dokumentation :

Mit Gradle:

```
repositories {
  mavenCentral() // jcenter() works as well because it pulls from Maven Central
}

dependencies {
  compile 'com.github.bumptech.glide:glide:4.0.0'
  compile 'com.android.support:support-v4:25.3.1'
  annotationProcessor 'com.github.bumptech.glide:compiler:4.0.0'
}
```

Mit Maven:

```
<dependency>
  <groupId>com.github.bumptech.glide</groupId>
  <artifactId>glide</artifactId>
  <version>4.0.0</version>
</dependency>
<dependency>
  <groupId>com.google.android</groupId>
  <artifactId>support-v4</artifactId>
  <version>r7</version>
</dependency>
<dependency>
  <groupId>com.github.bumptech.glide</groupId>
  <artifactId>compiler</artifactId>
  <version>4.0.0</version>
  <optional>true</optional>
</dependency>
```

Abhängig von Ihrer ProGuard (DexGuard) -Konfiguration und -nutzung müssen Sie möglicherweise auch die folgenden Zeilen in Ihre proguard.cfg einfügen (weitere Informationen finden Sie im Glide-Wiki):

```
-keep public class * implements com.bumptech.glide.module.GlideModule
-keep public class * extends com.bumptech.glide.AppGlideModule
-keep public enum com.bumptech.glide.load.resource.bitmap.ImageHeaderParser$**     {
  **[] $VALUES;
  public *;
}

# for DexGuard only
-keepresourcexmlelements manifest/application/meta-data@value=GlideModule
```

Ein Bild laden

Bildansicht

So laden Sie ein Bild von einer angegebenen URL, einem `ImageView` , einer Ressourcen-ID oder einem anderen Modell in eine `ImageView` :

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
String yourUrl = "http://www.yoururl.com/image.png";

Glide.with(context)
    .load(yourUrl)
    .into(imageView);
```

Ersetzen `yourUrl` für Uris `yourUrl` durch Ihren Uri (`content://media/external/images/1`). Ersetzen `yourUrl` für Drawables `yourUrl` durch Ihre Ressourcen-ID (`R.drawable.image`).

RecyclerView und ListView

In ListView oder RecyclerView können Sie genau die gleichen Zeilen verwenden:

```
@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);
}
```

Wenn Sie in `onBindViewHolder` kein `ImageView` starten `onBindViewHolder` , vergewissern Sie sich, dass Sie `ImageView` Glide `ImageView` `clear()` `ImageView` , bevor Sie die `ImageView` :

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    MyViewHolder myViewHolder = (MyViewHolder) viewHolder;
    String currentUrl = myUrls.get(position);

    if (TextUtils.isEmpty(currentUrl)) {
        Glide.clear(viewHolder.imageView);
        // Now that the view has been cleared, you can safely set your own resource
        viewHolder.imageView.setImageResource(R.drawable.missing_image);
    } else {
        Glide.with(context)
            .load(currentUrl)
            .into(myViewHolder.imageView);
    }
}
```

Glide Circle Transformation

Erstellen Sie ein Kreisbild mit Gleiten.

```
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();
    }
}
```

Verwendungszweck:

```
Glide.with(context)
    .load(yourimageurl)
    .transform(new CircleTransform(context))
    .into(userImageView);
```

Standardumwandlungen

Glide enthält zwei Standardumwandlungen, die Mitte und die Mitte des Zuschnitts.

Sitzmitte:

```
Glide.with(context)
    .load(yourUrl)
    .fitCenter()
    .into(yourView);
```

Das Fit-Center führt dieselbe Transformation durch wie ScaleType.FIT_CENTER von Android.

Center Ernte:

```
Glide.with(context)
    .load(yourUrl)
    .centerCrop()
    .into(yourView);
```

Die Mitte des Zuschnitts führt dieselbe Transformation durch wie ScaleType.CENTER_CROP von Android.

Weitere Informationen finden Sie im Glide-Wiki .

Bild mit abgerundeten Ecken mit benutzerdefiniertem Ziel "Gleiten"

Machen Sie zuerst eine Utility-Klasse oder verwenden Sie diese Methode in der erforderlichen

Klasse

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

Bild wird geladen:

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Da Sie asBitmap () verwenden, werden die Animationen jedoch entfernt. Sie können Ihre eigene Animation an dieser Stelle mithilfe der animate () -Methode verwenden.

Beispiel mit ähnlicher Einblendung in die Standard-Glide-Animation.

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .animate(R.anim.abc_fade_in)
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Bitte beachten Sie, dass es sich bei dieser Animation um eine private Ressource der Unterstützungsbibliothek handelt, deren Verwendung nicht empfohlen wird, da sie sich ändern oder sogar entfernen kann.

Beachten Sie, dass Sie auch eine Unterstützungsbibliothek benötigen, um RoundedBitmapDrawableFactory verwenden zu können

Bilder vorladen

So laden Sie Remote-Images vorab und stellen sicher, dass das Image nur einmal heruntergeladen wird:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE)
    .preload();
```

Dann:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE) // ALL works here too
    .into(imageView);
```

So laden Sie lokale Images vorab und stellen Sie sicher, dass sich eine transformierte Kopie im Festplattencache (und möglicherweise im Speichercache) befindet:

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // Or whatever transformation you want
    .preload(200, 200); // Or whatever width and height you want
```

Dann:

```
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);
```

Platzhalter und Fehlerbehandlung

Wenn Sie während des Ladens einen Drawable hinzufügen möchten, können Sie einen Platzhalter hinzufügen:

```
Glide.with(context)
    .load(yourUrl)
    .placeholder(R.drawable.placeholder)
    .into(imageView);
```

Wenn Sie möchten, dass ein Drawable angezeigt wird, wenn das Laden aus irgendeinem Grund fehlschlägt:

```
Glide.with(context)
    .load(yourUrl)
    .error(R.drawable.error)
    .into(imageView);
```

Wenn Sie möchten, dass ein Drawable angezeigt wird, wenn Sie ein Nullmodell (URL, Uri, Dateipfad usw.) angeben:

```
Glide.with(context)
    .load(maybeNullUrl)
    .fallback(R.drawable.fallback)
    .into(imageView);
```

Bild in einer kreisförmigen ImageView ohne benutzerdefinierte Transformationen laden

Erstellen Sie ein benutzerdefiniertes BitmapImageViewTarget, in das das Bild geladen werden soll:

```
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);
    }
}
```

Verwendungszweck:

```
Glide
    .with(context)
    .load(yourimageidentifier)
    .asBitmap()
    .into(new CircularBitmapImageViewTarget(context, imageView));
```

Fehler beim Laden des Glide-Images

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

Hier am Ort `SET_YOUR_DEFAULT_IMAGE` können Sie einen beliebigen `Drawable` . Dieses Bild wird angezeigt, wenn das Laden von Bildern fehlgeschlagen ist.

Gleiten online lesen: https://riptutorial.com/de/android/topic/1091/gleiten

Kapitel 110: Google Awareness-APIs

Bemerkungen

Denken Sie daran, dass die Snapshot-API verwendet wird, um den aktuellen Status anzufordern, während die Fence-API kontinuierlich nach einem bestimmten Status sucht und Rückrufe sendet, wenn eine App nicht ausgeführt wird.

Insgesamt gibt es einige grundlegende Schritte, um die Snapshot-API oder die Fence-API zu verwenden:

- Rufen Sie einen API-Schlüssel von der Google Developers Console ab
- Fügen Sie dem Manifest die erforderlichen Berechtigungen und den API-Schlüssel hinzu:

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

- Initialisieren Sie den `GoogleApiClient` irgendwo, vorzugsweise in der onCreate () -Methode Ihrer Aktivität.

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

- Rufen Sie die API Ihrer Wahl auf
- Ergebnis analysieren

Eine einfache Methode zum Überprüfen der erforderlichen Benutzerberechtigung ist eine Methode wie diese:

```
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

Aktuelle Benutzeraktivität mithilfe der Snapshot-API abrufen

Verwenden Sie für einmalige, nicht konstante Anforderungen für die physische Aktivität eines Benutzers die Snapshot-API:

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

Erhalten Sie den Kopfhörerstatus mit der Snapshot-API

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getHeadphoneState(client)
    .setResultCallback(new ResultCallback<HeadphoneStateResult>() {
        @Override
        public void onResult(@NonNull HeadphoneStateResult headphoneStateResult) {
            Log.i(TAG, "Headphone state connection state: " +
                headphoneStateResult.getHeadphoneState()
                .getState() == HeadphoneState.PLUGGED_IN));
        }
    });
```

Rufen Sie den aktuellen Standort mithilfe der Snapshot-API ab

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

Mit der Snapshot-API können Sie Orte in der Nähe abrufen

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

Um die Daten an diesen Orten zu erhalten, gibt es folgende Möglichkeiten:

```
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()));
```

Holen Sie sich das aktuelle Wetter mithilfe der Snapshot-API

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

Holen Sie sich Änderungen in der Benutzeraktivität mit der Fence-API

Wenn Sie feststellen möchten, wann Ihr Benutzer eine Aktivität startet oder beendet, wie z. B. Gehen, Laufen oder eine andere Aktivität der `DetectedActivityFence` Klasse, können Sie einen Zaun für die zu erkennende Aktivität erstellen und werden benachrichtigt, wenn Ihr Benutzer startet beendet diese Aktivität. Wenn Sie einen `BroadcastReceiver` , erhalten Sie eine `Intent` mit Daten, die die Aktivität enthalten:

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

Jetzt können Sie die Absicht mit einem `BroadcastReceiver` empfangen, um Rückrufe zu erhalten, wenn der Benutzer die Aktivität ändert:

```
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;
        }
    }
}
```

Rufen Sie mithilfe der Fence-API Änderungen für den Standort innerhalb

eines bestimmten Bereichs ab

Wenn Sie feststellen möchten, wann Ihr Benutzer einen bestimmten Ort betritt, können Sie einen Zaun für den bestimmten Ort mit einem gewünschten Radius erstellen und benachrichtigt werden, wenn Ihr Benutzer den Ort betritt oder verlässt.

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

Erstellen Sie nun einen BroadcastReciver, um Updates im Benutzerstatus zu erhalten:

```
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;
        }
    }
}
```

Google Awareness-APIs online lesen: https://riptutorial.com/de/android/topic/3361/google-awareness-apis

Kapitel 111: Google Drive API

Einführung

Google Drive ist ein von **Google** erstellter File-Hosting-Service. Es bietet Dateispeicherungsdienst und ermöglicht dem Benutzer, Dateien in die Cloud hochzuladen und auch mit anderen Personen zu teilen. Mit der Google Drive API können wir Dateien zwischen Computern oder mobilen Geräten und Google Drive Cloud synchronisieren.

Bemerkungen

Rechtliches

Wenn Sie die Google Drive Android-API in Ihrer Anwendung verwenden, müssen Sie den Text der Google Play Services-Zuordnung als Teil des Abschnitts "Rechtliche Hinweise" in Ihre Anwendung einfügen.

Es wird empfohlen, dass Sie rechtliche Hinweise als unabhängigen Menüpunkt oder als Teil eines "Info" -Menüpunkts hinzufügen.

Sie können `GooglePlayServicesUtil.getOpenSourceSoftwareLicenseInfo()` , um den Attributionstext zur Laufzeit `GooglePlayServicesUtil.getOpenSourceSoftwareLicenseInfo()` .

Examples

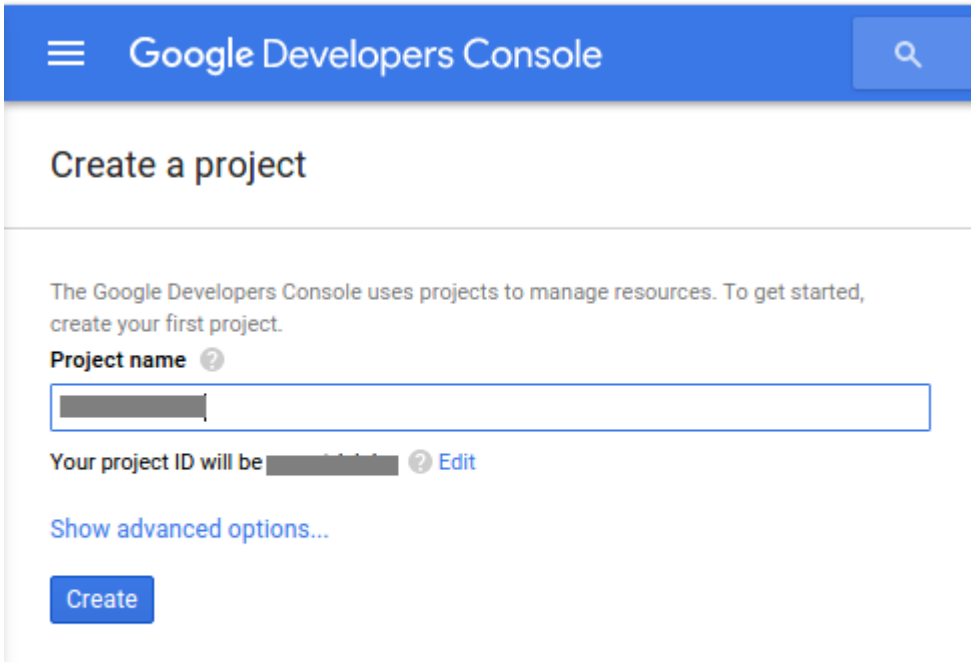
Integrieren Sie Google Drive in Android

Erstellen Sie ein neues Projekt in der Google Developer Console

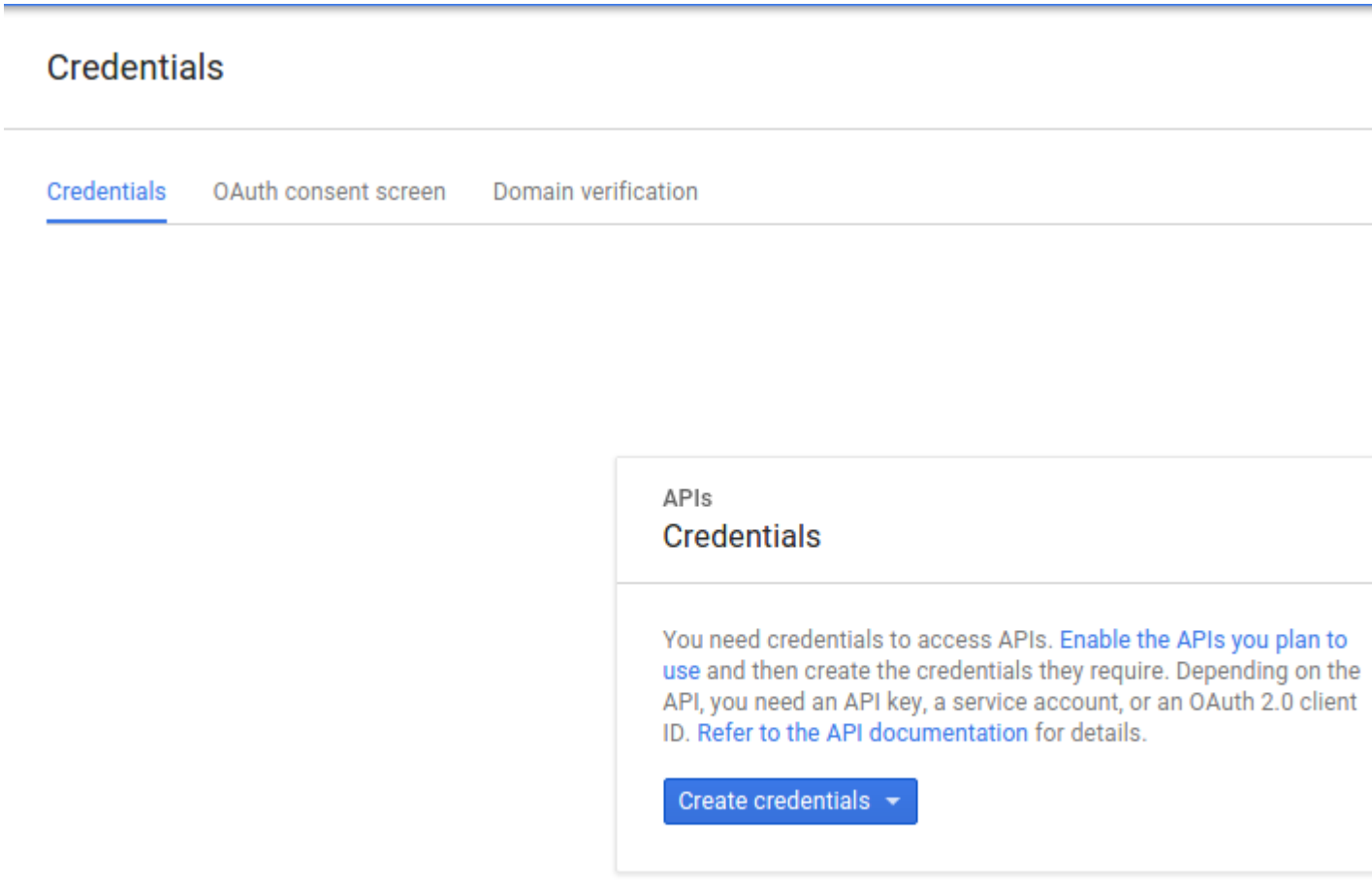
Um die Android-Anwendung in Google Drive zu integrieren, erstellen Sie die Anmeldeinformationen des Projekts in der Google Developers Console. Daher müssen wir ein Projekt in der Google Developer Console erstellen.

Führen Sie die folgenden Schritte aus, um ein Projekt in der Google Developer Console zu erstellen:

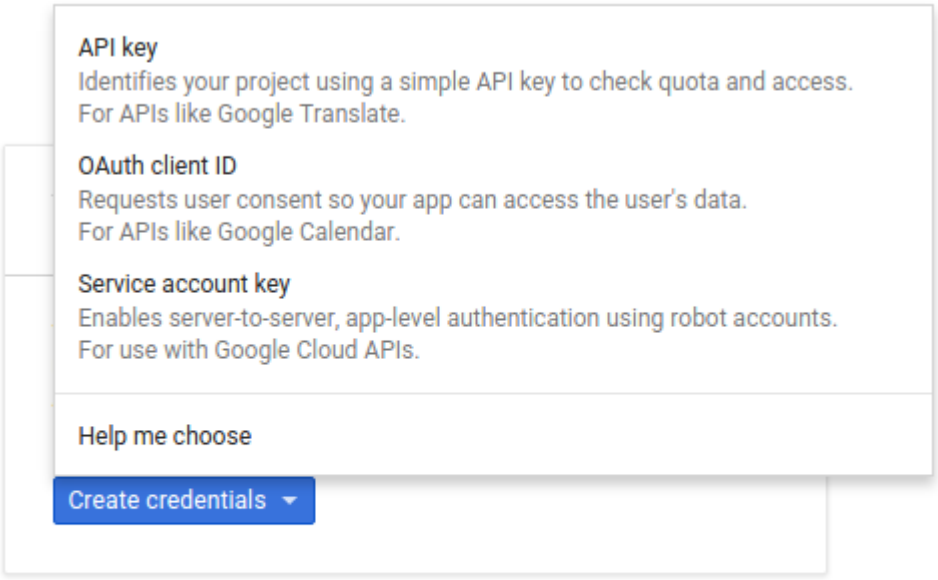
- Rufen Sie die Google Developer Console für Android auf. Füllen Sie Ihren **Projektnamen** in das Eingabefeld ein und klicken Sie auf die Schaltfläche **Erstellen** Sie ein neues Projekt auf Google Developer - Konsole zu erstellen.



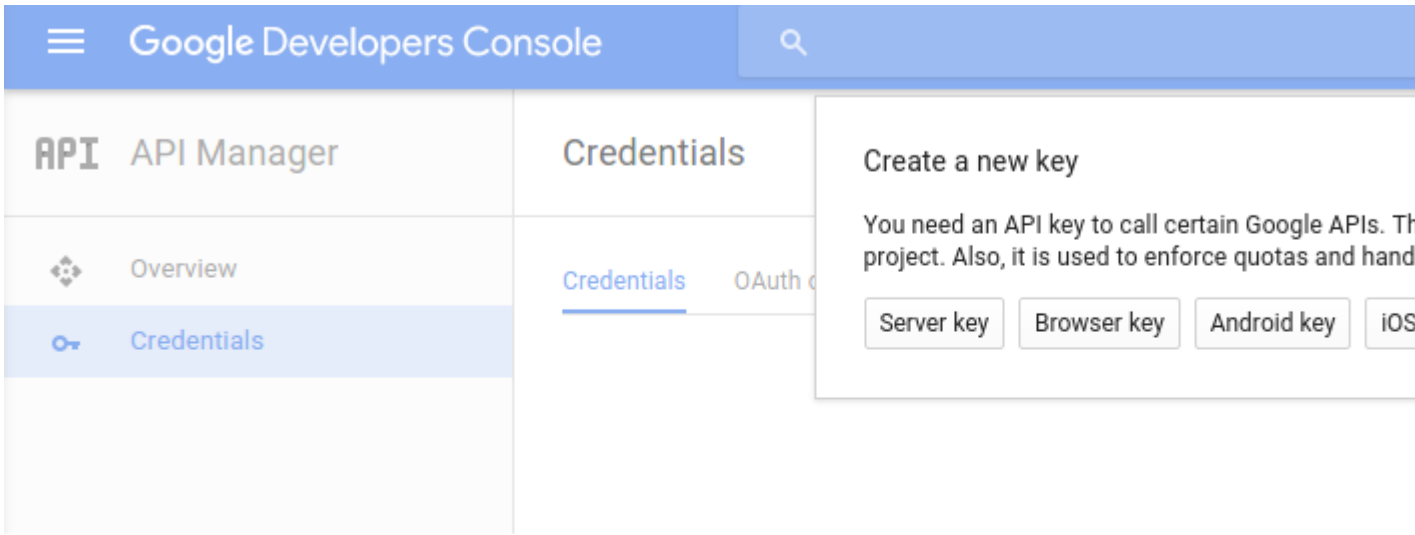
- Wir müssen Anmeldeinformationen erstellen, um auf die API zuzugreifen. Klicken Sie also auf die Schaltfläche **Anmeldeinformationen erstellen** .



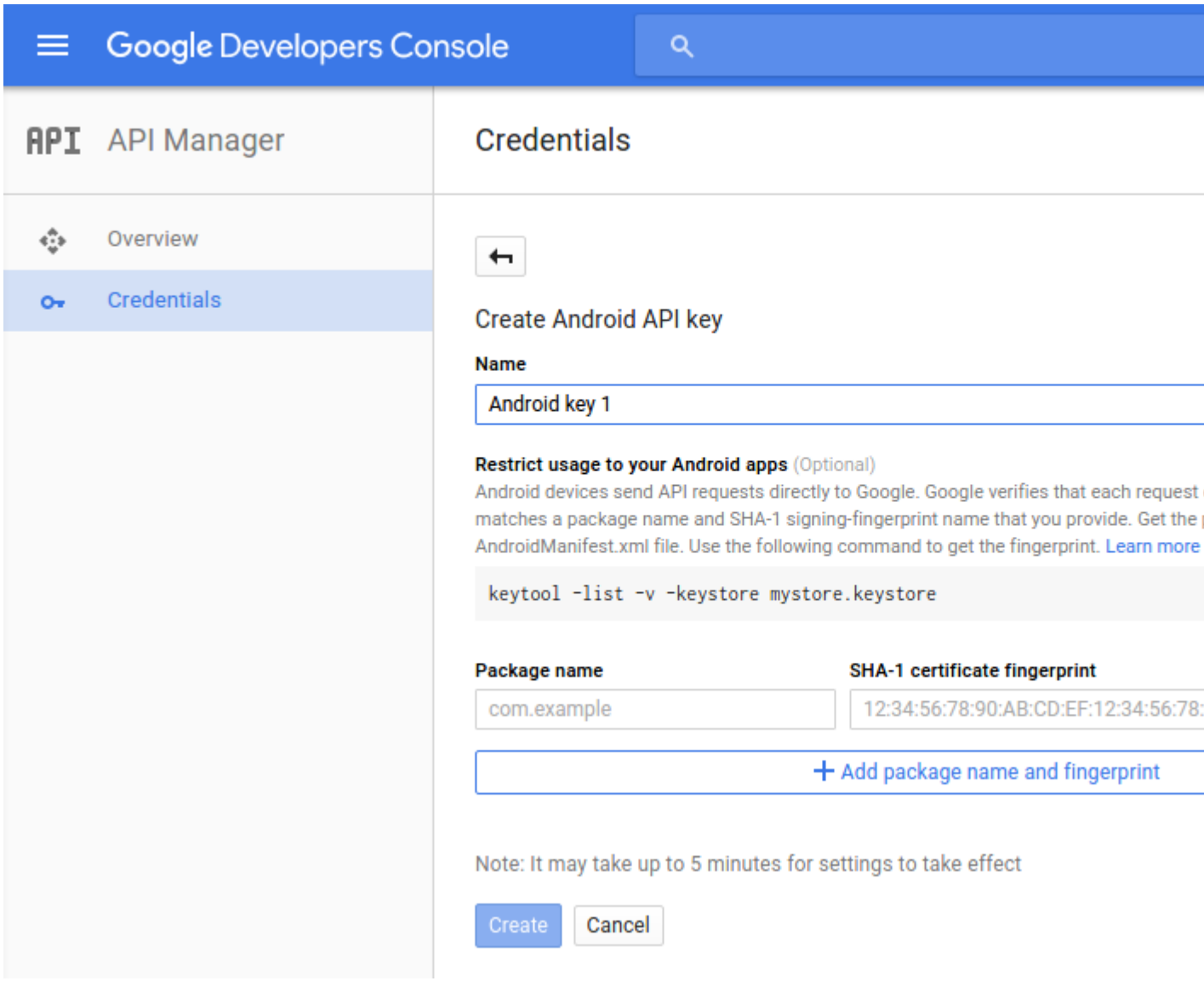
- Nun öffnet sich ein Popup-Fenster. Klicken Sie auf **API Key** - Option in der Liste API - Schlüssel zu erstellen.



- Wir benötigen einen API-Schlüssel, um Google-APIs für Android aufzurufen. Klicken Sie also auf den **Android-Schlüssel** , um Ihr Android-Projekt zu identifizieren.



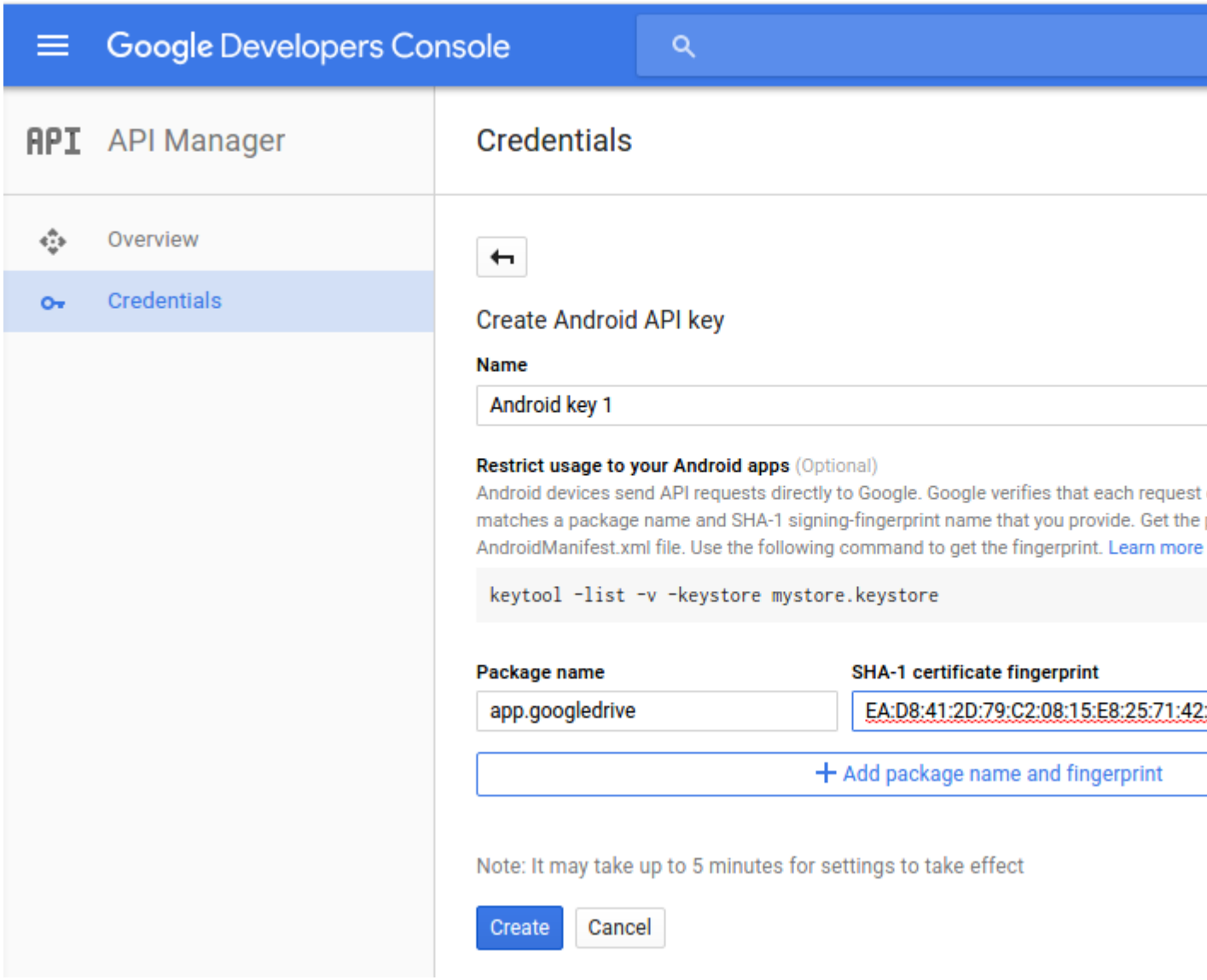
- Als Nächstes müssen wir den Paketnamen des Android-Projekts und den **SHA-1-Fingerabdruck** in den Eingabefeldern hinzufügen, um den API-Schlüssel zu erstellen.



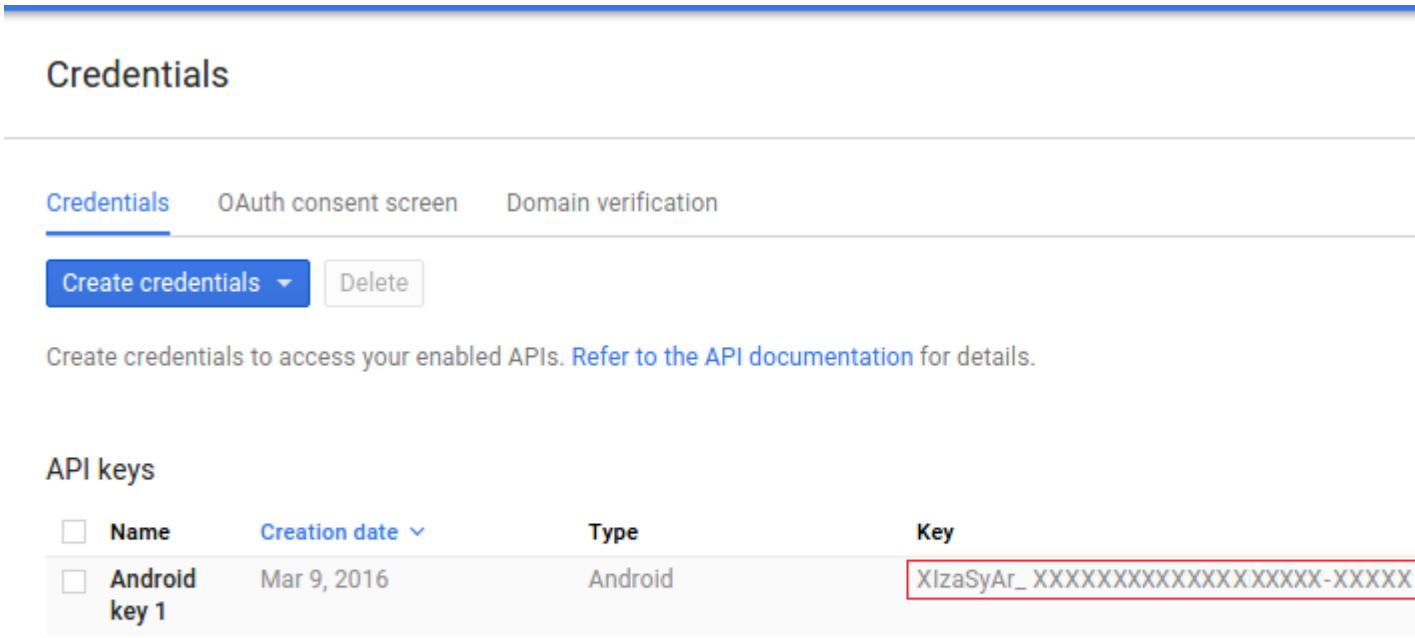
- Wir müssen **SHA-1-Fingerabdruck erstellen** . Öffnen Sie also Ihr Terminal und führen **Sie das Dienstprogramm Keytool aus** , um den SHA1-Fingerabdruck zu erhalten. Während Keytool - Dienstprogramm ausgeführt wird , müssen Sie **Kennwort** für **Schlüsselspeicher** zur Verfügung zu stellen. Das Standardkennwort für das Entwicklungs-Keytool lautet **„Android“** . `keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -list -v`

```
[redacted]@[redacted]:~$ keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -list -v
Enter keystore password:
Alias name: androiddebugkey
Creation date: 18 Jul, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 3adbdb98
Valid from: Sat Jul 18 09:32:08 IST 2015 until: Mon Jul 10 09:32:08 IST 2045
Certificate fingerprints:
         MD5:  77:C7:A9:6A:30:0F:43:B9:84:E0:61:0F:B2:B6:22:74
         SHA1: EA:D8:41:2D:79:C2:08:15:E8:25:71:42:3F:0E:51:A5:52:4C:EF:40
         SHA256: A2:12:5A:18:E2:F3:FE:8B:93:E8:03:0C:12:3A:52:8D:B5:B0:70:32:C
F3:A7:C3:47:F0:9E:B6:8E:AF:33:68
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: D3 8F C7 0C 95 B4 DA 73   6B 67 99 5A A3 C0 05 4A  .......skg.Z...J
0010: 93 BE 25 4F                                        ..%O
]
]
```

- **Fügen Sie nun** in den Eingabefeldern auf der Seite mit den Anmeldeinformationen den **Paketnamen** und den **SHA-1-Fingerabdruck hinzu** . Klicken Sie abschließend auf die Schaltfläche Erstellen, um den API-Schlüssel zu erstellen.



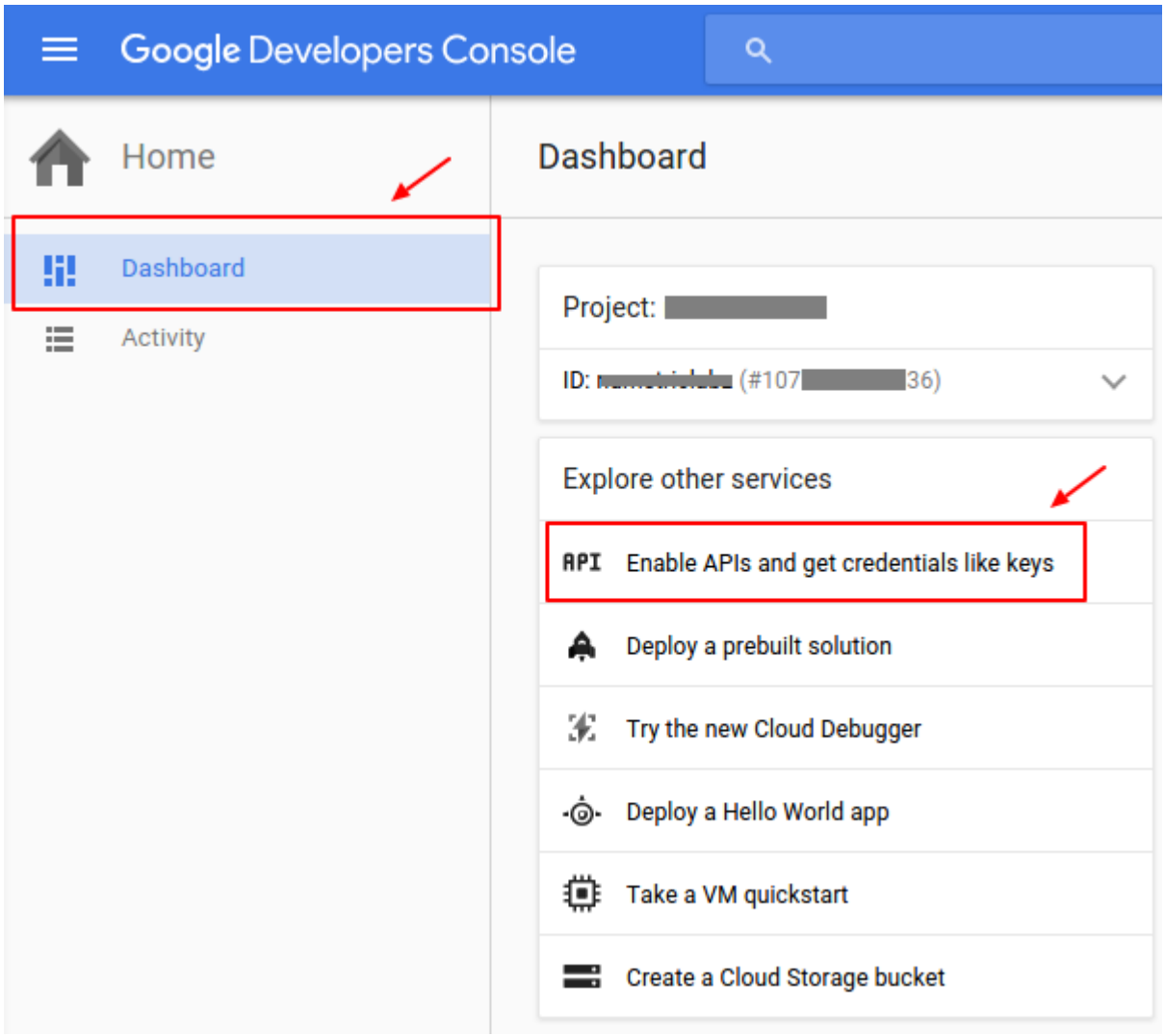
- Dadurch wird ein API-Schlüssel für Android erstellt. Wir verwenden diesen API-Schlüssel, um die Android-Anwendung in Google Drive zu integrieren.



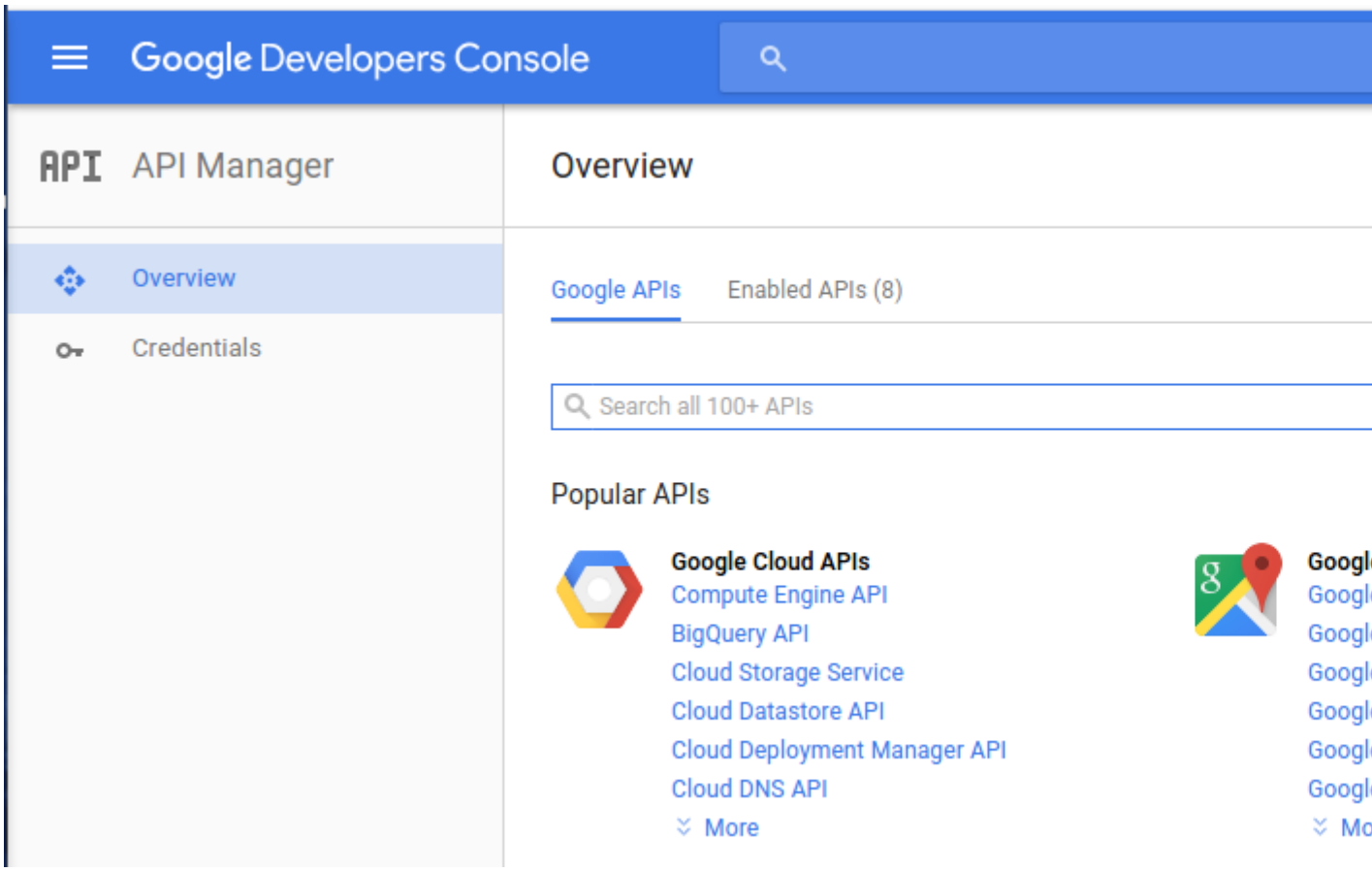
Aktivieren Sie die Google Drive-API

Wir müssen Google Drive Api für den Zugriff auf auf Google Drive gespeicherte Dateien über eine Android-Anwendung aktivieren. Führen Sie die folgenden Schritte aus, um die Google Drive-API zu aktivieren:

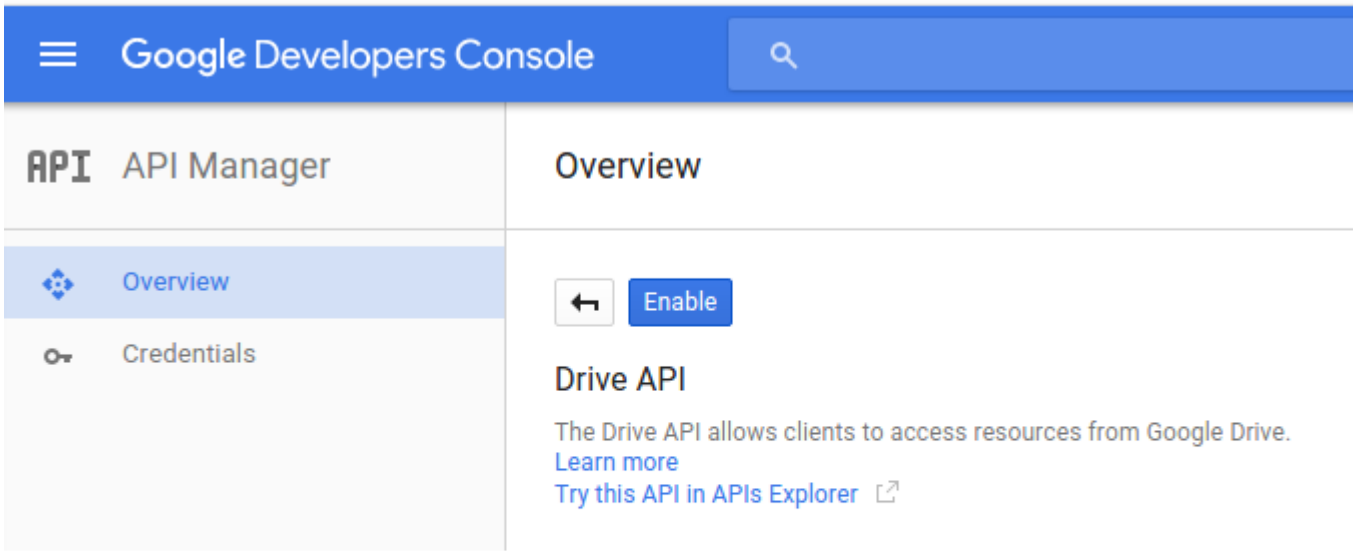
- Gehen Sie zu Ihrem Google Developer Console-Dashboard und klicken Sie auf " **APIs aktivieren", um Anmeldeinformationen wie Schlüssel zu erhalten.** Anschließend werden beliebte Google-APIs angezeigt.



- Klicken Sie auf den Link **Drive API** , um die Übersichtsseite der Google Drive API zu öffnen.



- Klicken Sie auf die Schaltfläche Aktivieren, um die Google Drive API zu aktivieren. Es ermöglicht dem Client den Zugriff auf Google Drive.



Internet-Berechtigung hinzufügen

App benötigt **Internet-** Zugriff auf Google Drive-Dateien. Verwenden Sie den folgenden Code, um

Internetberechtigungen in der Datei AndroidManifest.xml einzurichten:

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

Google Play-Services hinzufügen

Wir verwenden die **Google Play-Services-API, zu** der die **Google Drive-Android-API gehört** . Daher müssen wir das Google Play-Services-SDK in der Android-Anwendung einrichten. Öffnen Sie Ihre `build.gradle` Datei (App-Modul) und fügen Sie das SDK des Google play-Services als Abhängigkeiten hinzu.

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

Fügen Sie den API-Schlüssel in der Manifest-Datei hinzu

Um die Google-API in einer Android-Anwendung verwenden zu können, müssen Sie der Datei AndroidManifest.xml den API-Schlüssel und die Version des Google Play-Diensts hinzufügen. Fügen Sie die richtigen Metadaten-Tags in das Tag der Datei AndroidManifest.xml ein.

Verbinden und autorisieren Sie die Google Drive Android API

Wir müssen die **Google Drive Android API** mit der Android-Anwendung authentifizieren und verbinden. Die Autorisierung der **Google Drive Android API** wird vom **GoogleApiClient übernommen** . Wir verwenden **GoogleApiClient** innerhalb der **onResume ()** -Methode.

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

Trennen Sie die Google Deive Android API

Wenn die Aktivität endet, trennen wir die Verbindung zwischen der Google Drive Android API-Verbindung und der Android-Anwendung, indem wir die Methode **disconnect () in der** Methode **onStop () der** Aktivität **aufrufen** .

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

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

Implementieren Sie Verbindungsrückrufe und einen Listener für fehlgeschlagene Verbindung

Wir implementieren Connection Callbacks und Connection Failed Listener des Google API-Clients in der Datei MainActivity.java, um den Status der Verbindung des Google API-Clients zu erfahren. Diese Listener bieten die **Methoden onConnected (), onConnectionFailed (), onConnectionSuspended ()** , um Verbindungsprobleme zwischen App und Laufwerk zu behandeln.

Wenn der Benutzer die Anwendung autorisiert hat, wird die **onConnected () -** Methode aufgerufen. Wenn der Benutzer keine Anwendung autorisiert hat, **wird die** Methode **onConnectionFailed ()** aufgerufen, und dem Benutzer wird ein Dialogfeld angezeigt, dass Ihre App nicht zum Zugriff auf Google Drive berechtigt ist. **Wenn die** Verbindung unterbrochen wird, **wird die** Methode **onConnectionSuspended ()** aufgerufen.

Sie müssen **ConnectionCallbacks** und **OnConnectionFailedListener** in Ihrer Aktivität implementieren. Verwenden Sie den folgenden Code in Ihrer Java-Datei.

```
@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");
    }
```

Erstellen Sie eine Datei in Google Drive

Wir werden eine Datei in Google Drive hinzufügen. Wir verwenden die `createFile()` -Methode eines `Drive` Objekts, um Dateien programmgesteuert in Google Drive zu erstellen. In diesem Beispiel fügen wir eine neue Textdatei im Stammordner des Benutzers hinzu. Wenn eine Datei hinzugefügt wird, müssen der erste Satz Metadaten, der Inhalt der Datei und der übergeordnete Ordner angegeben werden.

Wir müssen eine `CreateMyFile()` Callback-Methode erstellen und innerhalb dieser Methode das `Drive` Objekt verwenden, um eine `DriveContents` Ressource `DriveContents` . Dann übergeben wir den API-Client an das `Drive` Objekt und rufen die `driveContentsCallback` Callback-Methode auf, um das Ergebnis von `DriveContents` .

Eine `DriveContents` Ressource enthält eine temporäre Kopie des binären Datenstroms der Datei, die nur für die Anwendung verfügbar ist.

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

Result Handler von DriveContents

Für die Bearbeitung der Antwort muss geprüft werden, ob der Anruf erfolgreich war oder nicht. Wenn der Aufruf erfolgreich war, können wir die `DriveContents` Ressource `DriveContents` .

Wir erstellen einen Ergebnis-Handler für `DriveContents` . In dieser Methode rufen wir die `CreateFileOnGoogleDrive()` Methode auf und übergeben das Ergebnis von `DriveContentsResult` :

```
/**
 * This is the Result result handler of Drive contents.
 * This callback method calls the CreateFileOnGoogleDrive() method.
 */
final ResultCallback<DriveContentsResult> driveContentsCallback =
         new ResultCallback<DriveContentsResult>() {
            @Override
            public void onResult(DriveContentsResult result) {
                if (result.getStatus().isSuccess()) {
                    if (fileOperation == true){
                        CreateFileOnGoogleDrive(result);
                    }
                }
            }
        };
```

Erstellen Sie eine Datei programmgesteuert

Um Dateien zu erstellen, müssen wir ein `MetadataChangeSet` Objekt verwenden. Mit diesem Objekt legen wir den Titel (Dateinamen) und den Dateityp fest. Außerdem müssen wir die `createFile()` Methode der `DriveFolder` Klasse verwenden und die Google-Client-API, das `MetaDataChangeSet` Objekt und das `driveContents` , um eine Datei zu erstellen. Wir rufen den Callback des Ergebnishandlers auf, um das Ergebnis der erstellten Datei zu behandeln.

Wir verwenden den folgenden Code, um eine neue Textdatei im Stammverzeichnis des Benutzers zu erstellen:

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

Ergebnis der erstellten Datei behandeln

Der folgende Code erstellt eine Callback-Methode, um das Ergebnis der erstellten Datei zu verarbeiten:

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

Google Drive API online lesen: https://riptutorial.com/de/android/topic/10646/google-drive-api

Kapitel 112: Google Maps API v2 für Android

Parameter

Parameter	Einzelheiten
Google Karte	Die `GoogleMap` ist ein Objekt, das bei einem `onMapReady()` empfangen wird
MarkerOptions	`MarkerOptions` ist die Builder-Klasse eines `Marker` und wird zum Hinzufügen eines Markers zu einer Karte verwendet.

Bemerkungen

Bedarf

1. Google Play Services-SDK installiert.
2. Ein Google Console-Konto.
3. Ein Google Maps-API-Schlüssel, der in Google Console abgerufen wird.

Examples

Standardmäßige Google Map-Aktivität

Dieser Aktivitätscode bietet grundlegende Funktionen zum Hinzufügen einer Google Map mithilfe eines SupportMapFragment.

Die Google Maps V2-API bietet eine völlig neue Möglichkeit zum Laden von Karten.

Aktivitäten müssen jetzt die **OnMapReadyCallBack-** Schnittstelle implementieren, die mit einer **onMapReady ()** -Methode überschrieben wird, die bei jeder Ausführung von SupportMapFragment ausgeführt **wird** . **getMapAsync (OnMapReadyCallback)** ; und der Anruf wurde erfolgreich abgeschlossen.

Karten verwenden **Marker** , **Polygone** und **PolyLines** , um dem Benutzer interaktive Informationen **anzuzeigen** .

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));
    }
}
```

Beachten Sie, dass der obige Code ein Layout aufbläst, das ein im Container-Layout verschachteltes SupportMapFragment enthält, das mit der ID `R.id.map` . Die Layoutdatei wird unten gezeigt:

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>
```

Benutzerdefinierte Google Map-Stile

Kartenstil

Google Maps enthält eine Reihe verschiedener Stile, die mithilfe dieses Codes angewendet werden können:

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

Die verschiedenen Kartenstile sind:

Normal

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

Typische Straßenkarte. Straßen, einige von Menschen gemachte Merkmale und wichtige natürliche Merkmale wie Flüsse werden gezeigt. Straßen- und Feature-Labels sind ebenfalls sichtbar.

Hybrid

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

Satellitenfotodaten mit hinzugefügten Straßenkarten. Straßen- und Feature-Labels sind ebenfalls sichtbar.

Satellit

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

Satellitenfotodaten. Straßen- und Feature-Labels sind nicht sichtbar.

Terrain

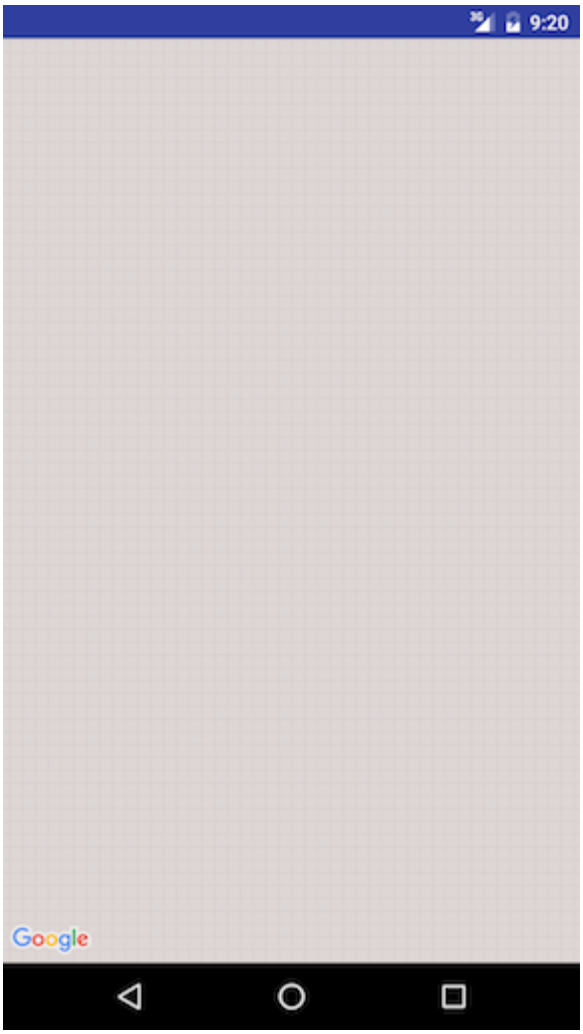
```
map.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

Topographische Daten. Die Karte enthält Farben, Konturlinien und Beschriftungen sowie Perspektivschattierungen. Einige Straßen und Beschriftungen sind ebenfalls sichtbar.

Keiner

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

Keine Fliesen Die Karte wird als leeres Raster ohne geladene Kacheln dargestellt.



ANDERE STYLE-OPTIONEN

Indoor-Karten

Bei hohen Zoomstufen zeigt die Karte Grundrisse für Innenräume. Diese Karten werden als Indoor-Karten bezeichnet und nur für die Kartentypen "Normal" und "Satellit" angezeigt.

Um Indoor-Karten zu aktivieren oder zu deaktivieren, gehen Sie folgendermaßen vor:

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

Wir können benutzerdefinierte Karten zu Karten hinzufügen.

Fügen Sie in der onMapReady-Methode den folgenden Code-Ausschnitt hinzu

```
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);
    }
```

unter *res* ordner erstellen sie einen ordnernamen *raw* und fügen sie die json-datei hinzu.
Beispieldatei style.json

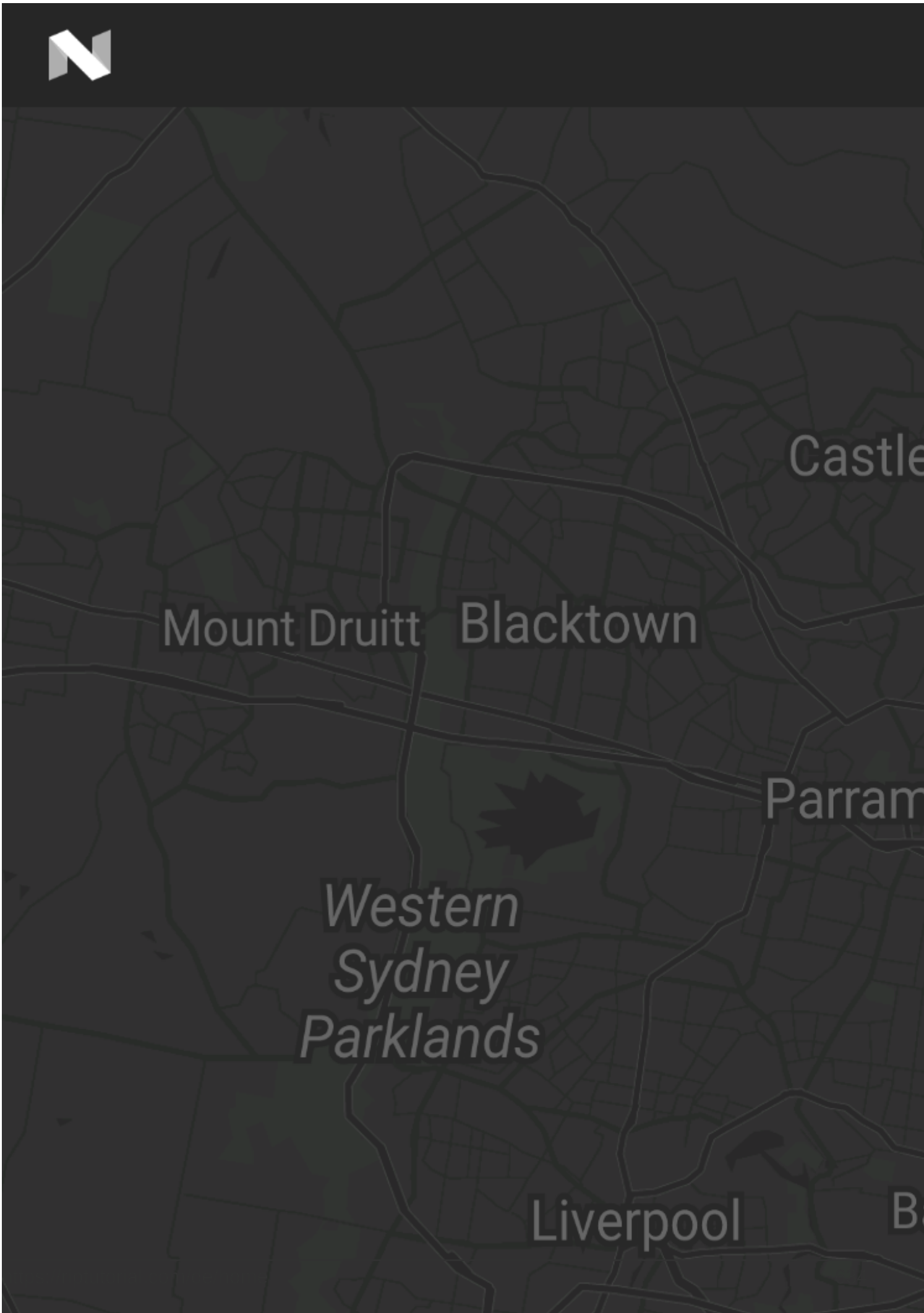
```
  [
{
  "featureType": "all",
  "elementType": "geometry",
  "stylers": [
    {
      "color": "#242f3e"
    }
  ]
},
{
  "featureType": "all",
  "elementType": "labels.text.stroke",
  "stylers": [
    {
      "lightness": -80
    }
  ]
},
{
  "featureType": "administrative",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#746855"
    }
  ]
},
{
  "featureType": "administrative.locality",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#d59563"
    }
  ]
},
{
  "featureType": "poi",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#d59563"
    }
  ]
},
{
  "featureType": "poi.park",
  "elementType": "geometry",
  "stylers": [
```

```
    {
      "color": "#263c3f"
    }
  ]
},
{
  "featureType": "poi.park",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#6b9a76"
    }
  ]
},
{
  "featureType": "road",
  "elementType": "geometry.fill",
  "stylers": [
    {
      "color": "#2b3544"
    }
  ]
},
{
  "featureType": "road",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#9ca5b3"
    }
  ]
},
{
  "featureType": "road.arterial",
  "elementType": "geometry.fill",
  "stylers": [
    {
      "color": "#38414e"
    }
  ]
},
{
  "featureType": "road.arterial",
  "elementType": "geometry.stroke",
  "stylers": [
    {
      "color": "#212a37"
    }
  ]
},
{
  "featureType": "road.highway",
  "elementType": "geometry.fill",
  "stylers": [
    {
      "color": "#746855"
    }
  ]
},
{
  "featureType": "road.highway",
```

```
    "elementType": "geometry.stroke",
    "stylers": [
      {
        "color": "#1f2835"
      }
    ]
  },
  {
    "featureType": "road.highway",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#f3d19c"
      }
    ]
  },
  {
    "featureType": "road.local",
    "elementType": "geometry.fill",
    "stylers": [
      {
        "color": "#38414e"
      }
    ]
  },
  {
    "featureType": "road.local",
    "elementType": "geometry.stroke",
    "stylers": [
      {
        "color": "#212a37"
      }
    ]
  },
  {
    "featureType": "transit",
    "elementType": "geometry",
    "stylers": [
      {
        "color": "#2f3948"
      }
    ]
  },
  {
    "featureType": "transit.station",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#d59563"
      }
    ]
  },
  {
    "featureType": "water",
    "elementType": "geometry",
    "stylers": [
      {
        "color": "#17263c"
      }
    ]
  },
```

```
  {
    "featureType": "water",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#515c6d"
      }
    ]
  },
  {
    "featureType": "water",
    "elementType": "labels.text.stroke",
    "stylers": [
      {
        "lightness": -20
      }
    ]
  }
]
```

Klicken Sie auf diesen Link, um eine Styles-Json-Datei zu erstellen


Castle
Mount Druitt
Blacktown
Parram
Western
Sydney
Parklands
Liverpool
B

`MyLocation` Objects, kann auf diese Weise erfolgen.

Die `MyLocation` :

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

Hier ist eine Methode, die eine Liste von `MyLocation` Objekten `MyLocation` und für jedes `MyLocation` einen Marker setzt:

```
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));
 }
}
```

Hinweis: In diesem Beispiel ist `mMap` eine Klassenmitgliedsvariable der Activity, in der wir sie der in der `onMapReady()` Überschreibung erhaltenen `onMapReady()` zugewiesen haben.

MapView: Einbetten einer GoogleMap in ein vorhandenes Layout

Es ist möglich, eine GoogleMap als Android-Ansicht zu behandeln, wenn Sie die bereitgestellte MapView-Klasse verwenden. Die Verwendung ist MapFragment sehr ähnlich.

Verwenden Sie MapView in Ihrem Layout wie folgt:

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

Ihre Aktivität muss die OnMapReadyCallback-Schnittstelle implementieren, um zu funktionieren:

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

Aktuellen Standort in Google Map anzeigen

Hier ist eine vollständige Aktivitätsklasse, die einen Marker an der aktuellen Position platziert und die Kamera auch an die aktuelle Position bewegt.

Es gibt ein paar Dinge, die hier nacheinander ablaufen:

- Überprüfen Sie die Standortberechtigung
- Wenn die Standortberechtigung erteilt wurde, rufen Sie `setMyLocationEnabled()` , erstellen den GoogleApiClient und stellen eine Verbindung her
- Wenn der GoogleApiClient verbunden ist, fordern Sie die Standortaktualisierungen an

```
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.
```

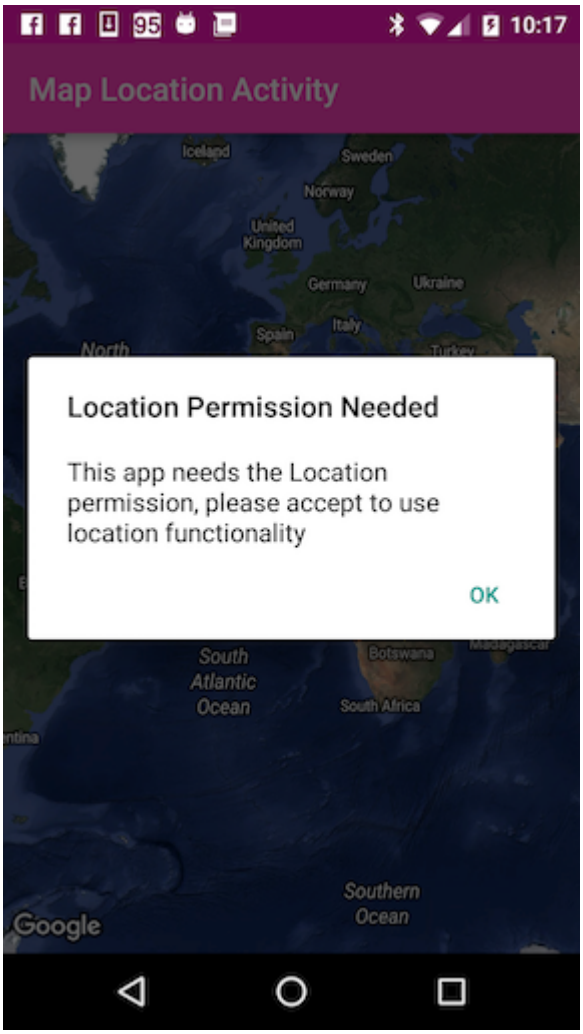
```
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                    // 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:

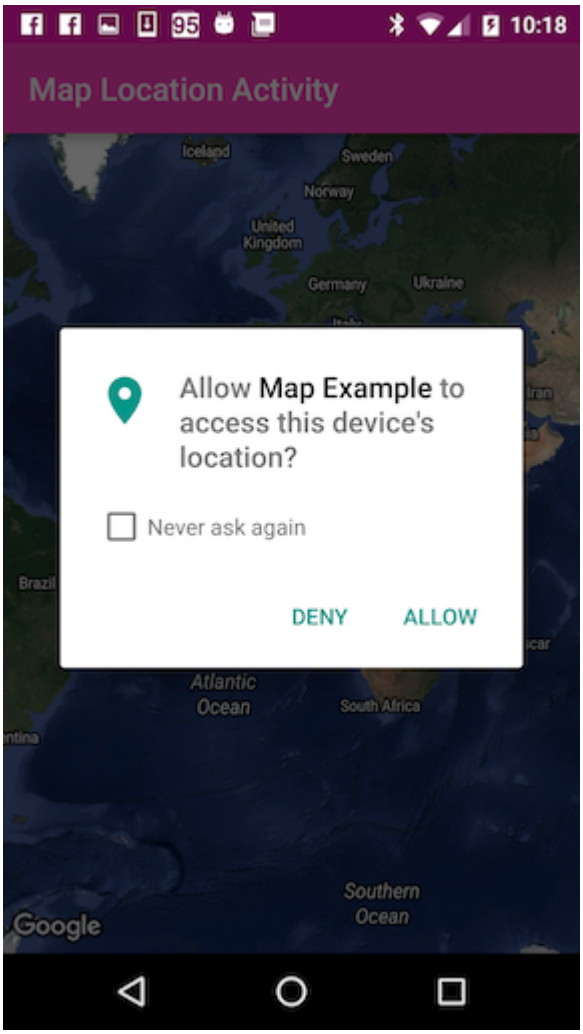
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <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>
```

Ergebnis:

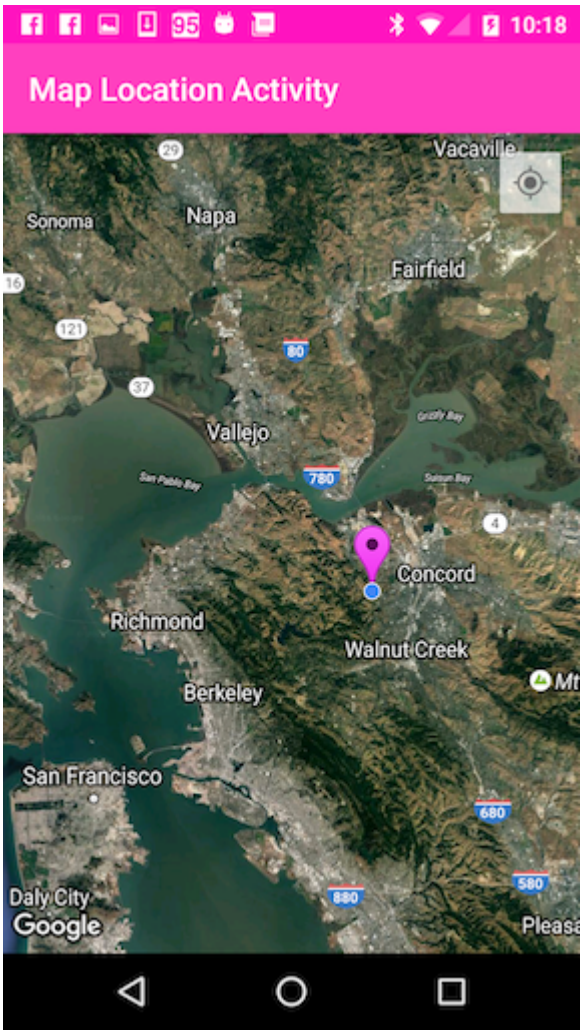
Zeigen Sie ggf. eine Erläuterung zu Marshmallow und Nougat mithilfe eines AlertDialogs an (dieser Fall tritt auf, wenn der Benutzer zuvor eine Berechtigungsanforderung abgelehnt hat oder die Berechtigung erteilt und diese später in den Einstellungen widerrufen hat):



Fordern Sie den Benutzer zur Eingabe der Standortberechtigung für Marshmallow und Nougat auf, indem Sie `ActivityCompat.requestPermissions()` aufrufen:



Bewegen Sie die Kamera an den aktuellen Standort und platzieren Sie den Marker, wenn die Standortberechtigung erteilt wurde:



Den SH1-Fingerprint Ihrer Zertifikatsschlüsselspeicherdatei erhalten

Um einen Google Maps API-Schlüssel für Ihr Zertifikat zu erhalten, müssen Sie der SHI-Fingerabdruck Ihres Debug- / Release-Keystores der API-Konsole mitteilen.

Den Keystore erhalten Sie mit dem **Keytool-** Programm des **JDK,** wie hier in den Dokumenten beschrieben.

Ein anderer Ansatz besteht darin, den Fingerabdruck programmgesteuert abzurufen, indem Sie dieses Snippet mit Ihrer mit dem Debug- / Release-Zertifikat signierten App ausführen und den Hash im Protokoll drucken.

```
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());
}
```

Starten Sie Google Maps nicht, wenn Sie auf die Karte klicken (Lite-Modus).

Wenn eine Google Map im Lite-Modus angezeigt wird, wird durch Klicken auf eine Karte die Google Maps-Anwendung geöffnet. Um diese Funktion zu deaktivieren, müssen Sie `setClickable(false)` in `MapView` , *z. B .* :

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

UISettings

Mit `UISettings` kann das Erscheinungsbild der Google Map geändert werden.

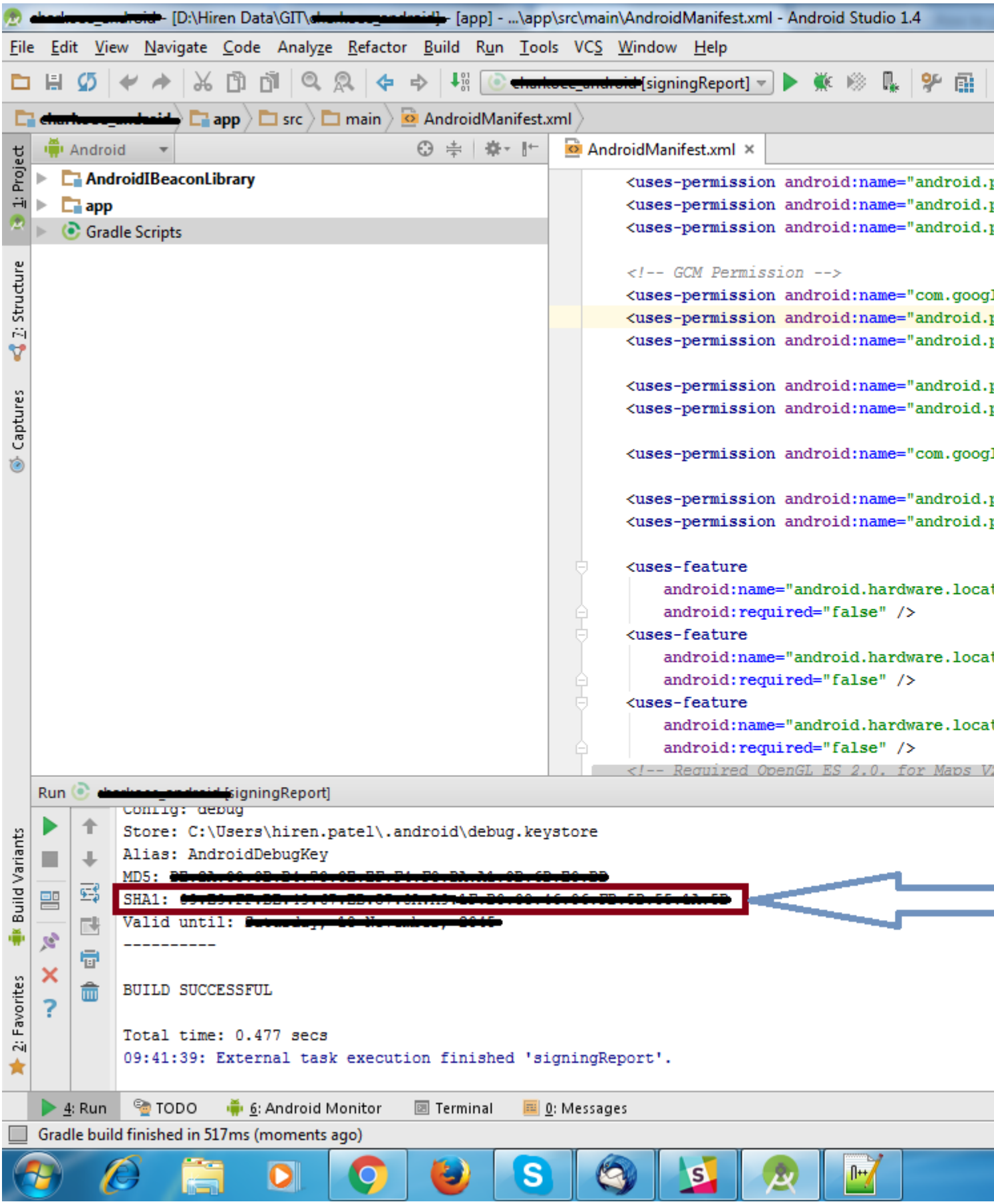
Hier ein Beispiel einiger allgemeiner Einstellungen:

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

Ergebnis:

Holen Sie sich den SHA1-Fingerabdruck

1. Öffnen Sie Android Studio
2. Öffnen Sie Ihr Projekt
3. Klicken Sie auf Gradle (Von der rechten Seite wird **Gradle Bar angezeigt**)
4. Klicken Sie auf Aktualisieren (Klicken Sie auf Aktualisieren von **Gradle Bar,** werden Sie **Liste** Gradle Skripte Ihres Projekts sehen)
5. Klicken Sie auf Ihr Projekt (Projektname Form **Liste** (root))
6. Klicken Sie auf Aufgaben
7. Klicken Sie auf Android
8. Klicken Sie doppelt auf signingReport (Sie **SHA1** und **MD5** in **Run Bar** erhalten)



InfoWindow Klicken Sie auf Listener

Hier ein Beispiel, wie Sie für jedes InfoWindow-Klickereignis jedes Markers eine andere Aktion

definieren.

Verwenden Sie eine HashMap, in der die Markierungs-ID der Schlüssel ist und der Wert die entsprechende Aktion ist, die beim Klicken auf das InfoWindow-Objekt ausgeführt werden soll.

Verwenden Sie dann einen `OnInfoWindowClickListener` , um das Ereignis eines Benutzers zu bearbeiten, der auf das InfoWindow klickt, und mithilfe der HashMap zu bestimmen, welche Aktion ausgeführt werden soll.

In diesem einfachen Beispiel werden wir eine andere Aktivität öffnen, basierend auf dem Marker-InfoWindow.

Deklarieren Sie die HashMap als Instanzvariable der Aktivität oder des Fragments:

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

Nehmen Sie dann jedes Mal, wenn Sie einen Marker hinzufügen, einen Eintrag in der HashMap mit der Marker-ID und der Aktion auf, die ausgeführt werden soll, wenn auf InfoWindow geklickt wird.

Fügen Sie beispielsweise zwei Markierungen hinzu und definieren Sie eine Aktion, die für jede Aktion ausgeführt werden soll:

```
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");
```

Rufen Sie im InfoWindow-Klicklistener die Aktion aus der HashMap ab und öffnen Sie die entsprechende Aktivität basierend auf der Aktion des Markers:

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

Hinweis: Wenn sich der Code in einem Fragment befindet, ersetzen Sie MainActivity.this durch getActivity ().

Offset ändern

Indem Sie die *x*- und *y*- Werte der Mappoint-Objekte nach Bedarf ändern, können Sie die Offset-Position von Google Map ändern. Standardmäßig wird sie in der Mitte der Kartenansicht angezeigt. Rufen Sie die Methode an, wo Sie sie ändern möchten! Verwenden Sie es besser in Ihrem `onLocationChanged` wie `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

        }
```

Google Maps API v2 für Android online lesen: https://riptutorial.com/de/android/topic/170/google-maps-api-v2-fur-android

Kapitel 113: Google Play Store

Examples

Öffnen Sie den Google Play Store-Eintrag für Ihre App

Das folgende Codefragment zeigt, wie Sie die Google Play Store-Auflistung Ihrer App auf sichere Weise öffnen. Normalerweise möchten Sie es verwenden, wenn Sie den Benutzer auffordern, eine Bewertung für Ihre App zu hinterlassen.

```
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);
}
```

Hinweis : Der Code öffnet den Google Play Store, wenn die App installiert ist. Ansonsten wird nur der Webbrowser geöffnet.

Öffnen Sie den Google Play Store mit der Liste aller Anwendungen Ihres Publisher-Kontos

Sie können in Ihrer App eine Schaltfläche "Unsere anderen Apps durchsuchen" hinzufügen, in der alle Ihre Anwendungen (Publisher) in der Google Play Store-App aufgeführt sind.

```
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);
    }
}
```

Google Play Store online lesen: https://riptutorial.com/de/android/topic/10900/google-play-store

Kapitel 114: Google-Anmeldung integrieren

Syntax

- newInstance () - Zum Erstellen einer einzelnen Instanz von Google Helper
- initGoogleSignIn () - Zum Initialisieren der Google-Anmeldung
- getGoogleAccountDetails () - Um in Kontodetails angemeldet zu werden
- signOut () - Zum Abmelden des Benutzers
- getGoogleClient () - Um GoogleApiClient zu verwenden

Parameter

Parameter	Detail
ETIKETT	Ein String, der während der Protokollierung verwendet wird
GoogleSignInHelper	Eine statische Referenz für Helfer
AppCompatActivity	Eine Aktivitätsreferenz
GoogleApiClient	Eine Referenz von GoogleAPIClient
RC_SIGN_IN	Eine ganze Zahl repräsentiert das Aktivitätsergebnis konstant
isLoggingOut	Ein boolescher Wert, um zu prüfen, ob eine Abmeldeaufgabe ausgeführt wird

Examples

Google Melden Sie sich mit der Helper-Klasse an

`build.gradle` unten zu deinem `build.gradle` aus `android` Tag hinzu:

```
// Apply plug-in to app.
apply plugin: 'com.google.gms.google-services'
```

Fügen Sie Ihrem Hilfspaket die folgende Helper-Klasse hinzu:

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

Fügen Sie Ihrer `OnActivityResult` in der Aktivitätsdatei den folgenden Code `OnActivityResult` :

```
 // [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]
```

Google-Anmeldung integrieren online lesen: https://riptutorial.com/de/android/topic/2837/google-anmeldung-integrieren

Kapitel 115: Google-Anmeldungsintegration auf Android

Einführung

Dieses Thema basiert auf So integrieren Sie Google-Anmeldung und Android-Apps

Examples

Integration von Google Auth in Ihr Projekt. (Holen Sie sich eine Konfigurationsdatei)

Rufen Sie zuerst die Konfigurationsdatei für die Anmeldung von auf

Link unten öffnen

[https://developers.google.com/identity/sign-in/android/start-integrating(

Klicken Sie auf Get A configuration file

- Geben Sie den App-Namen und den Paketnamen ein und klicken Sie auf Dienste auswählen und konfigurieren
- SHA1 bereitstellen Aktivieren Sie Google SIGNIN und generieren Sie Konfigurationsdateien

Laden Sie die Konfigurationsdatei herunter und legen Sie die Datei in App / Ordner Ihres Projekts ab

1. Fügen Sie die Abhängigkeit Ihrem build.gradle auf Projektebene hinzu:

   Klassenpfad "com.google.gms: google-services: 3.0.0"

2. Fügen Sie das Plugin zu Ihrem build.gradle auf App-Ebene hinzu: (unten)

   Plugin anwenden: 'com.google.gms.google-services'

3. Fügen Sie diese Abhängigkeit zu Ihrer App-Gradlendatei hinzu

   Abhängigkeiten {compile 'com.google.android.gms: play-services-auth: 9.8.0'}

Code-Implementierung Google SignIn

- Konfigurieren Sie in der onCreate-Methode Ihrer Anmeldeaktivität Google Sign-In, um die für Ihre App erforderlichen Benutzerdaten anzufordern.

```
 GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
```

```
        .requestEmail()
        .build();
```

- Erstellen Sie ein GoogleApiClient-Objekt mit Zugriff auf die Google-Anmelde-API und die von Ihnen angegebenen Optionen.

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
        .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
        .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
        .build();
```

- Wenn der Benutzer jetzt auf die Google-Anmeldeschaltfläche klickt, rufen Sie diese Funktion auf.

```
 private void signIn() {
 Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
 startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

- Implementieren Sie OnActivityResult, um die Antwort zu erhalten.

```
 @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);
    }
}
```

- Letzter Schritt Behandle das Ergebnis und erhalte Benutzerdaten

```
 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);
 }
}
```

Google-Anmeldungsintegration auf Android online lesen:
https://riptutorial.com/de/android/topic/9960/google-anmeldungsintegration-auf-android

Kapitel 116: Gradle für Android

Einführung

Gradle ist ein JVM-basiertes Buildsystem, mit dem Entwickler übergeordnete Skripts schreiben können, mit denen der Kompilierungs- und Anwendungsprozess automatisiert werden kann. Es ist ein flexibles Plugin-basiertes System, mit dem Sie verschiedene Aspekte des Erstellungsprozesses automatisieren können. einschließlich der Zusammenstellung und eine Unterzeichnung `.jar` , Herunterladen und Verwalten von externen Abhängigkeiten, Injizieren Felder in die `AndroidManifest` oder spezifische SDK - Versionen verwendet wird .

Syntax

- `apply plugin` : Die Plugins, die normalerweise verwendet werden sollten, sind nur `'com.android.application'` oder `'com.android.library'` .
- `android` : Die Hauptkonfiguration Ihrer App
    - `compileSdkVersion` : Die kompilierte SDK-Version
    - `buildToolsVersion` : Die Version der Build-Tools
    - `defaultConfig` : Die Standardeinstellungen, die von Flavors und Build-Typen überschrieben werden können
        - `applicationId` : Die Anwendungs-ID, die Sie zB im PlayStore verwenden, stimmt im Wesentlichen mit Ihrem Paketnamen überein
        - `minSdkVersion` : Die minimal erforderliche SDK-Version
        - `targetSdkVersion` : Die SDK-Version, für die Sie kompilieren (sollte immer die neueste sein)
        - `versionCode` : Die interne Versionsnummer, die bei jedem Update größer sein muss
        - `versionName` : Die Versionsnummer, die der Benutzer auf der App- `versionName` sehen kann
    - `buildTypes` : Siehe woanders (TODO)
- `dependencies` : Die Maven oder lokalen Abhängigkeiten Ihrer App
    - eine einzelne Abhängigkeit `compile`
    - `testCompile` : eine Abhängigkeit für die Einheiten- oder Integrationstests

Bemerkungen

Siehe auch

- Die offizielle Gradle-Homepage
- So konfigurieren Sie Gradle-Builds
- Das Android-Plugin für Gradle

- Android Gradle DSL

Gradle für Android - Erweiterte Dokumentation:

Es gibt ein weiteres Tag, wo Sie weitere Themen und Beispiele zur Verwendung von Gradle in Android finden können.
http://www.riptutorial.com/topic/2092

Examples

Eine grundlegende build.gradle-Datei

Dies ist ein Beispiel für eine Standarddatei `build.gradle` in einem Modul.

```
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 (domänenspezifische Sprache)

Jeder Block in der obigen Datei wird als `DSL` (domänenspezifische Sprache) bezeichnet.

Plugins

Die erste Zeile, `apply plugin: 'com.android.application'` , wendet das Android-Plugin für Gradle auf den Build an und macht den `android {}` -Block verfügbar, um Android-spezifische Build-Optionen zu deklarieren.

Für eine **Android-Anwendung** :

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

Für eine **Android-Bibliothek** :

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

Die DSLs im obigen Beispiel verstehen

Der zweite Teil, der `android {...}` Block `android {...}` , ist der Android- `DSL` der Informationen zu Ihrem Projekt enthält.

Sie können beispielsweise die `compileSdkVersion` die die Android-API-Ebene angibt.
`compileSdkVersion` sollte von Gradle zum Kompilieren Ihrer App verwendet werden.
Der `defaultConfig` enthält die `defaultConfig` für Ihr Manifest. Sie können sie mit Produkt-Flavors `override` .

Weitere Informationen finden Sie in diesen Beispielen:

- DSL für das App-Modul
- Typen erstellen
- Produkt-Geschmacksrichtungen
- Einstellungen signieren

Abhängigkeiten

Der `dependencies` wird außerhalb des `android` Blocks `{...}` : Dies bedeutet, dass er nicht vom Android-Plugin definiert wird, sondern der Standardgradle.
Der `dependencies` gibt an, welche externen Bibliotheken (normalerweise Android-Bibliotheken, aber

auch Java-Bibliotheken sind gültig) in Ihre App aufgenommen werden sollen. Gradle lädt diese Abhängigkeiten automatisch herunter (wenn keine lokale Kopie verfügbar ist). Sie müssen nur ähnliche `compile` hinzufügen `compile` wenn Sie eine andere Bibliothek hinzufügen möchten.

Schauen wir uns eine der hier dargestellten Zeilen an:

```
compile 'com.android.support:design:25.3.1'
```

Diese Zeile sagt im Grunde

> Fügen Sie meinem Projekt eine Abhängigkeit von der Android-Supportdesign-Bibliothek hinzu.

Gradle stellt sicher, dass die Bibliothek heruntergeladen und vorhanden ist, sodass Sie sie in Ihrer App verwenden können. Der Code wird auch in Ihre App aufgenommen.

Wenn Sie mit Maven vertraut sind, handelt es sich bei dieser Syntax um die *GroupId* , einen Doppelpunkt, *ArtifactId* , einen weiteren Doppelpunkt und die Version der Abhängigkeit, die Sie einschließen möchten. So haben Sie die vollständige Kontrolle über die Versionierung.

Es ist zwar möglich, Artefaktversionen mithilfe des Pluszeichens (+) anzugeben, dies sollte jedoch vermieden werden. Dies kann zu Problemen führen, wenn die Bibliothek ohne Ihr Wissen mit bahnbrechenden Änderungen aktualisiert wird, was zu Abstürzen in Ihrer App führen könnte.

Sie können verschiedene Arten von Abhängigkeiten hinzufügen:

- lokale binäre Abhängigkeiten
- Modulabhängigkeiten
- entfernte Abhängigkeiten

Ein besonderes Augenmerk sollte den aflachen Abhängigkeiten gewidmet werden .

Weitere Details finden Sie in diesem Thema.

Hinweis zu **-v7 in *appcompat-v7***

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

Dies bedeutet einfach, dass diese **Bibliothek** (`appcompat`) mit der Android-API-Ebene 7 und weiter kompatibel ist.

Anmerkung zum **junit: junit: 4.12**

Dies ist die Testabhängigkeit für Unit-Tests.

Festlegen von Abhängigkeiten für verschiedene Build-Konfigurationen

Sie können angeben, dass eine Abhängigkeit nur für eine bestimmte Build-Konfiguration verwendet werden soll, oder Sie können andere Abhängigkeiten für die Build-Typen oder die Produktvarianten (z. B. Debug, Test oder Release) `debugCompile` , indem Sie `debugCompile` , `testCompile` oder `releaseCompile` anstelle der üblichen `compile` .

Dies ist hilfreich, um Test- und Debug-abhängige Abhängigkeiten von Ihrem Release-Build fernzuhalten, wodurch Ihr Release- `APK` so gering wie möglich gehalten wird und sichergestellt wird, dass Debug-Informationen nicht zum Abrufen interner Informationen über Ihre App verwendet werden können.

---

signingConfig

Mit der `signingConfig` können Sie Ihren Gradle so konfigurieren, dass er `keystore` Informationen enthält und sicherstellen, dass die mit diesen Konfigurationen erstellten APK signiert und für die Play Store-Veröffentlichung bereit sind.

Hier finden Sie ein eigenes Thema .

Hinweis : Es wird jedoch nicht empfohlen, die Signaturdaten in Ihrer Gradle-Datei zu speichern. Um die `signingConfigs` zu entfernen, lassen Sie einfach den `signingConfigs` Teil aus.
Sie können sie auf verschiedene Arten angeben:

- Speichern in einer externen Datei
- Speichern Sie sie in Umgebungsvariablen .

Weitere Informationen finden Sie in diesem Thema: Unterzeichnen Sie APK, ohne das Keystore-Kennwort verfügbar zu machen .

Weitere Informationen zu Gradle für Android finden Sie im entsprechenden Gradle-Thema .

Produktgeschmack definieren

Produktvarianten werden in der `build.gradle` -Datei innerhalb des `android { ... }` Blocks `android { ... }` (siehe unten).

```
...
android {
    ...
    productFlavors {
        free {
            applicationId "com.example.app.free"
            versionName "1.0-free"
        }
        paid {
            applicationId "com.example.app.paid"
            versionName "1.0-paid"
        }
    }
```

```
}
```

Auf diese Weise haben wir jetzt zwei zusätzliche Produktvarianten: `free` und `paid` . Jeder kann seine spezifische Konfiguration und Attribute haben. Zum Beispiel hat unsere beiden neuen Geschmacksrichtungen einen separaten `applicationId` und `versionName` als unsere bestehenden `main` Geschmack (als Standard verfügbar sind , so hier nicht dargestellt).

Hinzufügen produktspezifischer Abhängigkeiten

Abhängigkeiten können für eine bestimmte Produktvariante hinzugefügt werden, ähnlich wie sie für bestimmte Build-Konfigurationen hinzugefügt werden können.

Nehmen Sie für dieses Beispiel an, dass wir bereits zwei Produktvarianten definiert haben, die als `free` und `paid` (mehr zur Definition von Geschmacksrichtungen hier).
Wir können dann die AdMob-Abhängigkeit für die `free` Variante und die Picasso-Bibliothek für die `paid` Version hinzufügen:

```
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'
}
...
```

Hinzufügen produktspezifischer Ressourcen

Ressourcen können für einen bestimmten Produktgeschmack hinzugefügt werden.

Nehmen Sie für dieses Beispiel an, dass wir bereits zwei Produktvarianten definiert haben, die als `free` und `paid` . Um produktspezifische Ressourcen hinzuzufügen, erstellen wir neben dem `main/res` Ordner weitere Ressourcenordner, die Sie wie üblich hinzufügen können. In diesem Beispiel definieren wir eine Zeichenfolge (`status`) für jede Produktvariante:

*/ 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 / **bezahlt** /res/values/strings.xml*

```
<resources>
    <string name="status">Paid</string>
</resources>
```

Die produktgeschmacksspezifischen `status` überschreiben den Wert für den `status` im `main` .

Definieren und Verwenden von Build-Konfigurationsfeldern

BuildConfigField

Mit Gradle können `buildConfigField` mit `buildConfigField` Linien Konstanten definieren. Auf diese Konstanten kann während der Laufzeit als statische Felder der `BuildConfig` Klasse `BuildConfig` . Dies kann zum Erstellen von Flavors verwendet werden, indem alle Felder im `defaultConfig` Block definiert und bei Bedarf für einzelne Build-Flavors überschrieben werden.

In diesem Beispiel wird das Erstellungsdatum definiert und der Build für die Produktion und nicht für den Test markiert:

```
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'
        }
    }
}
```

Der automatisch generierte <Paketname>. `BuildConfig` .java im Ordner gen enthält folgende Felder, die auf der obigen Anweisung basieren:

```
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";
}
```

Die definierten Felder können jetzt zur Laufzeit innerhalb der App verwendet werden, indem auf die generierte `BuildConfig` Klasse `BuildConfig` :

```
public void example() {
    // format the build date
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");
    String buildDate = dateFormat.format(new Date(BuildConfig.BUILD_DATE));
    Log.d("build date", buildDate);

    // do something depending whether this is a productive build
    if (BuildConfig.IS_PRODUCTION) {
        connectToProductionApiEndpoint();
    } else {
        connectToStagingApiEndpoint();
    }
}
```

ResValue

Der `resValue` in den `productFlavors` erstellt einen Ressourcenwert. Dabei kann es sich um einen beliebigen Ressourcentyp handeln (`string` , `dimen` , `color` usw.). Dies ist vergleichbar mit dem Definieren einer Ressource in der entsprechenden Datei: z. B. Definieren einer Zeichenfolge in einer Datei `strings.xml` . Der Vorteil besteht darin, dass der in Gradle definierte Wert basierend auf Ihrem productFlavor / buildVariant geändert werden kann. Um auf den Wert zuzugreifen, schreiben Sie den gleichen Code, als würden Sie aus der Ressourcendatei auf ein Res zugreifen:

```
getResources().getString(R.string.app_name)
```

Das Wichtigste ist, dass auf diese Weise definierte Ressourcen die in Dateien definierten Ressourcen nicht ändern können. Sie können nur neue Ressourcenwerte erstellen.

Einige Bibliotheken (z. B. Google Maps Android API) erfordern einen API-Schlüssel, der im Manifest als `meta-data` Tag bereitgestellt wird. Wenn für das Debugging und die Erstellung von Builds andere Schlüssel erforderlich sind, geben Sie einen durch Gradle ausgefüllten Platzhalter für Manifest an.

In Ihrer `AndroidManifest.xml` Datei:

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="${MAPS_API_KEY}"/>
```

`build.gradle` dann das Feld in Ihrer `build.gradle` -Datei entsprechend ein:

```
android {
    defaultConfig {
        ...
        // Your development key
        manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
    }

    productFlavors {
        prod {
            // Your production key
            manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
        }
    }
}
```

Das Android-Buildsystem generiert eine Reihe von Feldern automatisch und platziert sie in `BuildConfig.java` . Diese Felder sind:

Feld	Beschreibung
`DEBUG`	Ein `Boolean` Wert, der angibt, ob sich die App im Debug- oder Freigabemodus befindet
`APPLICATION_ID`	einen `String` mit der ID der Anwendung (zB `com.example.app`)
`BUILD_TYPE`	ein `String` , der den Build-Typ der Anwendung enthält (normalerweise `debug` oder `release`)
`FLAVOR`	Eine `String` die den bestimmten Geschmack des Builds enthält
`VERSION_CODE`	ein `int` das die Versionsnummer (Build-Nummer) enthält. Dies ist die gleiche wie `versionCode` in `build.gradle` oder `versionCode` in `AndroidManifest.xml`
`VERSION_NAME`	Ein `String` , der den Versionsnamen (Build) enthält. Dies ist das gleiche wie `versionName` in `build.gradle` oder `versionName` in `AndroidManifest.xml`

Wenn Sie mehrere Flavordimensionen definiert haben, hat jede Dimension ihren eigenen Wert. Wenn Sie zum Beispiel zwei Geschmacksdimensionen für `color` und `size` Sie auch die folgenden Variablen:

Feld	Beschreibung
`FLAVOR_color`	eine `String` die den Wert für den Farbton enthält.
`FLAVOR_size`	eine `String` die den Wert für den 'size'-Flavor enthält.

Abhängigkeiten über "dependencies.gradle" -Datei zentralisieren

Bei der Arbeit mit Projekten mit mehreren Modulen ist es hilfreich, Abhängigkeiten an einem einzigen Ort zu zentralisieren, anstatt sie auf viele Build-Dateien zu verteilen, insbesondere für gängige Bibliotheken wie die Android-Support-Bibliotheken und die Firebase-Bibliotheken .

Eine empfohlene Methode besteht darin, die Gradle-Build-Dateien mit einem `build.gradle` pro Modul sowie einer im Projektstammverzeichnis und einer für die Abhängigkeiten voneinander zu trennen. Beispiel:

```
root
  +- gradleScript/
  |     dependencies.gradle
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
```

Alle Ihre Abhängigkeiten können sich dann in `gradleScript/dependencies.gradle` :

```
ext {
    // Version
    supportVersion = '24.1.0'

    // Support Libraries dependencies
    supportDependencies = [
            design:            "com.android.support:design:${supportVersion}",
            recyclerView:      "com.android.support:recyclerview-v7:${supportVersion}",
            cardView:          "com.android.support:cardview-v7:${supportVersion}",
            appCompat:         "com.android.support:appcompat-v7:${supportVersion}",
            supportAnnotation: "com.android.support:support-annotations:${supportVersion}",
    ]

    firebaseVersion = '9.2.0';

    firebaseDependencies = [
            core:         "com.google.firebase:firebase-core:${firebaseVersion}",
            database:     "com.google.firebase:firebase-database:${firebaseVersion}",
            storage:      "com.google.firebase:firebase-storage:${firebaseVersion}",
            crash:        "com.google.firebase:firebase-crash:${firebaseVersion}",
            auth:         "com.google.firebase:firebase-auth:${firebaseVersion}",
            messaging:    "com.google.firebase:firebase-messaging:${firebaseVersion}",
            remoteConfig: "com.google.firebase:firebase-config:${firebaseVersion}",
            invites:      "com.google.firebase:firebase-invites:${firebaseVersion}",
            adMod:        "com.google.firebase:firebase-ads:${firebaseVersion}",
            appIndexing:  "com.google.android.gms:play-services-
appindexing:${firebaseVersion}",
    ];
}
```

Die kann dann aus dieser Datei in der obersten Datei `build.gradle` wie `build.gradle` werden:

```
// Load dependencies
apply from: 'gradleScript/dependencies.gradle'
```

und in der `module1/build.gradle` so:

```
// Module build file
dependencies {
    // ...
    compile supportDependencies.appCompat
    compile supportDependencies.design
    compile firebaseDependencies.crash
}
```

Ein anderer Ansatz

Ein weniger ausführlicher Ansatz zur Zentralisierung von Bibliotheksabhängigkeitsversionen kann erreicht werden, indem die Versionsnummer einmal als Variable deklariert und überall verwendet wird.

`build.gradle` Sie im Arbeitsbereich root `build.gradle` hinzu:

```
ext.v = [
    supportVersion:'24.1.1',
]
```

Fügen Sie in jedem Modul, das dieselbe Bibliothek verwendet, die benötigten Bibliotheken hinzu

```
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}"
```

Verzeichnisstruktur für geschmacksspezifische Ressourcen

Verschiedene Arten von Anwendungserstellungen können unterschiedliche Ressourcen enthalten. Um eine Flavour-spezifische Ressource zu erstellen, erstellen Sie im `src` Verzeichnis ein Verzeichnis mit dem Kleinbuchstaben Ihrer Flavour und fügen Sie Ihre Ressourcen auf die gleiche Weise hinzu, wie Sie es normalerweise tun würden.

Zum Beispiel hatten , wenn Sie einen Geschmack `Development` und wollten ein deutliches Launcher - Symbol dafür zur Verfügung zu stellen Sie ein Verzeichnis erstellen würden `src/development/res/drawable-mdpi` und in diesem Verzeichnis eine erstellen `ic_launcher.png` Datei mit entwicklungsspezifischen Symbol.

Die Verzeichnisstruktur sieht folgendermaßen aus:

```
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'
```

(In diesem Fall würden Sie natürlich auch Symbole für drawable-hdpi, drawable-xhdpi *usw.* erstellen).

Warum gibt es in einem Android Studio-Projekt zwei build.gradle-Dateien?

`<PROJECT_ROOT>\app\build.gradle` ist spezifisch für das **App-Modul** .

`<PROJECT_ROOT>\build.gradle` ist eine **"Build-Datei der obersten Ebene",** in der Sie Konfigurationsoptionen hinzufügen können, die für alle Unterprojekte / Module gelten.

Wenn Sie ein anderes Modul in Ihrem Projekt verwenden, haben Sie als lokale Bibliothek eine andere Datei `build.gradle` : `<PROJECT_ROOT>\module\build.gradle`

In der Datei der obersten Ebene können Sie allgemeine Eigenschaften als Buildscript-Block oder einige allgemeine Eigenschaften angeben.

```
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"
}
```

In `app\build.gradle` definieren Sie nur die Eigenschaften für das Modul:

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


android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion
}

dependencies {
    //.....
}
```

Ein Shell-Skript von Gradle ausführen

Ein Shellskript ist eine sehr vielseitige Möglichkeit, Ihren Build auf alles zu erweitern, was Sie sich vorstellen können.

Als Beispiel sei hier ein einfaches Skript zum Kompilieren von Protobuf-Dateien und zum Hinzufügen der Ergebnis-Java-Dateien zum Quellverzeichnis für die weitere Kompilierung aufgeführt:

```
def compilePb() {
    exec {
        // NOTICE: gradle will fail if there's an error in the protoc file...
        executable "../pbScript.sh"
    }
}

project.afterEvaluate {
    compilePb()
}
```

Das Shell-Skript 'pbScript.sh' für dieses Beispiel, das sich im Stammordner des Projekts befindet:

```
#!/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
```

Debuggen Ihrer Gradle-Fehler

Das Folgende ist ein Auszug aus Gradle - Was ist ein Exit-Wert ungleich Null und wie kann ich ihn beheben? Sehen Sie es für die vollständige Diskussion.

Nehmen wir an, Sie entwickeln eine Anwendung und erhalten einen Gradle-Fehler, der im Allgemeinen so aussieht.

```
: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
```

Sie suchen hier auf StackOverflow nach Ihrem Problem, und die Leute sagen, dass Sie Ihr Projekt bereinigen und neu erstellen oder MultiDex aktivieren sollten . Wenn Sie es versuchen, wird das

Problem dadurch nicht behoben .

Es gibt Möglichkeiten, mehr Informationen zu erhalten , aber die Gradle-Ausgabe selbst sollte auf den tatsächlichen Fehler in den wenigen Zeilen über dieser Meldung zwischen: `module:someTask FAILED` und dem letzten `:module:someOtherTask` , die bestanden haben. Wenn Sie also eine Frage zu Ihrem Fehler stellen, bearbeiten Sie Ihre Fragen, um dem Fehler mehr Kontext hinzuzufügen.

Sie erhalten also einen Exit-Wert ungleich Null. Nun, diese Zahl ist ein guter Indikator dafür, was Sie beheben sollten. Hier kommen einige am häufigsten vor.

- `1` ist nur ein allgemeiner Fehlercode und der Fehler ist wahrscheinlich in der Gradle-Ausgabe
- `2` scheint mit überlappenden Abhängigkeiten oder falscher Projektkonfiguration zu tun zu haben.
- `3` scheint zu viele Abhängigkeiten oder ein Speicherproblem zu haben.

Die allgemeinen Lösungen für das Vorstehende (nach dem Versuch eines Clean and Rebuild des Projekts) sind:

- `1` - Beheben Sie den genannten Fehler. Im Allgemeinen ist dies ein Fehler bei der Kompilierung, was bedeutet, dass ein Teil des Codes in Ihrem Projekt nicht gültig ist. Dies umfasst sowohl XML als auch Java für ein Android-Projekt.
- `2` & `3` - Viele Antworten geben an, Multidex zu aktivieren. Das Problem kann zwar behoben werden, es ist jedoch höchstwahrscheinlich eine Problemumgehung. Wenn Sie nicht verstehen, warum Sie es verwenden (siehe den Link), benötigen Sie es wahrscheinlich nicht. Bei allgemeinen Lösungen müssen Sie die übermäßige Nutzung von Bibliotheksabhängigkeiten reduzieren (z. B. alle Google Play-Services, wenn Sie nur eine Bibliothek verwenden müssen, wie z. B. Maps oder Anmelden).

Angeben verschiedener Anwendungs-IDs für Buildtypen und Produktvarianten

Sie können für jedes `buildType` oder `productFlavor` andere Anwendungs-IDs oder Paketnamen `productFlavor` indem Sie das Konfigurationsattribut **applicationIdSuffix** verwenden:

Beispiel für ein Suffix der `applicationId` für jeden `buildType` :

```
defaultConfig {
    applicationId "com.package.android"
    minSdkVersion 17
    targetSdkVersion 23
    versionCode 1
    versionName "1.0"
}

buildTypes {
    release {
        debuggable false
    }

    development {
        debuggable true
```

```
        applicationIdSuffix ".dev"
    }

    testing {
        debuggable true
        applicationIdSuffix ".qa"
    }
}
```

Unsere resultierenden `applicationIds` wären jetzt:

- com.package.android zur `release`
- com.package.android. **dev** für die `development`
- com.package.android. **qa** zum `testing`

Dies kann auch für `productFlavors` werden:

```
productFlavors {
    free {
        applicationIdSuffix ".free"
    }
    paid {
        applicationIdSuffix ".paid"
    }
}
```

Die resultierenden `applicationIds` wären:

- com.package.android. **kostenlos** für den `free` Geschmack
- com.package.android. für den **bezahlten** `paid` Geschmack

APK unterzeichnen, ohne das Keystore-Passwort freizulegen

Sie können die Signaturkonfiguration definieren, um die apk in der Datei `build.gradle` mit den folgenden Eigenschaften zu signieren:

- `storeFile` : die Keystore-Datei
- `storePassword` : das Schlüsselspeicherkennwort
- `keyAlias` : ein Aliasname
- `keyPassword` : Ein Schlüsselalias-Kennwort

In vielen Fällen müssen Sie diese Art von Informationen in der Datei `build.gradle` möglicherweise vermeiden.

Methode A: Konfigurieren Sie die Versionssignatur mit einer Datei keystore.properties

Es ist möglich , Ihre App zu konfigurieren `build.gradle` , so dass es Ihre Signatur - Konfigurationsinformationen von einer Eigenschaft wie Datei lesen `keystore.properties` .

Das Einrichten einer solchen Unterschrift ist vorteilhaft, weil:

- Die Informationen zur Signierungskonfiguration sind von Ihrer `build.gradle` Datei getrennt
- Sie müssen während des Signaturvorgangs nicht eingreifen, um Kennwörter für Ihre Keystore-Datei anzugeben
- Sie können die Datei `keystore.properties` problemlos von der Versionskontrolle ausschließen

Erstellen Sie zunächst eine Datei mit dem Namen `keystore.properties` im Stammverzeichnis Ihres Projekts mit folgendem Inhalt (Ersetzen der Werte durch Ihre eigenen):

```
storeFile=keystore.jks
storePassword=storePassword
keyAlias=keyAlias
keyPassword=keyPassword
```

`build.gradle` den `signingConfigs` Block nun in der `build.gradle` Datei Ihrer App wie folgt ein:

```
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']
            }
        }
    }
}
```

Das ist wirklich alles, **aber vergessen Sie nicht, sowohl Ihre Keystore-Datei als auch Ihre `keystore.properties` Datei von der Versionskontrolle auszuschließen** .

Ein paar Dinge zu beachten:

- Der `storeFile` Pfad in der angegebenen `keystore.properties` Datei sollte zu Ihrer App relativ seine `build.gradle` Datei. In diesem Beispiel wird davon `build.gradle` dass sich die Keystore-Datei im selben Verzeichnis wie die `build.gradle` -Datei der App `build.gradle` .
- In diesem Beispiel befindet sich die Datei `keystore.properties` im Stammverzeichnis des Projekts. Wenn Sie es an einer anderen Stelle `rootProject.file('keystore.properties')` , müssen Sie den Wert in `rootProject.file('keystore.properties')` relativ zum Stammverzeichnis Ihres Projekts in `rootProject.file('keystore.properties')` ändern.

Methode B: Mit einer Umgebungsvariablen

Dasselbe kann auch ohne eine Eigenschaftsdatei erreicht werden, wodurch das Passwort schwieriger zu finden ist:

```
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
    }
}
```

Die Umgebungsvariable `"ps"` kann global sein, es ist jedoch sicherer, sie nur der Shell von Android Studio hinzuzufügen.
In Linux kann dies durch Bearbeiten des `Desktop Entry` Eintrags von Android Studio erfolgen

```
Exec=sh -c "export ps=myPassword123 ; /path/to/studio.sh"
```

Weitere Details finden Sie in diesem Thema .

Versionierung Ihrer Builds über die Datei "version.properties"

Sie können Gradle verwenden, um die Paketversion bei jeder Erstellung automatisch zu erhöhen. Erstellen Sie dazu eine `version.properties` Datei in demselben Verzeichnis wie Ihr `build.gradle` mit dem folgenden Inhalt:

```
VERSION_MAJOR=0
VERSION_MINOR=1
VERSION_BUILD=1
```

(Ändern Sie die Werte für Dur und Minor nach Ihren Wünschen). `build.gradle` Sie dann in Ihrem `build.gradle` den folgenden Code zum `android` Abschnitt hinzu:

```
// 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)
  }
}
```

Auf die Informationen kann in Java als String `BuildConfig.VERSION_NAME` für die vollständige {major}. {Minor}. {Build} `BuildConfig.VERSION_CODE` und als Ganzzahl `BuildConfig.VERSION_CODE` nur für die Build-Nummer `BuildConfig.VERSION_CODE` .

Ändern des Ausgabe-APK-Namens und Hinzufügen des Versionsnamens:

Dies ist der Code zum Ändern des Dateinamens der Ausgabeanwendung (.apk). Der Name kann konfiguriert werden, indem Sie `newName` einen anderen Wert `newName`

```
android {

    applicationVariants.all { variant ->
        def newName = "ApkName";
        variant.outputs.each { output ->
            def apk = output.outputFile;

            newName += "-v" + defaultConfig.versionName;
            if (variant.buildType.name == "release") {
                newName += "-release.apk";
            } else {
                newName += ".apk";
            }
            if (!output.zipAlign) {
                newName = newName.replace(".apk", "-unaligned.apk");
            }

            output.outputFile = new File(apk.parentFile, newName);
            logger.info("INFO: Set outputFile to "
                        + output.outputFile
                        + " for [" + output.name + "]");
        }
    }
}
```

Deaktivieren Sie die Bildkomprimierung für eine kleinere APK-Dateigröße

Wenn Sie alle Bilder manuell optimieren, deaktivieren Sie APT Cruncher für eine kleinere APK-Dateigröße.

```
android {

    aaptOptions {
        cruncherEnabled = false
    }
```

```
}
```

Aktivieren Sie Proguard mit Gradle

Um die Proguard-Konfigurationen für Ihre Anwendung zu aktivieren, müssen Sie sie in der Gradle-Datei auf Modulebene aktivieren. Sie müssen den Wert von `minifyEnabled` auf `true` .

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

Mit dem obigen Code werden Ihre Proguard-Konfigurationen, die im Standard-Android-SDK in Kombination mit der Datei "proguard-rules.pro" in Ihrem Modul enthalten sind, auf Ihre freigegebene apk-Datei angewendet.

Aktivieren Sie die experimentelle Unterstützung für das NDK-Plugin für Gradle und AndroidStudio

Aktivieren und konfigurieren Sie das experimentelle Gradle-Plugin, um die NDK-Unterstützung von AndroidStudio zu verbessern. Stellen Sie sicher, dass Sie die folgenden Anforderungen erfüllen:

- Gradle 2.10 (für dieses Beispiel)
- Android NDK R10 oder höher
- Android SDK mit Build-Tools v19.0.0 oder höher

Konfigurieren Sie die Datei MyApp / build.gradle

Bearbeiten Sie die Zeile dependencies.classpath in build.gradle von zB

```
classpath 'com.android.tools.build:gradle:2.1.2'
```

zu

```
classpath 'com.android.tools.build:gradle-experimental:0.7.2'
```

(v0.7.2 war zum Zeitpunkt des Schreibens die neueste Version. Prüfen Sie die aktuelle Version selbst und passen Sie Ihre Zeile entsprechend an.)

Die build.gradle-Datei sollte folgendermaßen aussehen:

```
buildscript {
    repositories {
```

```
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle-experimental:0.7.2'
    }
}

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Konfigurieren Sie die Datei MyApp / app / build.gradle

Bearbeiten Sie die Datei build.gradle so, dass sie dem folgenden Beispiel ähnelt. Ihre Versionsnummern können anders aussehen.

```
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'])
}
```

Synchronisieren Sie und überprüfen Sie, ob die Gradle-Dateien fehlerfrei sind, bevor Sie fortfahren.

Testen Sie, ob das Plugin aktiviert ist

Stellen Sie zunächst sicher, dass Sie das Android NDK-Modul heruntergeladen haben. Erstellen Sie dann eine neue App in AndroidStudio und fügen Sie der ActivityMain-Datei Folgendes hinzu:

```
public class MainActivity implements Activity {
    onCreate() {
        // Pregenerated code. Not important here
    }
    static {
        System.loadLibrary("myLib");
    }
    public static native String getString();
}
```

Der `getString()` Teil sollte rot hervorgehoben werden, um `getString()` , dass die entsprechende JNI-Funktion nicht gefunden werden konnte. Bewegen Sie die Maus über den Funktionsaufruf, bis eine rote Glühbirne erscheint. Klicken Sie auf die Birne und wählen Sie `create function JNI_...` . Dies sollte eine myLib.c-Datei im Verzeichnis myApp / app / src / main / jni mit dem richtigen JNI-Funktionsaufruf generieren. Es sollte ähnlich aussehen:

```
#include <jni.h>

JNIEXPORT jstring JNICALL
Java_com_example_mydomain_myapp_MainActivity_getString(JNIEnv *env, jobject instance)
{
    // TODO

    return (*env)->NewStringUTF(env, returnValue);
}
```

Wenn es nicht so aussieht, wurde das Plugin nicht richtig konfiguriert oder der NDK wurde nicht heruntergeladen

Zeige alle gradle Projektaufgaben

```
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
```

Löschen Sie "nicht ausgerichtet" automatisch

Wenn Sie keine automatisch generierten APK-Dateien mit `unaligned` Suffix benötigen (was Sie wahrscheinlich nicht tun), können Sie den folgenden Code zur `build.gradle` Datei `build.gradle` :

```
// delete unaligned files
android.applicationVariants.all { variant ->
  variant.assemble.doLast {
    variant.outputs.each { output ->
        println "aligned " + output.outputFile
        println "unaligned " + output.packageApplication.outputFile

        File unaligned = output.packageApplication.outputFile;
        File aligned = output.outputFile
        if (!unaligned.getName().equalsIgnoreCase(aligned.getName())) {
            println "deleting " + unaligned.getName()
            unaligned.delete()
        }
    }
  }
}
```

Von hier aus

Build-Variante ignorieren

Aus einigen Gründen möchten Sie möglicherweise Ihre Build-Varianten ignorieren. Zum Beispiel: Sie haben ein "Mock" -Produktaroma und verwenden es nur für Debugging-Zwecke, wie z. B. Unit- / Instrumentationstests.

Lassen Sie uns die **mockRelease-** Variante aus unserem Projekt ignorieren. Öffnen **Sie die** Datei **build.gradle** und schreiben Sie:

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

Abhängigkeitsbaum sehen

Verwenden Sie die Aufgabenabhängigkeiten. Abhängig davon, wie Ihre Module eingerichtet sind, kann es sich entweder um `./gradlew dependencies` oder um die Abhängigkeiten der Modul-App zu sehen `./gradlew :app:dependencies`

Das Beispiel folgt der build.gradle -Datei

```
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'
}
```

erzeugt die folgende Grafik:

```
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 (*)

. . .
```

Hier können Sie sehen, dass das Projekt direkt `com.android.support:design` Version 23.2.1 enthält,

das selbst `com.android.support:support-v4` mit Version 23.2.1 enthält. `com.google.android.gms:play-services` selbst hängt jedoch von derselben `support-v4` Version `support-v4` jedoch von einer älteren Version 21.0.0, die von gradle erkannt wird.

`(*)` werden verwendet, wenn Gradle den Teilbaum überspringt, da diese Abhängigkeiten bereits zuvor aufgeführt wurden.

Verwenden Sie gradle.properties für zentrale Versionsnummern- / Buildkonfigurationen

Sie können zentrale Konfigurationsinformationen in definieren

- eine separate gradle include-Datei Zentralisierung von Abhängigkeiten über die Datei "dependencies.gradle"
- eine eigenständige Eigenschaftendatei Versionierung Ihrer Builds über die Datei "version.properties"

oder machen Sie es mit der root `gradle.properties` Datei

die Projektstruktur

```
root
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
  +- gradle.properties
```

globale Einstellung für alle Submodule in gradle.properties

```
# used for manifest
# todo increment for every release
appVersionCode=19
appVersionName=0.5.2.160726

# android tools settings
appCompileSdkVersion=23
appBuildToolsVersion=23.0.2
```

Verwendung in einem Submodul

```
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 {
    ...
}
```

Hinweis: Wenn Sie Ihre App im F-Droid App Store veröffentlichen möchten, müssen Sie in der Gradle-Datei magische Zahlen verwenden, da der f-Droid-Roboter die aktuelle Versionsnummer nicht lesen kann, um Versionsänderungen zu erkennen / zu überprüfen.

Signaturinformationen anzeigen

Unter bestimmten Umständen (z. B. beim Erwerb eines Google-API-Schlüssels) müssen Sie den Fingerabdruck Ihres Keystores ermitteln. Gradle hat eine praktische Aufgabe, die alle Signaturinformationen einschließlich der Fingerabdrücke des Keystores anzeigt:

```
./gradlew signingReport
```

Dies ist eine Beispielausgabe:

```
: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
----------
```

Build-Typen definieren

Sie können Build-Typen in der `build.gradle` Datei auf `build.gradle` im `android {}` -Block erstellen

und konfigurieren.

```
    android {
        ...
        defaultConfig {...}

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

            debug {
                applicationIdSuffix ".debug"
            }
        }
    }
```

Gradle für Android online lesen: https://riptutorial.com/de/android/topic/95/gradle-fur-android

Kapitel 117: GreenDAO

Einführung

GreenDAO ist eine Object-Relational Mapping-Bibliothek, die Entwicklern die Verwendung von SQLite-Datenbanken für den dauerhaften lokalen Speicher unterstützt.

Examples

Hilfsmethoden für SELECT-, INSERT-, DELETE-, UPDATE-Abfragen

Dieses Beispiel zeigt eine Hilfsklasse, die nützliche Methoden beim Ausführen der Datenabfragen enthält. Jede Methode hier verwendet Java Generic's, um sehr flexibel zu sein.

```
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();
}
```

Erstellen einer Entität mit GreenDAO 3.X, die einen zusammengesetzten Primärschlüssel enthält

Wenn Sie ein Modell für eine Tabelle erstellen, die einen zusammengesetzten Primärschlüssel enthält, ist für das Objekt Entity zusätzliche Arbeit erforderlich, damit die Entität des Modells diese Einschränkungen erfüllt.

Das folgende Beispiel für eine SQL-Tabelle und Entity veranschaulicht die Struktur zum Speichern einer Überprüfung, die ein Kunde für einen Artikel in einem Online-Shop hinterlassen hat. In diesem Beispiel möchten wir, dass die Spalten `customer_id` und `item_id` ein zusammengesetzter

Primärschlüssel sind, der nur eine Überprüfung zwischen einem bestimmten Kunden und einem Artikel `item_id` .

SQL-Tabelle

```
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)
);
```

Normalerweise würden wir die Annotationen `@Id` und `@Unique` über den jeweiligen Feldern in der Entitätsklasse verwenden. Für einen zusammengesetzten Primärschlüssel machen wir jedoch Folgendes:

1. Fügen Sie die `@Index` Annotation in die `@Entity` `@Index` Annotation auf Klassenebene ein. Die value-Eigenschaft enthält eine durch Kommas getrennte Liste der Felder, aus denen der Schlüssel besteht. Verwenden Sie die angegebene `unique` Eigenschaft, um die Eindeutigkeit des Schlüssels zu erzwingen.

2. GreenDAO erfordert jede Entität einen hat `long` oder `Long` Objekt als Primärschlüssel. Wir müssen dies immer noch zur Entity-Klasse hinzufügen, wir müssen es jedoch nicht verwenden oder sich keine Sorgen darüber machen, dass dies Auswirkungen auf unsere Implementierung hat. Im Beispiel unten heißt es `localID`

Entität

```
@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() {}
}
```

Erste Schritte mit GreenDao v3.X

Nachdem Sie die GreenDao-Bibliotheksabhängigkeit und das Gradle-Plugin hinzugefügt haben, müssen Sie zunächst ein Entitätsobjekt erstellen.

Entität

Eine Entität ist ein einfaches altes Java-Objekt *(POJO)* , das einige Daten in der Datenbank modelliert. GreenDao verwendet diese Klasse, um eine Tabelle in der SQLite-Datenbank zu erstellen und automatisch Hilfsklassen zu generieren, mit denen auf Daten zugegriffen werden kann, ohne dass SQL-Anweisungen geschrieben werden müssen.

```
@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...

}
```

Einmaliges GreenDao-Setup

Bei jedem Start einer Anwendung muss GreenDao initialisiert werden. GreenDao schlägt vor, diesen Code in einer Application-Klasse zu behalten oder irgendwo nur einmal ausgeführt zu werden.

```
DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this, "mydatabase", null);
db = helper.getWritableDatabase();
DaoMaster daoMaster = new DaoMaster(db);
DaoSession daoSession = daoMaster.newSession();
```

GreenDao-Hilfsklassen

Nachdem das Entitätsobjekt erstellt wurde, erstellt GreenDao automatisch die Hilfsklassen, die für die Interaktion mit der Datenbank verwendet werden. Diese werden ähnlich wie der Name des erstellten Entitätsobjekts benannt, gefolgt von `Dao` und werden vom `daoSession` Objekt abgerufen.

```
UsersDao usersDao = daoSession.getUsersDao();
```

Viele typische Datenbankaktionen können jetzt mit diesem Dao-Objekt mit dem Entitätsobjekt ausgeführt werden.

Abfrage

```
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();
```

Einfügen

```
Users newUser = new User("John","Doe","jdoe@example.com");
usersDao.insert(newUser);
```

Aktualisieren

```
// Modify a previously retrieved user object and update
user.setLastname("Dole");
usersDao.update(user);
```

Löschen

```
// Delete a previously retrieved user object
usersDao.delete(user);
```

GreenDAO online lesen: https://riptutorial.com/de/android/topic/1345/greendao

Kapitel 118: GreenRobot EventBus

Syntax

- @Subscribe (threadMode = ThreadMode.POSTING) public void onEvent (EventClass-Ereignis) {}

Parameter

Thread-Modus	Beschreibung
`ThreadMode.POSTING`	Wird in demselben Thread aufgerufen, in dem das Ereignis gepostet wurde. Dies ist der Standardmodus.
`ThreadMode.MAIN`	Wird im Haupt-UI-Thread aufgerufen.
`ThreadMode.BACKGROUND`	Wird in einem Hintergrundthread aufgerufen. Wenn der Beitragsthread nicht der Hauptthread ist, wird er verwendet. Wenn im Haupt-Thread `EventBus` hat `EventBus` einen einzigen Hintergrund-Thread, den es verwenden wird.
`ThreadMode.ASYNC`	Wird in einem eigenen Thread aufgerufen.

Examples

Ereignisobjekt erstellen

Zum Senden und Empfangen von Ereignissen benötigen wir zunächst ein Ereignisobjekt. Ereignisobjekte sind eigentlich einfache POJOs.

```
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;
    }
}
```

Ereignisse empfangen

Um Ereignisse empfangen zu können, müssen Sie Ihre Klasse im `EventBus` .

```
@Override
public void onStart() {
    super.onStart();
    EventBus.getDefault().register(this);
}

@Override
public void onStop() {
   EventBus.getDefault().unregister(this);
   super.onStop();
}
```

Und dann abonnieren Sie die Ereignisse.

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void handleEvent(ArbitaryEvent event) {
    Toast.makeText(getActivity(), "Event type: "+event.getEventType(),
Toast.LENGTH_SHORT).show();
}
```

Ereignisse senden

Das Senden von Ereignissen ist so einfach wie das Erstellen des Ereignisobjekts und das Versenden dieses Ereignisses.

```
EventBus.getDefault().post(new ArbitaryEvent(ArbitaryEvent.TYPE_1));
```

Ein einfaches Ereignis übergeben

Das erste, was wir tun müssen, füge EventBus zur Gradle-Datei unseres Moduls hinzu:

```
dependencies {
    ...
    compile 'org.greenrobot:eventbus:3.0.0'
    ...
}
```

Jetzt müssen wir ein Modell für unsere Veranstaltung erstellen. Es kann alles enthalten, was wir weitergeben möchten. Für jetzt machen wir nur eine leere Klasse.

```
public class DeviceConnectedEvent
{
}
```

Jetzt können wir den Code zu unserer `Activity` hinzufügen, der sich bei EventBus registriert und das Ereignis abonniert.

```
public class MainActivity extends AppCompatActivity
{
    private EventBus _eventBus;

    @Override
```

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

        _eventBus = EventBus.getDefault();
    }

    @Override
    protected void onStart ()
    {
        super.onStart();
        _eventBus.register(this);
    }

    @Override
    protected void onStop ()
    {
        _eventBus.unregister(this);
        super.onStop();
    }

    @Subscribe(threadMode = ThreadMode.MAIN)
    public void onDeviceConnected (final DeviceConnectedEvent event)
    {
        // Process event and update UI
    }
}
```

In dieser `Activity` wir eine Instanz von `EventBus` in der `onCreate()` Methode. Wir registrieren / unregistrieren für Ereignisse in `onStart()` / `onStop()` . Es ist wichtig, sich daran zu erinnern, dass die Registrierung aufgehoben wird, wenn der Zuhörer den Gültigkeitsbereich verliert oder Sie Ihre `Activity` verlieren könnten.

Schließlich definieren wir die Methode, die mit dem Ereignis aufgerufen werden soll. Die `@Subscribe` Annotation gibt EventBus an, nach welchen Methoden es für die Verarbeitung von Ereignissen suchen kann. Sie müssen mindestens eine Methode mit `@Subscribe` annotiert haben, um sich bei `@Subscribe` registrieren zu können. Andernfalls wird eine Ausnahme ausgelöst. In der Anmerkung definieren wir den Thread-Modus. Dadurch wird EventBus mitgeteilt, in welchem Thread die Methode aufgerufen werden soll. Dies ist eine sehr praktische Möglichkeit, Informationen aus einem Hintergrund-Thread an den UI-Thread zu übergeben! Genau das machen wir hier. `ThreadMode.MAIN` bedeutet, dass diese Methode im Haupt-UI-Thread von Android aufgerufen wird. `ThreadMode.MAIN` Sie hier alle erforderlichen UI-Manipulationen `ThreadMode.MAIN` . Der Name der Methode spielt keine Rolle. Der einzige `@Subscribe` , nach dem EventBus nach der `@Subscribe` Anmerkung sucht, ist der Typ des Arguments. Solange der Typ übereinstimmt, wird er aufgerufen, wenn ein Ereignis gepostet wird.

Das letzte, was wir tun müssen, um eine Veranstaltung zu posten. Dieser Code befindet sich in unserem `Service` .

```
EventBus.getDefault().post(new DeviceConnectedEvent());
```

Das ist alles dazu! EventBus nimmt das DeviceConnectedEvent und durchsucht seine

registrierten Listener, durchsucht die von ihnen abonnierten Methoden, sucht nach denen, die ein DeviceConnectedEvent als Argument akzeptieren, und ruft sie in dem Thread auf, für den sie aufgerufen werden sollen.

GreenRobot EventBus online lesen: https://riptutorial.com/de/android/topic/3551/greenrobot-eventbus

Kapitel 119: Gson

Einführung

Gson ist eine Java-Bibliothek, mit der Java-Objekte in ihre JSON-Darstellung konvertiert werden können. Gson betrachtet beide als sehr wichtige Designziele.

Gson-Eigenschaften:

Stellen Sie einfache `toJson()` und `fromJson()` Methoden `fromJson()` , um Java-Objekte in JSON und umgekehrt zu konvertieren

Zulassen, dass vorhandene, nicht veränderbare Objekte in und aus JSON konvertiert werden

Umfassende Unterstützung von Java Generics

Unterstützung beliebig komplexer Objekte (mit tiefen Vererbungshierarchien und umfangreicher Verwendung generischer Typen)

Syntax

- Ausschluss-Ausschluss ()
- FieldNamingStrategy fieldNamingStrategy ()
- <T> T von Json (JsonElement Json, Klasse <T> classOfT)
- <T> T von Json (JsonElement Json, Type TypeOfT)
- <T> T fromJson (JsonReader-Leser, Typ typeOfT)
- <T> T von Json (Reader Json, Class <T> classOfT)
- <T> T von Json (Reader Json, Type TypeOfT)
- <T> T fromJson (String json, Klasse <T> classOfT)
- <T> T fromJson (String json, Type typeOfT)
- <T> TypeAdapter <T> getAdapter (Klasse <T> -Typ)
- <T> TypeAdapter <T> getAdapter (TypeToken <T> -Typ)
- <T> TypeAdapter <T> getDelegateAdapter (TypeAdapterFactory skipPast, TypeToken <T> -Typ)
- JsonReader newJsonReader (Leser)
- JsonWriter newJsonWriter (Writer Writer)
- JsonElement toJsonTree (Object src)
- JsonElement toJsonTree (Object src, Typ typeOfSrc)
- boolean serializeNulls ()
- boolean htmlSafe ()
- Zeichenfolge toJson (JsonElement JsonElement)
- Zeichenfolge toJson (Object src)
- Zeichenfolge toJson (Object src, Type typeOfSrc)
- String toString ()
- void toJson (Object src, Type typeOfSrc, Appendable Writer)

- void toJson (Object src, Type typeOfSrc, JsonWriter-Schreiber)
- toJson ungültig (JsonElement JsonElement, Appendable Writer)
- toJson ungültig (JsonElement JsonElement, JsonWriter Writer)
- void toJson (Object src, Anhängbarer Schreiber)

Examples

Parsen von JSON mit Gson

Das Beispiel zeigt die Analyse eines JSON-Objekts mit der Gson-Bibliothek von Google .

Objekte analysieren:

```
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 = "";
    }


}
```

Verwenden Sie dann, wo eine Analyse erfolgen muss, Folgendes:

```
String robotJson = "{
                        \"version\": \"JellyBean\",
                        \"age\": 3,
                        \"robotName\": \"Droid\"
                    }";

Gson gson = new Gson();
Robot robot = gson.fromJson(robotJson, Robot.class);
```

Eine Liste analysieren:

Wenn Sie eine Liste von JSON-Objekten abrufen, möchten Sie sie häufig analysieren und in Java-Objekte konvertieren.

Die JSON-Zeichenfolge, die wir zu konvertieren versuchen, lautet wie folgt:

```
{
  "owned_dogs": [
    {
      "name": "Ron",
      "age": 12,
      "breed": "terrier"
    },
    {
      "name": "Bob",
      "age": 4,
      "breed": "bulldog"
    },
    {
      "name": "Johny",
      "age": 3,
      "breed": "golden retriever"
    }
  ]
}
```

Dieses spezielle JSON-Array enthält drei Objekte. In unserem Java-Code möchten wir diese Objekte `Dog` Objekten `Dog` . Ein Dog-Objekt würde folgendermaßen aussehen:

```
private class Dog {
    public String name;
    public int age;

    @SerializedName("breed")
    public String breedName;
}
```

So konvertieren Sie das JSON-Array in einen `Dog[]` :

```
Dog[] arrayOfDogs = gson.fromJson(jsonArrayString, Dog[].class);
```

Konvertieren eines `Dog[]` in einen JSON-String:

```
String jsonArray = gson.toJson(arrayOfDogs, Dog[].class);
```

Um das JSON-Array in eine `ArrayList<Dog>` zu konvertieren, können Sie Folgendes tun:

```
Type typeListOfDogs = new TypeToken<List<Dog>>(){}.getType();
List<Dog> listOfDogs = gson.fromJson(jsonArrayString, typeListOfDogs);
```

Das Type-Objekt `typeListOfDogs` definiert, wie eine Liste von `Dog` Objekten aussehen würde. GSON kann dieses Typobjekt verwenden, um das JSON-Array den richtigen Werten zuzuordnen.

Alternativ kann das Konvertieren einer `List<Dog>` in ein JSON-Array auf ähnliche Weise erfolgen.

```
String jsonArray = gson.toJson(listOfDogs, typeListOfDogs);
```

Analysieren der JSON-Eigenschaft mit Gson

Wenn Sie einen String für die Enumeration mit Gson analysieren möchten:

{"status": "open"}

```
public enum Status {
    @SerializedName("open")
    OPEN,
    @SerializedName("waiting")
    WAITING,
    @SerializedName("confirm")
    CONFIRM,
    @SerializedName("ready")
    READY
}
```

Eine Liste analysieren mit Gson

Methode 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());
```

Dies ist für die meisten generischen Containerklassen hilfreich, da Sie die Klasse eines parametrisierten Typs nicht abrufen können (z. B. `List<String>.class`).

Methode 2

```
public class StringList extends ArrayList<String> { }

...

List<String> members = gson.fromJson(json, StringList.class);
```

Alternativ können Sie den gewünschten Typ immer einer Unterklasse zuordnen und diese Klasse dann übergeben. Dies ist jedoch nicht immer die bewährte `StringList` , da ein Objekt vom Typ `StringList` an Sie `StringList` .

JSON-Serialisierung / Deserialisierung mit AutoValue und Gson

Importieren Sie in Ihrer Gradle-Root-Datei

```
classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
```

Importieren Sie Ihre Gradle-App-Datei

```
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'
```

Objekt mit automatischem Wert erstellen:

```
@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);
    }
}
```

Erstellen Sie Ihren Gson-Konverter mit Ihrem GsonBuilder

```
Gson gson = new GsonBuilder()
                .registerTypeAdapterFactory(
                    new AutoValueGsonTypeAdapterFactory())
                .create());
```

Deserialisieren

```
String myJsonData = "{
    \"signin_token\": \"mySigninToken\",
    \"username\": \"myUsername\" }";
SignIn signInData = gson.fromJson(myJsonData, Signin.class);
```

Serialisieren

```
Signin myData = SignIn.create("myTokenData", "myUsername");
String myJsonData = gson.toJson(myData);
```

Die Verwendung von Gson ist eine großartige Möglichkeit, den Serialisierungs- und Deserialisierungscode mithilfe von POJO-Objekten zu vereinfachen. Der Nebeneffekt ist, dass die Reflexion kostspielig ist. Durch die Verwendung von AutoValue-Gson zur Generierung von CustomTypeAdapter werden diese Reflexionskosten vermieden und es bleibt sehr einfach zu aktualisieren, wenn eine API-Änderung stattfindet.

Parsen von JSON in generische Klassenobjekte mit Gson

Angenommen, wir haben eine JSON-Zeichenfolge:

```
["first","second","third"]
```

Wir können diesen JSON-String in ein `String` Array parsen:

```
Gson gson = new Gson();
```

```
String jsonArray = "[\"first\",\"second\",\"third\"]";
String[] strings = gson.fromJson(jsonArray, String[].class);
```

Wenn wir es jedoch in ein `List<String>` -Objekt analysieren wollen, müssen wir `TypeToken` .

Hier ist die Probe:

```
Gson gson = new Gson();
String jsonArray = "[\"first\",\"second\",\"third\"]";
List<String> stringList = gson.fromJson(jsonArray, new TypeToken<List<String>>()
{}.getType());
```

Nehmen wir an, wir haben zwei Klassen:

```
public class Outer<T> {
    public int index;
    public T data;
}

public class Person {
    public String firstName;
    public String lastName;
}
```

und wir haben eine JSON-Zeichenfolge, die in ein `Outer<Person>` -Objekt analysiert werden sollte.

In diesem Beispiel wird veranschaulicht, wie diese JSON-Zeichenfolge in das verwandte generische Klassenobjekt analysiert wird:

```
String json = "......";
Type userType = new TypeToken<Outer<Person>>(){}.getType();
Result<User> userResult = gson.fromJson(json,userType);
```

Wenn die JSON-Zeichenfolge in ein `Outer<List<Person>>` -Objekt analysiert werden soll:

```
Type userListType = new TypeToken<Outer<List<Person>>>(){}.getType();
Result<List<User>> userListResult = gson.fromJson(json,userListType);
```

Hinzufügen von Gson zu Ihrem Projekt

```
dependencies {
    compile 'com.google.code.gson:gson:2.8.1'
}
```

Um die neueste Version von Gson zu verwenden

Die folgende Zeile wird die neueste Version der Gson-Bibliothek bei jedem Kompilieren kompilieren.

Vorteile: Sie können die neuesten Funktionen, die Geschwindigkeit und weniger Fehler verwenden.

Nachteile: Es könnte die Kompatibilität mit Ihrem Code beeinträchtigen.

```
compile 'com.google.code.gson:gson:+'
```

Gson verwenden, um eine JSON-Datei von der Festplatte zu laden.

Dadurch wird eine JSON-Datei von der Festplatte geladen und in den angegebenen Typ konvertiert.

```
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);
}
```

Hinzufügen eines benutzerdefinierten Konverters zu Gson

Manchmal müssen Sie einige Felder in einem gewünschten Format serialisieren oder deserialisieren. Beispielsweise verwendet Ihr Backend das Format "JJJJ-MM-tt HH: mm" für Datumsangaben und Sie möchten, dass Ihr POJOS die DateTime-Klasse in Joda Time verwendet.

Um diese Zeichenfolgen automatisch in ein DateTimes-Objekt zu konvertieren, können Sie einen benutzerdefinierten Konverter verwenden.

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

Damit Gson den neu erstellten Konverter verwenden kann, müssen Sie ihn beim Erstellen des Gson-Objekts zuweisen:

```
    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
```

Um das Datum in diesem Format zu deserialisieren, müssen Sie lediglich ein Feld im DateTime-Format definieren:

```
public class SomePojo {
    private DateTime someDate;
}
```

Wenn Gson auf ein Feld des Typs DateTime trifft, ruft es Ihren Konverter auf, um das Feld zu deserialisieren.

Verwendung von Gson als Serializer mit Retrofit

Zunächst müssen Sie die `GsonConverterFactory` zu Ihrer build.gradle-Datei hinzufügen

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

Dann müssen Sie die Konverter-Factory hinzufügen, wenn Sie den Retrofit-Service erstellen:

```
Gson gson = new GsonBuilder().create();
new Retrofit.Builder()
        .baseUrl(someUrl)
        .addConverterFactory(GsonConverterFactory.create(gson))
        .build()
        .create(RetrofitService.class);
```

Sie können benutzerdefinierte Konverter hinzufügen, wenn Sie das Gson-Objekt erstellen, das Sie an die Factory übergeben. So können Sie benutzerdefinierte Typkonvertierungen erstellen.

Json-Array wird mit Gson in eine generische Klasse geparst

Angenommen, wir haben einen Json

```
{
  "total_count": 132,
  "page_size": 2,
  "page_index": 1,
  "twitter_posts": [
```

```
    {
      "created_on": 1465935152,
      "tweet_id": 210462857140252672,
      "tweet": "Along with our new #Twitterbird, we've also updated our Display Guidelines",
      "url": "https://twitter.com/twitterapi/status/210462857140252672"
    },
    {
      "created_on": 1465995741,
      "tweet_id": 735128881808691200,
      "tweet": "Information on the upcoming changes to Tweets is now on the developer site",
      "url": "https://twitter.com/twitterapi/status/735128881808691200"
    }
  ]
}
```

Wir können dieses Array manuell in ein Objekt für benutzerdefinierte Tweets (Tweets- `fromJson`) parsen, es ist jedoch einfacher, die `fromJson` Methode zu verwenden:

```
Gson gson = new Gson();
String jsonArray = "....";
Tweets tweets = gson.fromJson(jsonArray, Tweets.class);
```

Nehmen wir an, wir haben zwei Klassen:

```
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;
}
```

und wenn Sie nur ein Json-Array analysieren müssen, können Sie diesen Code bei der Analyse verwenden:

```
String tweetsJsonArray = "[{.....},{.....}]"
List<Tweet> tweets = gson.fromJson(tweetsJsonArray, new TypeToken<List<Tweet>>()
{}.getType());
```

Benutzerdefinierter JSON-Deserializer mit Gson

Stellen Sie sich vor, Sie haben alle Datumsangaben in allen Antworten in einem benutzerdefinierten Format, z. B. `/Date(1465935152)/` und Sie möchten die Regel anwenden, um alle Json-Datumsangaben zu java `Date` Instanzen zu deserialisieren. In diesem Fall müssen Sie einen benutzerdefinierten Json Deserializer implementieren.

Beispiel für Json:

```
{
  "id": 1,
  "created_on": "Date(1465935152)",
  "updated_on": "Date(1465968945)",
  "name": "Oleksandr"
}
```

Angenommen, wir haben diese Klasse unten:

```
class User {
    @SerializedName("id")
    long id;
    @SerializedName("created_on")
    Date createdOn;
    @SerializedName("updated_on")
    Date updatedOn;
    @SerializedName("name")
    String name;
}
```

Kundenspezifischer Deserialisierer:

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

Und die Verwendung:

```
Gson gson = new GsonBuilder()
                .registerTypeAdapter(Date.class, new DateDeSerializer())
                .create();
String json = "....";
User user = gson.fromJson(json, User.class);
```

Serialisieren und deserialisieren Sie Jackson JSON-Zeichenfolgen mit Date-Typen

Dies gilt beispielsweise auch für den Fall, dass Sie die Konvertierung von Gson Date mit Jackson kompatibel machen möchten.

Jackson serialisiert Date normalerweise in "Millisekunden seit der Epoche", während Gson ein lesbares Format wie den `Aug 31, 2016 10:26:17` , um Date darzustellen. Dies führt zu JsonSyntaxExceptions in Gson, wenn Sie versuchen, ein Datum im Jackson-Format zu deserialisieren.

Um dies zu umgehen, können Sie einen benutzerdefinierten Serializer und einen benutzerdefinierten Deserializer hinzufügen:

```
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();
```

Verwendung von Gson mit Vererbung

Gson unterstützt nicht die sofortige Vererbung.

Nehmen wir an, wir haben die folgende Klassenhierarchie:

```
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;
    }
}
```

Und jetzt möchten wir eine Instanz von `DerivedClass1` in einen JSON-String serialisieren

```
DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;

Gson gson = new Gson();
String derivedClass1Json = gson.toJson(derivedClass1);
```

Nun, an einem anderen Ort, erhalten wir diesen Json-String und möchten ihn deserialisieren - aber zur Kompilierzeit wissen wir nur, dass es sich um eine Instanz von `BaseClass` :

```
BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());
```

Aber GSON weiß nicht, dass `derivedClass1Json` ursprünglich eine Instanz von `DerivedClass1` , daher werden 10 `DerivedClass1` .

Wie löse ich das?

Sie müssen einen eigenen `JsonDeserializer` , der solche Fälle behandelt. Die Lösung ist nicht perfekt sauber, aber ich könnte mir keine bessere vorstellen.

Fügen Sie zunächst der Basisklasse das folgende Feld hinzu

```
@SerializedName("type")
private String typeName;
```

Und initialisieren Sie es im Basisklassenkonstruktor

```
public BaseClass() {
    typeName = getClass().getName();
}
```

Fügen Sie nun folgende Klasse hinzu:

```
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);
 }
}
```

Jetzt bleibt nur noch alles anzuschließen -

```
GsonBuilder builder = new GsonBuilder();
 builder
 .registerTypeAdapter(BaseClass.class, new JsonDeserializerWithInheritance<BaseClass>());
 Gson gson = builder.create();
```

Und jetzt den folgenden Code ausführen:

```
 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());
```

Wird ausgedruckt 5.

Gson online lesen: https://riptutorial.com/de/android/topic/4158/gson

Kapitel 120: Handler

Bemerkungen

Ein Handler kann leicht verwendet werden, um Code nach einer verzögerten Zeit auszuführen. Es ist auch nützlich, um Code nach einer bestimmten Zeit wiederholt auszuführen, indem die Handler.postDelayed () - Methode erneut aus der run () - Methode von Runnable aufgerufen wird.

Examples

Verwenden eines Handlers, um Code nach einer verzögerten Zeit auszuführen

Code nach 1,5 Sekunden ausführen:

```
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
```

Code wird alle 1 Sekunde wiederholt ausgeführt:

```
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 und Kommunikation zwischen Threads

Als `Handler` s verwendet werden , senden `Message` s und `Runnable` s auf das Thema der Nachricht Warteschlange ist es einfach , ereignisbasierte Kommunikation zwischen mehreren Threads zu implementieren. Jeder Thread, der über einen `Looper` verfügt, kann Nachrichten empfangen und verarbeiten. Ein `HandlerThread` ist ein Thread, der einen solchen `Looper` implementiert. Beispielsweise implementiert der Haupt-Thread (UI-Thread) die Funktionen eines `HandlerThread` .

Einen Handler für den aktuellen Thread erstellen

```
Handler handler = new Handler();
```

Handler für den Haupt-Thread erstellen (UI-Thread)

```
Handler handler = new Handler(Looper.getMainLooper());
```

Senden Sie eine ausführbare Datei aus einem anderen Thread an den Haupt-Thread

```
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();
```

Erstellen eines Handlers für ein anderes HandlerThread und Senden von Ereignissen an ihn

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

Stoppen Sie den Handler von der Ausführung ab

Um die Ausführung des Handlers zu stoppen, entfernen Sie den an ihn gebundenen Rückruf mithilfe der ausführbaren ausführbaren Instanz.

```
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);
}
```

Verwenden Sie den Handler, um einen Timer zu erstellen (ähnlich javax.swing.Timer).

Dies kann nützlich sein, wenn Sie ein Spiel schreiben oder etwas, das alle paar Sekunden einen Code ausführen muss.

```
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 ();
    }
}
```

Verwendungsbeispiel:

```
Timer timer = new Timer(new Runnable() {
    public void run() {
        System.out.println("Hello");
    }
}, 1000, true)
```

Dieser Code gibt jede Sekunde "Hallo" aus.

Handler online lesen: https://riptutorial.com/de/android/topic/1425/handler

Kapitel 121: Hardwaretastenereignisse / Absichten (PTT, LWP usw.)

Einführung

Mehrere Android-Geräte verfügen über benutzerdefinierte Schaltflächen, die vom Hersteller hinzugefügt wurden. Dies eröffnet dem Entwickler neue Möglichkeiten beim Umgang mit diesen Schaltflächen, insbesondere bei der Erstellung von Apps für Hardware-Geräte.

In diesem Thema werden Schaltflächen beschrieben, die mit Absichten versehen sind, auf die Sie über Absichtsempfänger hören können.

Examples

Sonim-Geräte

Sonim-Geräte haben je nach Modell viele verschiedene benutzerdefinierte Schaltflächen:

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
```

Registrieren der Tasten

Um diese Absichten zu erhalten, müssen Sie Ihre Tasten in den Telefoneinstellungen Ihrer App zuweisen. Sonim hat die Möglichkeit, die Schaltflächen bei der Installation automatisch für die App zu registrieren. Dazu müssen Sie sich an sie wenden und einen paketspezifischen Schlüssel erhalten, der in Ihr Manifest aufgenommen werden soll:

```
<meta-data
    android:name="app_key_green_data"
    android:value="your-key-here" />
```

RugGear-Geräte

PTT-Taste

```
android.intent.action.PTT.down
android.intent.action.PTT.up
```

Bestätigt am: RG730, RG740A

Hardwaretastenereignisse / Absichten (PTT, LWP usw.) online lesen:
https://riptutorial.com/de/android/topic/10418/hardwaretastenereignisse---absichten--ptt--lwp-usw--

Kapitel 122: HTTP-Verbindung

Syntax

- abstrakter Verbindungsabbruch ()
- abstraktes boolesches usingProxy ()
- statisch boolean getFollowRedirects ()
- statisch void setFollowRedirects (boolescher Satz)
- String getHeaderField (int n)
- String getHeaderFieldKey (int n)
- String getRequestMethod ()
- String getResponseMessage ()
- int getResponseCode ()
- long getHeaderFieldDate (Stringname, long Standard)
- boolean getInstanceFollowRedirects ()
- Berechtigung getPermission ()
- InputStream getErrorStream ()
- void setChunkedStreamingMode (int chunklen)
- void setFixedLengthStreamingMode (int contentLength)
- void setFixedLengthStreamingMode (long contentLength)
- void setInstanceFollowRedirects (boolean followRedirects)
- void setRequestMethod (String-Methode)

Bemerkungen

`HttpURLConnection` ist der Standard-HTTP-Client für Android, der zum Senden und Empfangen von Daten über das Web verwendet wird. Es ist eine konkrete Implementierung von URLConnection für HTTP (RFC 2616).

Examples

Erstellen einer HttpURLConnection

Um einen neuen Android-HTTP-Client `HttpURLConnection` zu erstellen, rufen Sie `openConnection()` für eine URL-Instanz auf. Da `openConnection()` eine `URLConnection` , müssen Sie den zurückgegebenen Wert explizit `URLConnection` .

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
// do something with the connection
```

Wenn Sie eine neue `URL` erstellen, müssen Sie auch die Ausnahmen behandeln, die der URL-Analyse zugeordnet sind.

```
try {
    URL url = new URL("http://example.com");
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    // do something with the connection
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

Nachdem der Antworttext gelesen wurde und die Verbindung nicht mehr erforderlich ist, sollte die Verbindung durch Aufrufen von `disconnect()` geschlossen werden.

Hier ist ein Beispiel:

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
try {
    // do something with the connection
} finally {
    connection.disconnect();
}
```

Senden einer HTTP-GET-Anforderung

```
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();
}
```

Bitte beachten Sie, dass Ausnahmen im obigen Beispiel nicht behandelt werden. Ein vollständiges Beispiel, einschließlich (einer trivialen) Ausnahmebehandlung, wäre:

```
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();
    }
}
```

Lesen des Körpers einer HTTP-GET-Anforderung

```
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();
}
```

Bitte beachten Sie, dass Ausnahmen im obigen Beispiel nicht behandelt werden.

Verwenden Sie HttpURLConnection für mehrteilige / Formulardaten

Erstellen Sie eine benutzerdefinierte Klasse für den Aufruf einer HttpURLConnection-Anforderung für mehrere Teile / Formulardaten

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;
        }
    }
```

Verwenden Sie es (asynchrone Art und Weise)

```
    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;
                }
```

Senden einer HTTP-POST-Anforderung mit Parametern

Verwenden Sie eine HashMap, um die Parameter zu speichern, die über POST-Parameter an den Server gesendet werden sollen:

```
HashMap<String, String> params;
```

Wenn die `params` HashMap `params` ist, erstellen Sie den StringBuilder, mit dem sie an den Server gesendet werden:

```
    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++;
    }
```

Erstellen Sie dann die HttpURLConnection, öffnen Sie die Verbindung und senden Sie die POST-Parameter:

```
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();
}
```

Dann erhalten Sie das Ergebnis, das der Server zurücksendet:

```
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();
    }
}
```

Upload (POST) -Datei mit HttpURLConnection

Häufig ist es notwendig, eine Datei an einen Remote-Server zu senden / hochzuladen, beispielsweise ein Image, Video, Audio oder ein Backup der Anwendungsdatenbank auf einen Remote-Private-Server. Angenommen, der Server erwartet eine POST-Anforderung mit dem Inhalt. Hier ein einfaches Beispiel, wie diese Aufgabe in Android ausgeführt wird.

Datei-Uploads werden unter Verwendung von POST-Anforderungen für `multipart/form-data` gesendet. Es ist sehr einfach zu implementieren:

```
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
}
```

Natürlich müssen Ausnahmen gefangen oder als geworfen deklariert werden. Ein paar Punkte zu diesem Code zu beachten:

1. `postTarget` ist die Ziel-URL des POST. `oauthToken` ist das Authentifizierungstoken. `fileDescription` ist die Beschreibung der Datei, die als der Wert des Feldes gesendet `description` ; `file` ist die zu sendende Datei - es ist vom Typ `java.io.File` Wenn Sie über den Dateipfad verfügen, können Sie stattdessen die `new File(filePath)` verwenden.
2. Es setzt den `Authorization` für eine oAuth-Authentifizierung
3. Es verwendet Apache Common `FileUtil` , um die Datei in ein Byte-Array zu lesen. Wenn Sie den Inhalt der Datei bereits in einem Byte-Array oder auf andere Weise im Speicher haben, müssen Sie ihn nicht lesen.

Eine vielseitige HttpURLConnection-Klasse für die Verarbeitung aller Arten von HTTP-Anforderungen

Die folgende Klasse kann als eine einzelne Klasse verwendet werden , die verarbeiten können `GET` , `POST` , `PUT` , `PATCH` und andere Anfragen:

```
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();
    }
}
```

Verwendungszweck

Verwenden Sie einen der angegebenen Konstruktoren der Klasse, je nachdem, ob Sie `POST` Daten oder zusätzliche Header senden müssen.

Die `onComplete()` -Methode wird aufgerufen, wenn der Datenabruf abgeschlossen ist. Die Daten werden als Objekt der `APIResponseObject` Klasse zurückgegeben, deren Statuscode den HTTP-Statuscode der Anforderung und eine Zeichenfolge mit der Antwort enthält. Sie können diese Antwort in Ihrer Klasse analysieren, z. B. XML oder JSON.

Rufen Sie `execute()` für das Objekt der Klasse auf, um die Anforderung auszuführen, wie im folgenden Beispiel gezeigt:

```
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();
}
```

HTTP-Verbindung online lesen: https://riptutorial.com/de/android/topic/781/http-verbindung

Kapitel 123: Im Play Store veröffentlichen

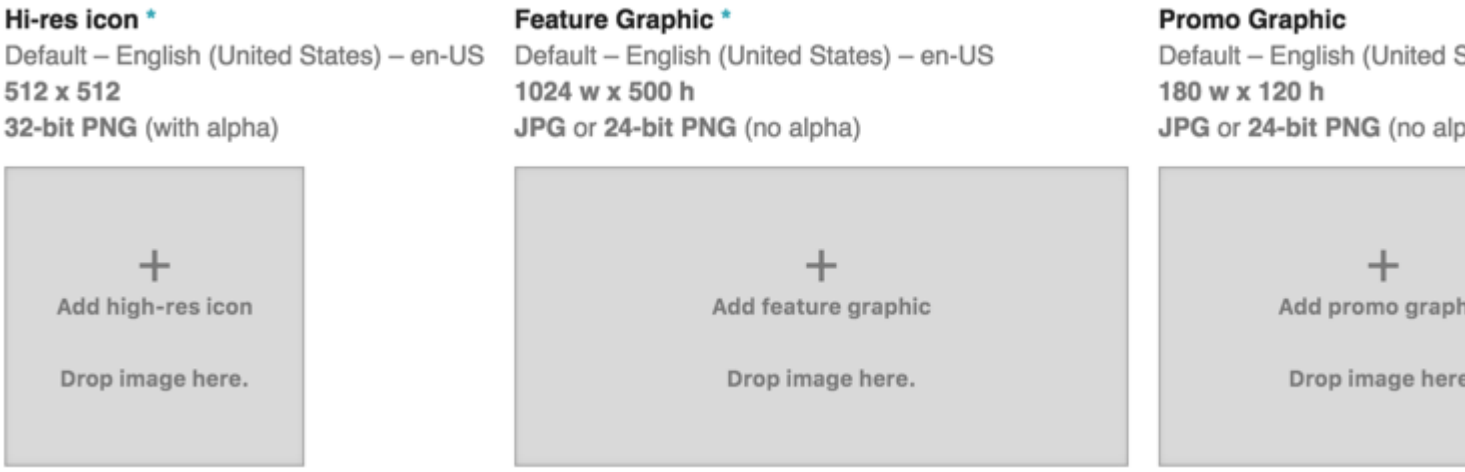
Examples

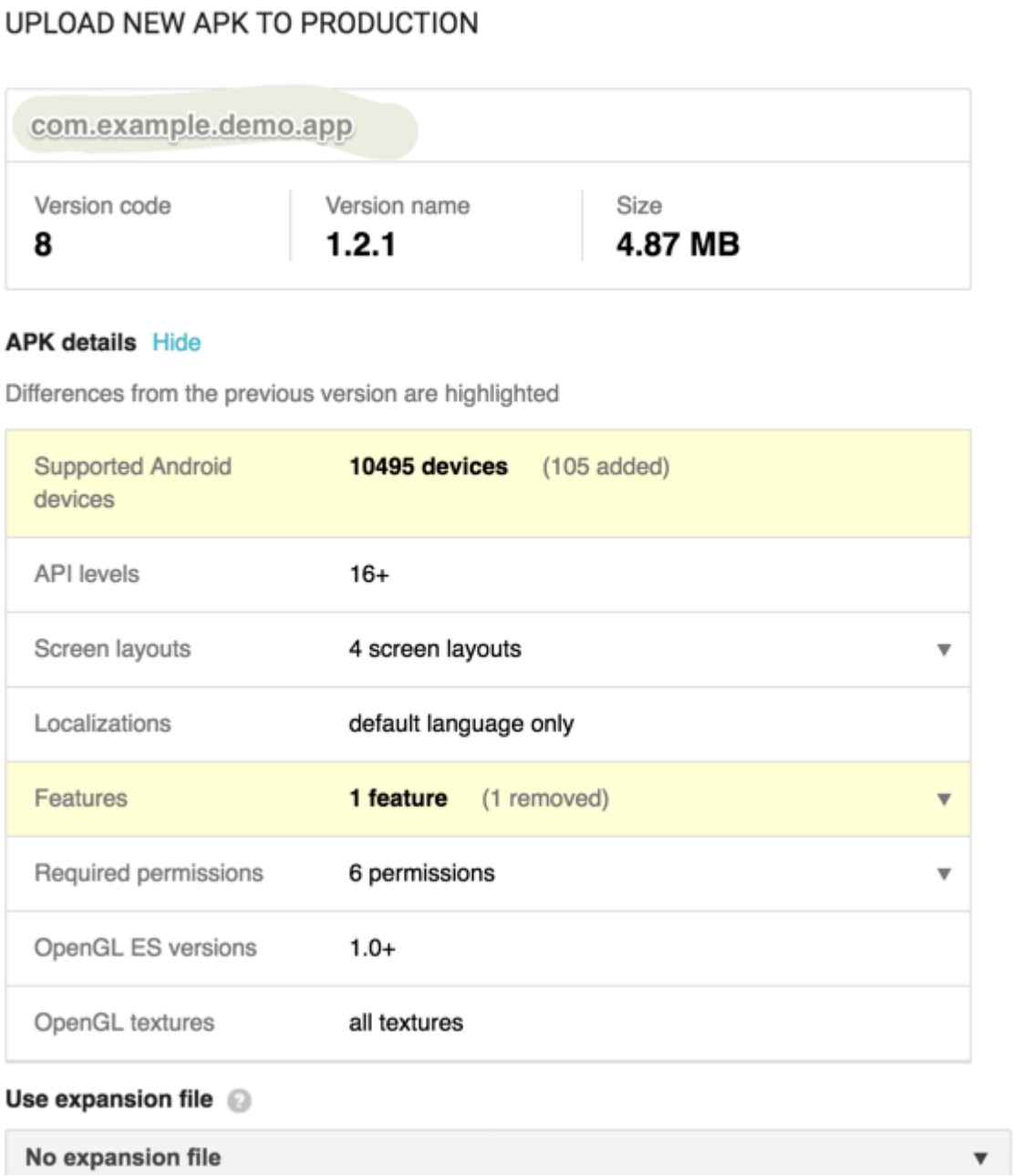
Minimaler App-Einreichungsleitfaden

Bedarf:

- Ein Entwicklerkonto
- Eine apk ist bereits erstellt und mit einem Nicht-Debug-Schlüssel signiert
- Eine kostenlose App, die keine In-App-Abrechnung hat
- keine Firebase Cloud Messaging oder Game Services

1. Gehen Sie zu https://play.google.com/apps/publish/.
   1a) Erstellen Sie Ihr Entwicklerkonto, falls Sie noch keinen haben
2. Klicken Sie auf die Schaltfläche `Create new Application`
3. Klicken Sie auf APK absenden
4. Füllen Sie alle erforderlichen Felder im Formular aus, einschließlich einiger Assets, die im Play Store angezeigt werden (siehe Abbildung unten).
5. Wenn Sie zufrieden sind, `Publish app` Schaltfläche " `Publish app`





Weitere Informationen zum Signieren finden Sie unter Signiereinstellungen konfigurieren

Im Play Store veröffentlichen online lesen: https://riptutorial.com/de/android/topic/5369/im-play-store-veroffentlichen

Kapitel 124: Imbissbude

Syntax

- Snackbar machen (View-Ansicht, CharSequence-Text, int duration)
- Snackbar machen (View view, int resId, int duration)

Parameter

Parameter	Beschreibung
Aussicht	Ansicht: Die Ansicht, aus der ein übergeordnetes Element gesucht wird.
Text	CharSequence: Der anzuzeigende Text. Kann formatierter Text sein.
resId	int: Die Ressourcen-ID der zu verwendenden String-Ressource. Kann formatierter Text sein.
Dauer	int: Wie lange wird die Nachricht angezeigt? Dies können LENGTH_SHORT, LENGTH_LONG oder LENGTH_INDEFINITE sein

Bemerkungen

Die Snackbar bietet leichtes Feedback zu einer Operation. Bei Mobilgeräten erscheint unten auf dem Bildschirm eine kurze Meldung und bei größeren Geräten unten links. Snackbars erscheinen über allen anderen Elementen auf dem Bildschirm und es kann jeweils nur eine angezeigt werden.

Sie verschwinden automatisch nach einem Timeout oder nach Benutzerinteraktion an anderen Stellen auf dem Bildschirm, insbesondere nach Interaktionen, die eine neue Oberfläche oder Aktivität aufrufen. Snackbar kann vom Bildschirm gezogen werden.

Bevor Sie `SnackBar` , müssen Sie die Abhängigkeit der Design-Unterstützungsbibliothek in der Datei `build.gradle` :

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

Offizielle Dokumentation

https://developer.android.com/reference/android/support/design/widget/Snackbar.html

Examples

Eine einfache Snackbar erstellen

Das Erstellen einer `Snackbar` kann wie folgt durchgeführt werden:

```
Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG).show();
```

Die `view` wird verwendet, um ein geeignetes übergeordnetes `Snackbar` zum Anzeigen der `Snackbar` . In der Regel handelt es sich hierbei um ein `CoordinatorLayout` , das Sie in Ihrem XML-Objekt definiert haben. Dadurch können Sie Funktionen hinzufügen, beispielsweise Streichen, um andere Widgets (z. B. `FloatingActionButton`) zu schließen und automatisch zu verschieben. Wenn kein `CoordinatorLayout` ist, wird die Inhaltsansicht des Fensterdekors verwendet.

Sehr oft fügen wir der `Snackbar` auch eine Aktion hinzu. Ein häufiger Anwendungsfall wäre eine Aktion "Rückgängig machen".

```
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();
```

Sie können eine `Snackbar` erstellen und später `Snackbar` :

```
Snackbar snackbar = Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG);
snackbar.show();
```

Wenn Sie die Farbe des `Snackbar` -Texts ändern möchten:

```
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();
```

Standardmäßig verwirft `Snackbar` den rechten Swipe . Dieses Beispiel zeigt, wie die snackBar auf dem linken Swipe verworfen wird .

Benutzerdefinierte Snackbar

Funktion zum Anpassen der Snackbar

```
public static Snackbar makeText(Context context, String message, int duration) {
        Activity activity = (Activity) context;
        View layout;
        Snackbar snackbar = Snackbar
                .make(activity.findViewById(android.R.id.content), message, duration);
        layout = snackbar.getView();
```

```
        //setting background color
        layout.setBackgroundColor(context.getResources().getColor(R.color.orange));
        android.widget.TextView text = (android.widget.TextView)
layout.findViewById(android.support.design.R.id.snackbar_text);
        //setting font color
        text.setTextColor(context.getResources().getColor(R.color.white));
        Typeface font = null;
        //Setting font
        font = Typeface.createFromAsset(context.getAssets(), "DroidSansFallbackanmol256.ttf");
        text.setTypeface(font);
        return snackbar;

    }
```

Rufen Sie die Funktion aus Fragment oder Aktivität auf

```
 SnackBar.makeText(MyActivity.this, "Please Locate your address at Map",
Snackbar.LENGTH_SHORT).show();
```

Snackbar mit Rückruf

Sie können Snackbar verwenden. Rückruf, um zu hören, ob die Snackbar vom Benutzer oder vom Timeout abgelehnt wurde.

```
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();
```

Benutzerdefinierte Snackbar

Dieses Beispiel zeigt eine weiße Snackbar mit einem benutzerdefinierten Rückgängig-Symbol.

```
Snackbar customBar = Snackbar.make(view , "Text to be displayed", Snackbar.LENGTH_LONG);
customBar.setAction("UNDO", new View.OnClickListener() {
    @Override
    public void onClick(View view) {
    //Put the logic for undo button here

    }
});

View sbView = customBar.getView();
//Changing background to White
sbView.setBackgroundColor(Color.WHITE));

TextView snackText = (TextView)
sbView.findViewById(android.support.design.R.id.snackbar_text);
if (snackText!=null) {
    //Changing text color to Black
   snackText.setTextColor(Color.BLACK);
}

TextView actionText = (TextView)
sbView.findViewById(android.support.design.R.id.snackbar_action);
if (actionText!=null) {
    // Setting custom Undo icon
    actionText.setCompoundDrawablesRelativeWithIntrinsicBounds(R.drawable.custom_undo, 0, 0,
0);
}
customBar.show();
```

Snackbar vs Toasts: Welches sollte ich verwenden?

Toasts werden im Allgemeinen verwendet, wenn wir dem Benutzer Informationen zu einer erfolgreich (oder nicht erfolgreich) erfolgten Aktion anzeigen möchten. Bei dieser Aktion muss der Benutzer keine weiteren Aktionen ausführen. Zum Beispiel, wenn eine Nachricht gesendet wurde:

```
Toast.makeText(this, "Message Sent!", Toast.LENGTH_SHORT).show();
```

Snackbars dienen auch zur Anzeige von Informationen. Diesmal können wir dem Benutzer jedoch die Möglichkeit geben, Maßnahmen zu ergreifen. Angenommen, der Benutzer hat ein Bild aus Versehen gelöscht und möchte es zurückbekommen. Wir können eine Snackbar mit der Aktion "Rückgängig machen" bereitstellen. So was:

```
Snackbar.make(getCurrentFocus(), "Picture Deleted", Snackbar.LENGTH_SHORT)
        .setAction("Undo", new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Return his picture
            }
        })
        .show();
```

Fazit: Toasts werden verwendet, wenn keine Benutzerinteraktion erforderlich ist. Snackbars ermöglichen Benutzern, eine andere Aktion auszuführen oder eine vorherige Aktion rückgängig zu machen.

Benutzerdefinierte Snackbar (keine Notwendigkeit)

Erstellen einer Snackbar, ohne dass die Ansicht auf Snackbar geändert werden muss. Alle Layouts werden in android in Android.R.id.content erstellt.

```
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();
    }
}
```

für Anrufklasse

new CustomSnackBar (findViewById (CustomSnackBar.VIEW_PARENT), "message", CustomSnackBar.STATE_ERROR);

Imbissbude online lesen: https://riptutorial.com/de/android/topic/1500/imbissbude

Kapitel 125: Implizite Absichten

Syntax

- Absicht()
- Absicht (Absicht o)
- Absicht (String-Aktion)
- Intent (String-Aktion, Uri Uri)
- Intent (Context packageContext, Klasse <?> Cls)
- Intent (String-Aktion, Uri-URI, Kontext-PackageContext, Klasse <?> Cls)

Parameter

Parameter	Einzelheiten
O	Absicht
Aktion	`String`: Die Absichtsaktion, z. B. `ACTION_VIEW`.
uri	`Uri`: Der Intent-Daten-URI.
packageContext	`Context`: Ein Kontext des Anwendungspakets, das diese Klasse implementiert.
cls	`Class`: Die Komponentenklasse, die für die Absicht verwendet werden soll.

Bemerkungen

- Mehr über Absicht
- Mehr über Absichtstypen

Examples

Implizite und explizite Absichten

Eine explizite Absicht wird zum Starten einer Aktivität oder eines Services innerhalb desselben Anwendungspakets verwendet. In diesem Fall wird der Name der beabsichtigten Klasse explizit erwähnt:

```
Intent intent = new Intent(this, MyComponent.class);
startActivity(intent);
```

Es wird jedoch eine implizite Absicht für jede auf dem Gerät des Benutzers installierte Anwendung

über das System gesendet, die diese Absicht verarbeiten kann. Dies wird verwendet, um Informationen zwischen verschiedenen Anwendungen auszutauschen.

```
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
}
```

Weitere Einzelheiten zu den Unterschieden finden Sie in den Android-Entwicklerdokumenten hier: Intent Resolution

Implizite Absichten

Implizite Absichten benennen keine bestimmte Komponente, sondern deklarieren stattdessen eine allgemeine Aktion, die ausgeführt werden soll, sodass eine Komponente aus einer anderen App damit umgehen kann.

Wenn Sie dem Benutzer beispielsweise einen Ort auf einer Karte anzeigen möchten, können Sie eine implizite Absicht verwenden, um anzufordern, dass eine andere fähige App einen bestimmten Ort auf einer Karte anzeigt.

Beispiel:

```
// 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);
}
```

Implizite Absichten online lesen: https://riptutorial.com/de/android/topic/5336/implizite-absichten

Kapitel 126: In-App-Abrechnung

Examples

Verbrauchbare In-App-Käufe

Verbrauchsfähige verwaltete Produkte sind Produkte, die mehrmals gekauft werden können, wie beispielsweise Spielwährung, Spieldauer, Power-Ups usw.

In diesem Beispiel werden 4 verschiedene **verwaltete** Verbrauchsmaterialien `"item1", "item2", "item3", "item4"` .

Schritte in Zusammenfassung:

1. Fügen Sie Ihrem Projekt die In-App-Abrechnungsbibliothek hinzu (AIDL-Datei).
2. Fügen Sie die erforderliche Berechtigung in der Datei `AndroidManifest.xml` .
3. Stellen Sie eine signierte apk für Google Developers Console bereit.
4. Definieren Sie Ihre Produkte.
5. Implementieren Sie den Code.
6. Testen Sie die In-App-Abrechnung (optional).

Schritt 1:

Zuerst müssen wir die AIDL-Datei zu Ihrem Projekt hinzufügen, wie in der Google-Dokumentation hier klar beschrieben.

`IInAppBillingService.aidl` ist eine AIDL-Datei (Android Interface Definition Language), in der die Schnittstelle zum In-App-Abrechnungsdienst der Version 3 definiert wird. Sie verwenden diese Schnittstelle, um Abrechnungsanforderungen durch Aufrufen von IPC-Methodenaufrufen auszuführen.

Schritt 2:

Fügen Sie nach dem Hinzufügen der AIDL-Datei die BILLING-Berechtigung in `AndroidManifest.xml` :

```
<!-- Required permission for implementing In-app Billing -->
<uses-permission android:name="com.android.vending.BILLING" />
```

Schritt 3:

Generieren Sie eine signierte apk und laden Sie sie in die Google Developers Console hoch. Dies ist erforderlich, damit wir unsere In-App-Produkte dort definieren können.

Schritt 4:

Definieren Sie alle Ihre Produkte mit einer anderen productID und legen Sie für jedes Produkt einen Preis fest. Es gibt zwei Arten von Produkten (verwaltete Produkte und Abonnements). Wie bereits erwähnt, werden wir 4 verschiedene **verwaltete** Verbrauchsmaterialien `"item1", "item2", "item3", "item4"`.

Schritt 5:

Nachdem Sie alle oben genannten Schritte ausgeführt haben, können Sie nun den Code selbst in Ihre eigene Aktivität implementieren.

Hauptaktivität:

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

Schritt 6:

Nachdem Sie den Code implementiert haben, können Sie ihn testen, indem Sie Ihre APK im Beta- / Alphakanal bereitstellen und andere Benutzer den Code für Sie testen lassen. Im Testmodus können jedoch keine echten In-App-Käufe getätigt werden. Sie müssen Ihre App / Ihr Spiel zunächst im Play Store veröffentlichen, damit alle Produkte vollständig aktiviert sind.

Weitere Informationen zum Testen der In-App-Abrechnung finden Sie hier .

In-App v3-Bibliothek von Drittanbietern

Schritt 1: Führen Sie zunächst die folgenden zwei Schritte aus, um die App-Funktionalität hinzuzufügen:

1. Fügen Sie die Bibliothek mit folgendem Befehl hinzu:

```
repositories {
           mavenCentral()
       }
       dependencies {
          compile 'com.anjlab.android.iab.v3:library:1.0.+'
       }
```

2. Fügen Sie die Berechtigung in der Manifestdatei hinzu.

```
<uses-permission android:name="com.android.vending.BILLING" />
```

Schritt 2: Initialisieren Sie Ihren Rechnungsprozessor:

```
BillingProcessor bp = new BillingProcessor(this, "YOUR LICENSE KEY FROM GOOGLE PLAY CONSOLE
HERE", this);
```

und implementieren Sie Billing Handler: BillingProcessor.IBillingHandler, der 4 Methoden enthält: a. onBillingInitialized (); b. onProductPurchased (String productId, TransactionDetails-Details): Hier müssen Sie Aktionen ausführen, die nach einem erfolgreichen Kauf ausgeführt werden sollen. c. onBillingError (int errorCode, Throwable-Fehler): Behandlung von Fehlern, die während des Kaufvorgangs aufgetreten sind. d. onPurchaseHistoryRestored (): Zum Wiederherstellen in App-Käufen

Schritt 3: Wie kaufe ich ein Produkt?

So kaufen Sie ein verwaltetes Produkt:

```
bp.purchase(YOUR_ACTIVITY, "YOUR PRODUCT ID FROM GOOGLE PLAY CONSOLE HERE");
```

Und um ein Abonnement zu erwerben:

```
bp.subscribe(YOUR_ACTIVITY, "YOUR SUBSCRIPTION ID FROM GOOGLE PLAY CONSOLE HERE");
```

Schritt 4: Ein Produkt konsumieren.

Um ein Produkt zu konsumieren, rufen Sie die consumePurchase-Methode auf.

bp.consumePurchase ("IHRE PRODUKT-ID VON GOOGLE PLAY CONSOLE HERE");

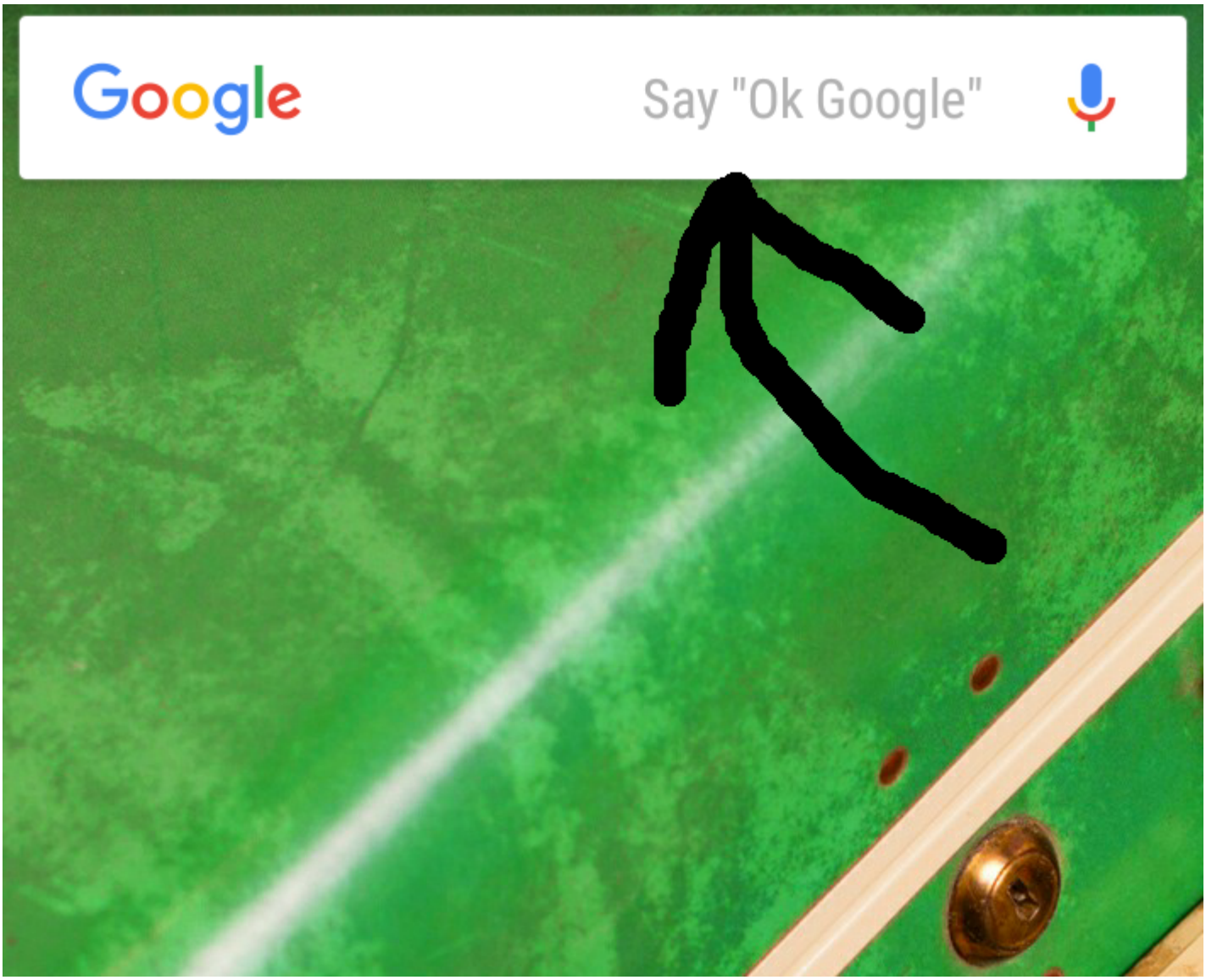
Für andere Methoden in der App besuchen Sie github

In-App-Abrechnung online lesen: https://riptutorial.com/de/android/topic/2843/in-app-abrechnung

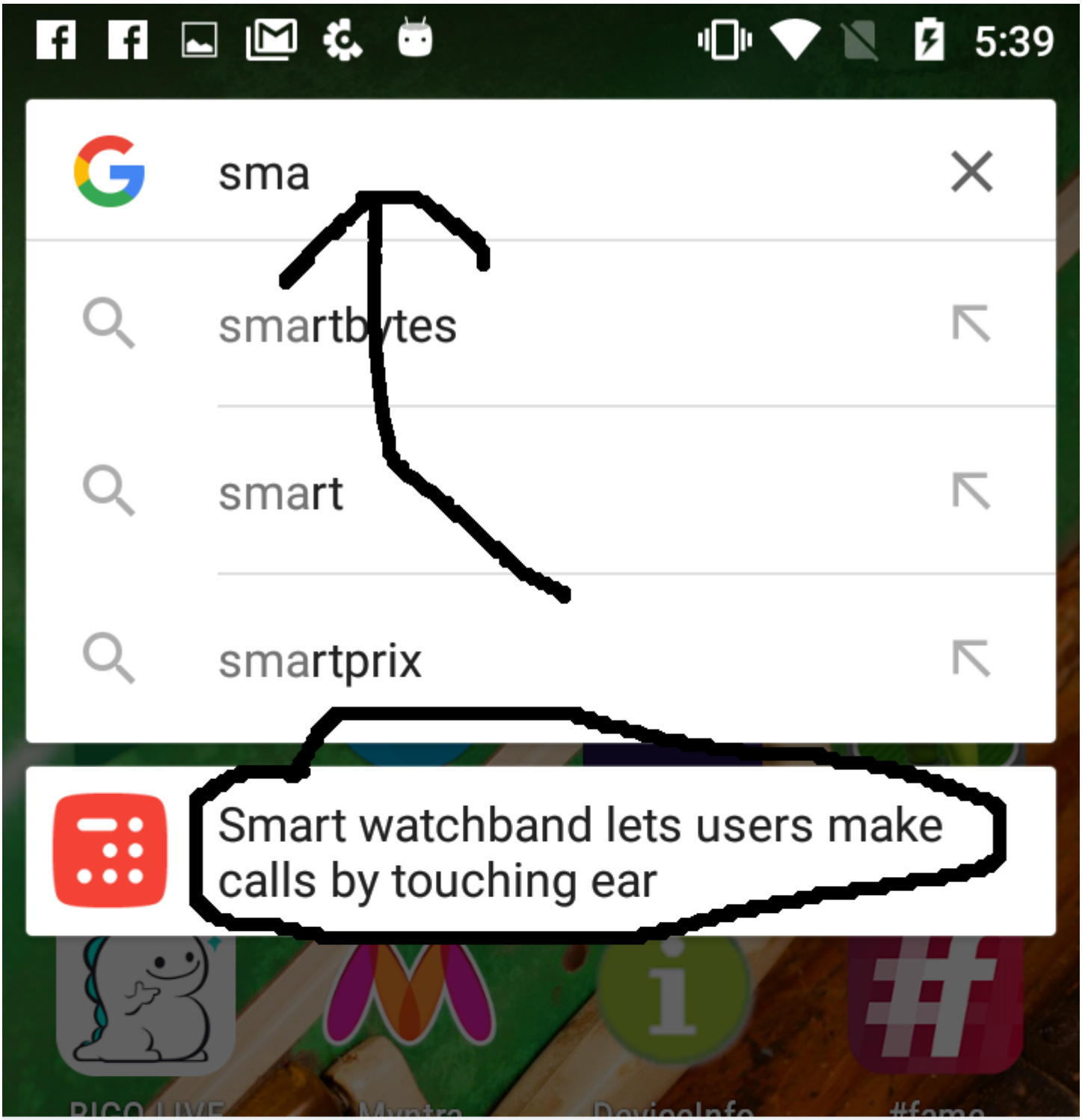
Kapitel 127: Indizierung der Firebase-App

Bemerkungen

- Wenn Sie sich für die App-Indexierung entscheiden, finden Sie möglicherweise viele Blogs, Dokumentationen, die Sie verwirren können. In diesem Fall empfehle ich Ihnen, sich an die offiziellen Dokumente von Firebase-Google zu halten. Auch wenn Sie Dritte dazu verwenden möchten, befolgen Sie zuerst diese Dokumentation, da Sie so eine klare Vorstellung davon bekommen, wie die Dinge funktionieren.
- Es dauert etwa 24 Stunden, bis Google Ihre Inhalte indiziert. Also sei geduldig. Sie können Tests durchführen, um sicherzustellen, dass auf Ihrer Seite alles in Ordnung ist.
- Im ersten Beispiel können Sie die HTTP-URL Ihrer Website unterstützen, um sie in Ihrer App umzuleiten. Dies funktioniert beispielsweise, wenn Sie eine Suchanfrage in der Google-Suche durchsucht haben. Die Ergebnisse zeigen eine URL Ihrer Website an, deren App-Links in Ihrer bereits installierten App vorhanden sind. Wenn Sie auf diese URL klicken, werden Sie direkt in Ihrem App-Bildschirm entsprechend dem Suchergebnis weitergeleitet. Das ist es, was ich dafür entdeckt habe.
- Durch das Hinzufügen der AppIndexing-API werden Ihre Inhalte indiziert und bei der automatischen Vervollständigung in der Google-Suchleiste verwendet. Nehmen wir ein Beispiel für eine inShorts-Anwendung für jede Seite. Es gibt eine Überschrift und eine kleine Beschreibung. Schließen Sie die Anwendung nach dem Lesen von 2 oder 3 Überschriften und wechseln Sie zu google searchBar.



Versuchen Sie, eine Schlagzeile einzugeben, die Sie gerade durchgesehen haben. Sie erhalten einen Vorschlag für eine App-Seite mit dieser Überschrift als Titel. Dies unterscheidet sich von App-Vorschlägen, die Sie beim Suchen nach Apps erhalten. Dies geschieht, weil Sie AppIndexing-API-Code für diese bestimmte Seite geschrieben haben und der Titel mit dem in `onCreate()` initialisiert ist.


5:39
sma
smartbytes
smart
smartprix
Smart watchband lets users make calls by touching ear

Geben Sie die Absichtsaktion ACTION_VIEW an, damit der Absichtsfilter über die Google-Suche erreicht werden kann.

<data> Fügen Sie ein oder mehrere Tags hinzu, wobei jedes Tag ein URI-Format darstellt, das in die Aktivität aufgelöst wird. Das Tag muss mindestens das Attribut android: scheme enthalten. Sie können zusätzliche Attribute hinzufügen, um den von der Aktivität akzeptierten URI-Typ weiter zu verfeinern. Möglicherweise verfügen Sie über mehrere Aktivitäten, die ähnliche URIs akzeptieren, die sich jedoch einfach nach dem Pfadnamen unterscheiden. Verwenden Sie in diesem Fall das Attribut android: path oder dessen Varianten (pathPattern oder pathPrefix), um zu unterscheiden, welche Aktivität das System für verschiedene URI-Pfade öffnen soll.

<category> Fügen Sie die Kategorie BROWSABLE hinzu. Die Kategorie BROWSABLE ist erforderlich, damit der Absichtsfilter über einen Webbrowser aufgerufen werden kann. Ohne das Klicken auf einen Link in einem Browser kann Ihre App nicht aufgelöst werden. Die Kategorie DEFAULT ist optional, wird jedoch empfohlen. Ohne diese Kategorie kann die Aktivität nur mit einer expliziten Absicht unter Verwendung des Namens der App-Komponente gestartet werden.

Schritt 4: - Behandeln Sie eingehende URLs

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

}

Schritt 5: - Sie können dies testen, indem Sie den Befehl Android Debug Bridge oder Studio-Konfigurationen verwenden. Adb-Befehl: - Starten Sie Ihre Anwendung und führen Sie dann den folgenden Befehl aus: -

```
adb shell am start –a android.intent.action.VIEW –d "{URL}" < package name >
```

Android Studio-Konfigurationen: - **Android Studio> Erstellen> Konfiguration bearbeiten>** Startoptionen **> URL auswählen>** Geben Sie **hier Ihre URL ein>** Übernehmen und testen. Starten Sie Ihre Anwendung, wenn das Fenster "Ausführen" einen Fehler anzeigt App-Links, die im Manifest erwähnt werden, andernfalls werden sie erfolgreich ausgeführt, und auf die Seite weitergeleitet, auf der die URL angegeben ist, sofern angegeben.

AppIndexing-API hinzufügen

Um dies zu einem Projekt hinzuzufügen, können Sie das offizielle Dokument leicht finden, aber in diesem Beispiel möchte ich einige der wichtigsten Bereiche hervorheben, die zu beachten sind.

Schritt 1: - Google-Dienst hinzufügen

```
dependencies {
      ...
      compile 'com.google.android.gms:play-services-appindexing:9.4.0'
      ...
    }
```

Schritt 2: - Klassen importieren

```
import com.google.android.gms.appindexing.Action;
import com.google.android.gms.appindexing.AppIndex;
import com.google.android.gms.common.api.GoogleApiClient;
```

Schritt 3: - Fügen Sie App-Indizierungs-API-Aufrufe hinzu

```
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();
}
```

Um dies zu testen, folgen Sie den Schritten 4 in den nachstehenden Anmerkungen.

Indizierung der Firebase-App online lesen:
https://riptutorial.com/de/android/topic/5957/indizierung-der-firebase-app

Kapitel 128: Inhalt Anbieter

Bemerkungen

Inhaltsanbieter verwalten den Zugriff auf strukturierte Daten. Sie kapseln die Daten und bieten Mechanismen zur Definition der Datensicherheit. Inhaltsanbieter sind die Standardschnittstelle, die Daten in einem Prozess mit Code verbindet, der in einem anderen Prozess ausgeführt wird.

Wenn Sie auf Daten in einem Inhaltsanbieter zugreifen möchten, verwenden Sie das `ContentResolver` Objekt im `Context` Ihrer Anwendung, um mit dem Anbieter als Client zu kommunizieren. Das `ContentResolver` Objekt kommuniziert mit dem Provider-Objekt, einer Instanz einer Klasse, die `ContentProvider` implementiert. Das Provider-Objekt empfängt Datenanforderungen von Clients, führt die angeforderte Aktion aus und gibt die Ergebnisse zurück.

Sie müssen keinen eigenen Anbieter entwickeln, wenn Sie Ihre Daten nicht für andere Anwendungen freigeben möchten. Sie benötigen jedoch einen eigenen Anbieter, um benutzerdefinierte Suchvorschläge in Ihrer eigenen Anwendung bereitzustellen. Sie benötigen auch einen eigenen Provider, wenn Sie komplexe Daten oder Dateien aus Ihrer Anwendung in andere Anwendungen kopieren und einfügen möchten.

Android selbst umfasst Inhaltsanbieter, die Daten wie Audio, Video, Bilder und persönliche Kontaktinformationen verwalten. Einige davon sind in der Referenzdokumentation für das `android.provider` Paket aufgeführt. Mit einigen Einschränkungen sind diese Anbieter für jede Android-Anwendung verfügbar.

Examples

Implementieren einer grundlegenden Klasse von Inhaltsanbietern

1) Erstellen Sie eine Vertragsklasse

Eine Vertragsklasse definiert Konstanten, mit deren Hilfe Anwendungen mit den Inhalts-URIs, Spaltennamen, Absichtsaktionen und anderen Funktionen eines Inhaltsanbieters zusammenarbeiten können. Vertragsklassen werden bei einem Anbieter nicht automatisch berücksichtigt. Der Entwickler des Providers muss diese definieren und anderen Entwicklern zur Verfügung stellen.

Ein Anbieter hat normalerweise eine einzige Autorität, die als Android-interner Name dient. Verwenden Sie eine eindeutige Inhaltsautorität, um Konflikte mit anderen Anbietern zu vermeiden. Da diese Empfehlung auch für Android-Paketnamen gilt, können Sie Ihre Anbieterberechtigung als Erweiterung des Namens des Pakets definieren, das den Anbieter enthält. Wenn Ihr Android-Paketname beispielsweise `com.example.appname` , sollten Sie Ihrem Provider die Berechtigung `com.example.appname.provider` .

```
public class MyContract {
    public static final String CONTENT_AUTHORITY = "com.example.myApp";
    public static final String PATH_DATATABLE = "dataTable";
    public static final String TABLE_NAME = "dataTable";
}
```

Ein Inhalts-URI ist ein URI, der Daten in einem Anbieter identifiziert. Inhalts-URIs enthalten den symbolischen Namen des gesamten Providers (seine Autorität) und einen Namen, der auf eine Tabelle oder Datei (einen Pfad) verweist. Der optionale ID-Teil verweist auf eine einzelne Zeile in einer Tabelle. Jede Datenzugriffsmethode von ContentProvider hat einen Inhalts-URI als Argument. Auf diese Weise können Sie die Tabelle, Zeile oder Datei bestimmen, auf die zugegriffen werden soll. Definieren Sie diese in der Vertragsklasse.

```
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) Erstellen Sie die Helfer-Klasse

Eine Hilfsklasse verwaltet die Datenbankerstellung und die Versionsverwaltung.

```
public class DatabaseHelper extends SQLiteOpenHelper {

    // Increment the version when there is a change in the structure of database
    public static final int DATABASE_VERSION = 1;
    // The name of the database in the filesystem, you can choose this to be anything
    public static final String DATABASE_NAME = "weather.db";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // Called when the database is created for the first time. This is where the
        // creation of tables and the initial population of the tables should happen.
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Called when the database needs to be upgraded. The implementation
        // should use this method to drop tables, add tables, or do anything else it
        // needs to upgrade to the new schema version.
    }
}
```

3) Erstellen Sie eine Klasse, die die ContentProvider-Klasse erweitert

```
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;
```

Ein UriMatcher ordnet Autorität und Pfad einem ganzzahligen Wert zu. Die Methode `match()` gibt einen eindeutigen ganzzahligen Wert für eine URI zurück (es kann eine beliebige Zahl sein, solange sie eindeutig ist). Eine switch-Anweisung wählt zwischen der Abfrage der gesamten Tabelle und der Abfrage eines einzelnen Datensatzes. Unser UriMatcher gibt 100 zurück, wenn der URI der Inhalts-URI von Table ist, und 101, wenn der URI auf eine bestimmte Zeile in dieser Tabelle verweist. Sie können das Platzhalterzeichen `#` , um mit einer beliebigen Zahl abzugleichen, und `*` , um mit einer beliebigen Zeichenfolge übereinzustimmen.

```
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;
}
```

WICHTIG: Die Bestellung von `addURI()` ruft das hervor! Der UriMatcher sucht in sequenzieller Reihenfolge vom ersten bis zum letzten Hinzufügen. Da Platzhalter wie `#` und `*` gierig sind, müssen Sie sicherstellen, dass Sie Ihre URIs richtig bestellt haben. Zum Beispiel:

```
uriMatcher.addURI(CONTENT_AUTHORITY, "/example", 1);
uriMatcher.addURI(CONTENT_AUTHORITY, "/*",       2);
```

ist die richtige Reihenfolge, da der Matcher zuerst nach `/example` sucht, bevor er zum `/*` -Match greift. Wenn diese Methodenaufrufe rückgängig gemacht wurden und Sie `uriMatcher.match("/example")` aufgerufen haben, `uriMatcher.match("/example")` der UriMatcher auf, nach Übereinstimmungen zu suchen, sobald er auf den Pfad `/*` trifft, und liefert das falsche Ergebnis!

Sie müssen dann diese Funktionen überschreiben:

onCreate () : Initialisieren Sie Ihren Provider. Das Android-System ruft diese Methode unmittelbar nach der Erstellung Ihres Providers auf. Beachten Sie, dass Ihr Provider erst erstellt wird, wenn ein ContentResolver-Objekt versucht, darauf zuzugreifen.

```
@Override
public boolean onCreate() {
    dbhelper = new DatabaseHelper(getContext());
    return true;
}
```

getType () : **Gibt** den MIME-Typ zurück, der einem Inhalts-URI entspricht

```
@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);

    }
}
```

Abfrage () : Rufen Sie Daten von Ihrem Provider ab. Verwenden Sie die Argumente, um die abzurufende Tabelle, die zurückzugebenden Zeilen und Spalten und die Sortierreihenfolge des Ergebnisses auszuwählen. Gibt die Daten als Cursorobjekt zurück.

```
@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;
}
```

Fügen Sie eine neue Zeile in Ihren Provider ein. Verwenden Sie die Argumente, um die Zieltabelle auszuwählen und die zu verwendenden Spaltenwerte abzurufen. Gibt einen Inhalts-URI für die neu eingefügte Zeile zurück.

```
@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 () : Löscht Zeilen von Ihrem Provider. Verwenden Sie die Argumente, um die Tabelle und die zu löschenden Zeilen auszuwählen. Gibt die Anzahl der gelöschten Zeilen zurück.

```
@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 () : Aktualisieren Sie vorhandene Zeilen in Ihrem Provider. Verwenden Sie die Argumente, um die zu aktualisierende Tabelle und Zeilen auszuwählen und die neuen Spaltenwerte abzurufen. Gibt die Anzahl der aktualisierten Zeilen zurück.

```
@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) Manifestdatei aktualisieren

```
<provider
        android:authorities="com.example.myApp"
        android:name=".DatabaseProvider"/>
```

Inhalt Anbieter online lesen: https://riptutorial.com/de/android/topic/3075/inhalt-anbieter

Kapitel 129: Integrieren Sie OpenCV in Android Studio

Bemerkungen

Die Open CV-Bibliotheken können mithilfe einer Suchmaschine im Internet gefunden werden.

Die **Gotchas** :

- Wenn Sie Ihre Zielplattform unter KitKat senken, funktionieren einige der OpenCV-Bibliotheken nicht mehr, insbesondere die mit *org.opencv.android.Camera2Renderer* und anderen verwandten Klassen verbundenen Klassen. Sie können dies wahrscheinlich umgehen, indem Sie einfach die entsprechenden Java-Dateien von OpenCV entfernen.
- Wenn Sie Ihre Zielplattform auf Lollipop oder höher einstellen, funktioniert mein Beispiel für das Laden einer Datei möglicherweise nicht, da die Verwendung absoluter Dateipfade nicht erlaubt ist. Möglicherweise müssen Sie das Beispiel ändern, um eine Datei aus der Galerie oder anderswo zu laden. Es gibt zahlreiche Beispiele.

Examples

Anleitung

Getestet mit AS v1.4.1, sollte aber auch mit neueren Versionen funktionieren.

1. Erstellen Sie ein neues Android Studio-Projekt mit dem Projektassistenten (Menü: / Datei / Neues Projekt):

    - Nennen Sie es " **cvtest1** "
    - Formfaktor: **API 19, Android 4.4 (KitKat)**
    - **Leere Aktivität mit dem** Namen **MainActivity**

    Sie sollten über ein Verzeichnis *cvtest1* verfügen, in dem dieses Projekt gespeichert ist. (Die Titelleiste des Android-Studios zeigt an, wo sich cvtest1 befindet, wenn Sie das Projekt öffnen.)

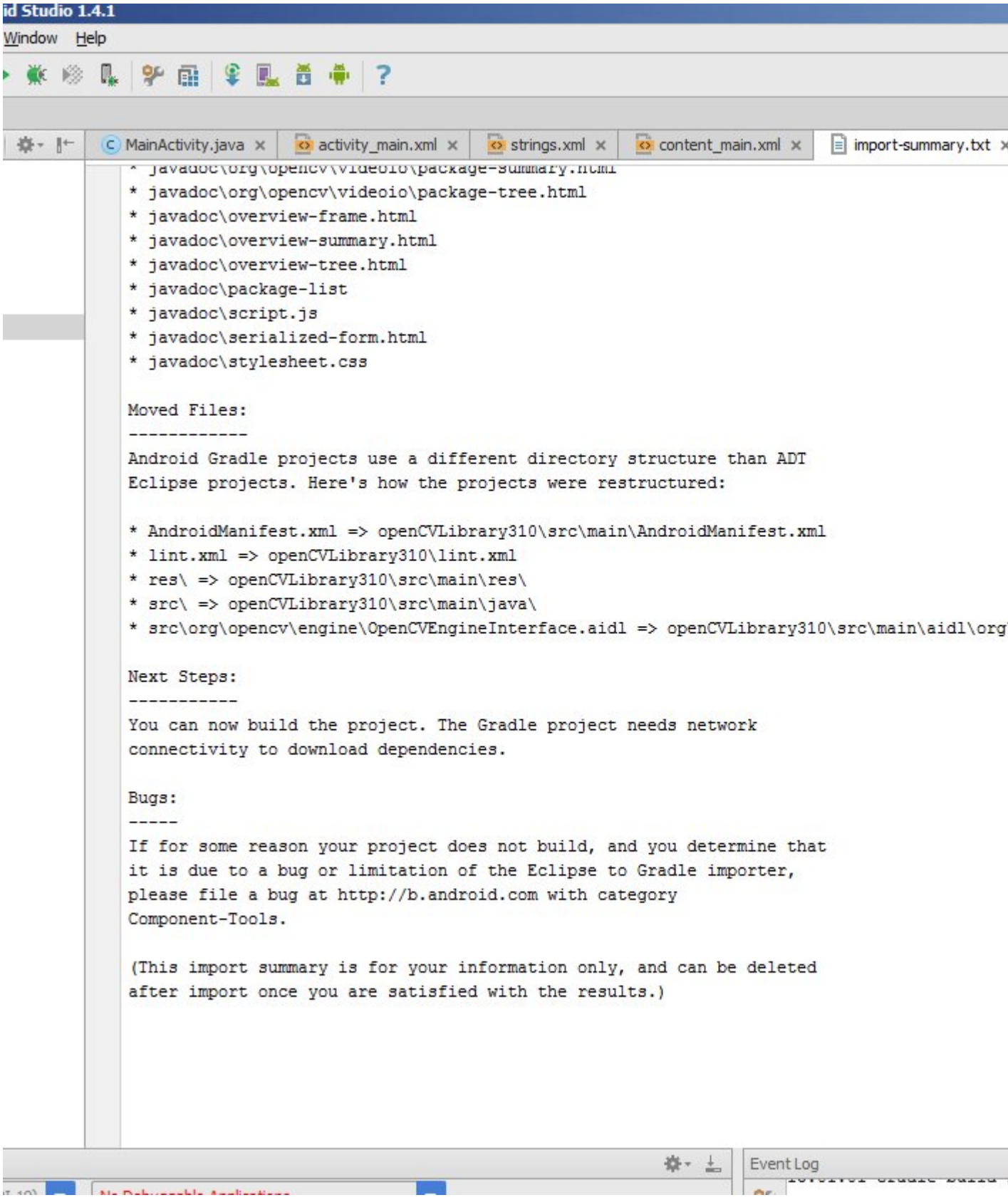
2. Stellen Sie sicher, dass Ihre App ordnungsgemäß ausgeführt wird. Ändern Sie den Text "Hello World", um zu bestätigen, dass der Build- / Testzyklus für Sie in Ordnung ist. (Ich teste mit einem Emulator eines API 19-Geräts).

3. Laden Sie das OpenCV-Paket für Android v3.1.0 herunter und entpacken Sie es in einem temporären Verzeichnis. (Stellen Sie sicher , dass es das Paket ist speziell für Android und nicht nur die OpenCV für Java - Paket.) Ich werde dieses Verzeichnis als **„unzip-dir“** Below **unzip-dir** sollten Sie eine **sdk** haben **/ native / libs** Verzeichnis mit Unterverzeichnissen , die mit starten Dinge wie **Arm** ..., **Mips** ... und **x86** ... (eine für jeden Typ von "Architektur", auf dem Android läuft)

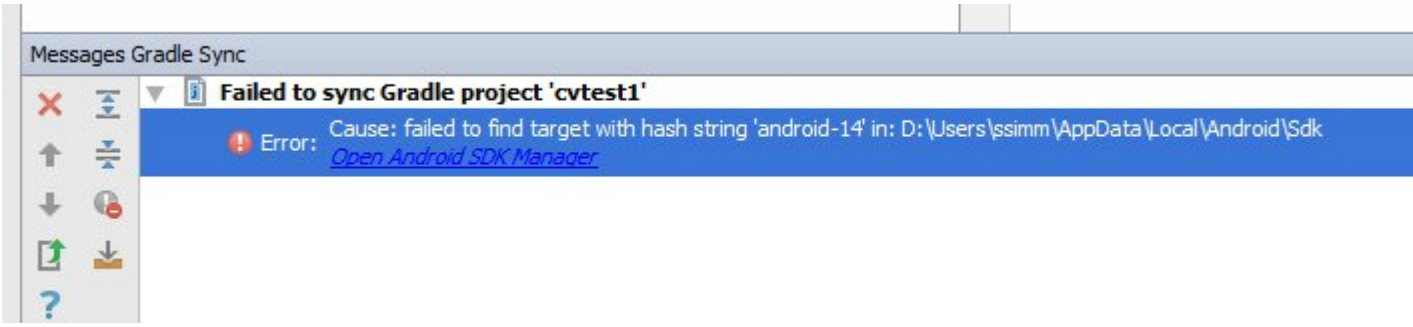
4. Importieren Sie OpenCV aus Android Studio als Modul in Ihr Projekt: **Menü: / Datei / Neu / Import_Module** :

    - **Quellverzeichnis** : **{unzip-dir} / sdk / java**
    - **Modulname** : Android Studio füllt dieses Feld automatisch mit **openCVLibrary310 aus** (der genaue Name ist wahrscheinlich **unwichtig** , aber wir gehen davon aus).
    - Klicken Sie auf **Weiter** . Sie erhalten einen Bildschirm mit drei Kontrollkästchen und Fragen zu Gläsern, Bibliotheken und Importoptionen. Alle drei sollten überprüft werden. Klicken Sie auf **Fertig stellen.**

    Android Studio beginnt mit dem Import des Moduls, und es wird eine Datei **import-summary.txt angezeigt** , die eine Liste der nicht importierten Dateien (hauptsächlich Javadoc-Dateien) und anderer Informationen enthält.



Sie erhalten jedoch auch eine Fehlermeldung, dass **das Ziel mit der Hash-Zeichenfolge 'android-14' nicht gefunden werden konnte** . Dies ist darauf zurückzuführen, dass die Datei build.gradle in der von Ihnen heruntergeladenen OpenCV-ZIP-Datei mit der Android-API-Version 14 kompiliert wird, über die Sie in Android Studio v1.4.1 standardmäßig nicht verfügen.

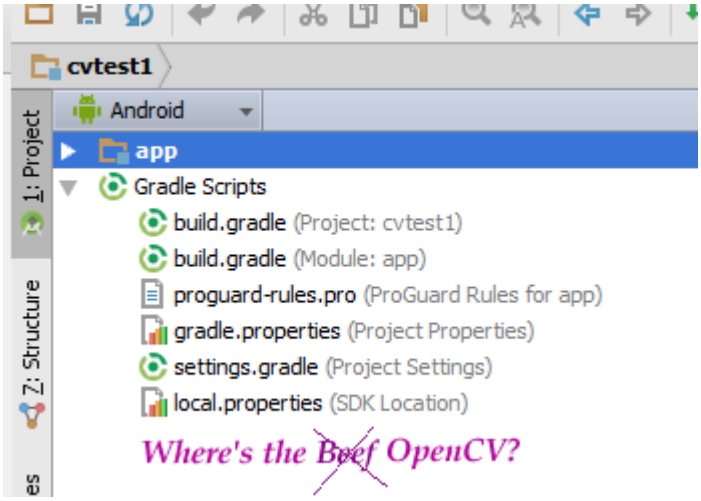


5. Öffnen Sie den Projektstruktur-Dialog (**Menü: / Datei / Projektstruktur**). Wählen Sie das "App" -Modul aus, klicken Sie auf die Registerkarte " **Abhängigkeiten"** und fügen Sie **Folgendes** hinzu **: openCVLibrary310** als **Modulabhängigkeit** . Wenn Sie **Add / Module_Dependency** auswählen, sollte es in der Liste der Module **angezeigt** werden, die Sie hinzufügen können. Es wird jetzt als Abhängigkeit **angezeigt** , aber im Ereignisprotokoll werden einige weitere " **not-find-android-14"** -Fehler angezeigt.

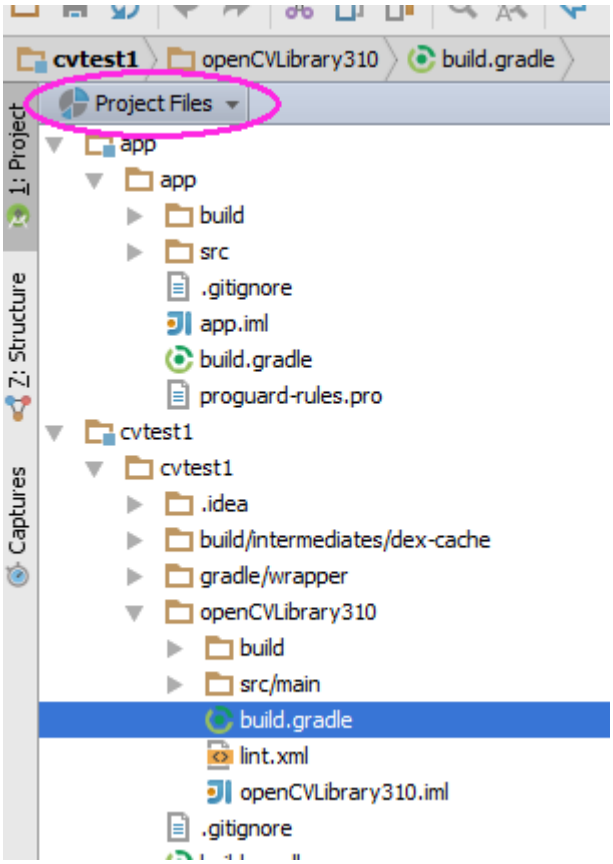
6. Suchen Sie in der Datei **build.gradle** nach Ihrem App-Modul. In einem Android-Projekt gibt es mehrere build.gradle-Dateien. Die gewünschte **Datei** befindet sich im **Verzeichnis cvtest1 / app** und sieht in der Projektansicht wie **build.gradle (Module: app) aus** . Beachten Sie die Werte dieser vier Felder:

    - compileSDKVersion (mein sagt 23)
    - buildToolsVersion (meine sagt 23.0.2)
    - minSdkVersion (meiner sagt 19)
    - targetSdkVersion (meine sagt 23)

7. Ihr Projekt hat jetzt ein **Verzeichnis cvtest1 / OpenCVLibrary310** , ist jedoch in der Projektansicht nicht sichtbar:

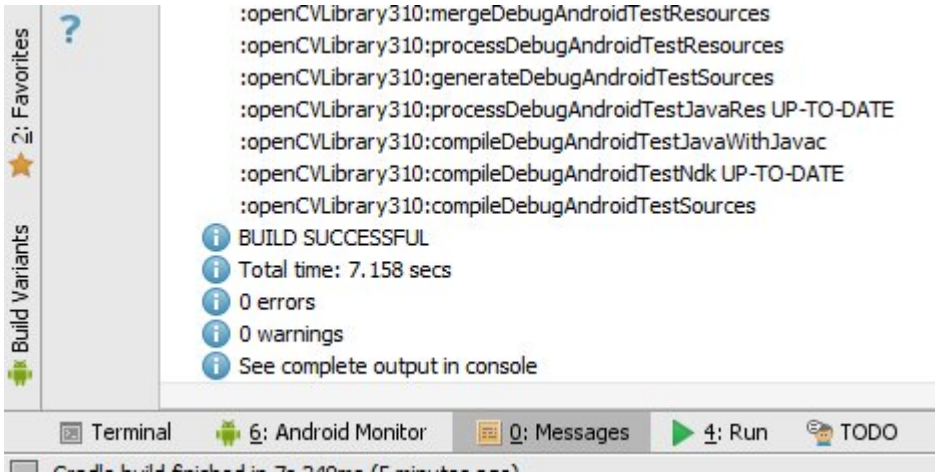


Verwenden Sie ein anderes Tool, beispielsweise einen Dateimanager, und wechseln Sie in dieses Verzeichnis. Sie können auch die Projektansicht von **Android** auf **Projektdateien** umstellen und dieses Verzeichnis wie in diesem Screenshot gezeigt finden:



Darin befindet sich eine weitere **build.gradle-** Datei (diese ist im obigen Screenshot hervorgehoben). Aktualisieren Sie diese Datei mit den vier Werten aus Schritt 6.

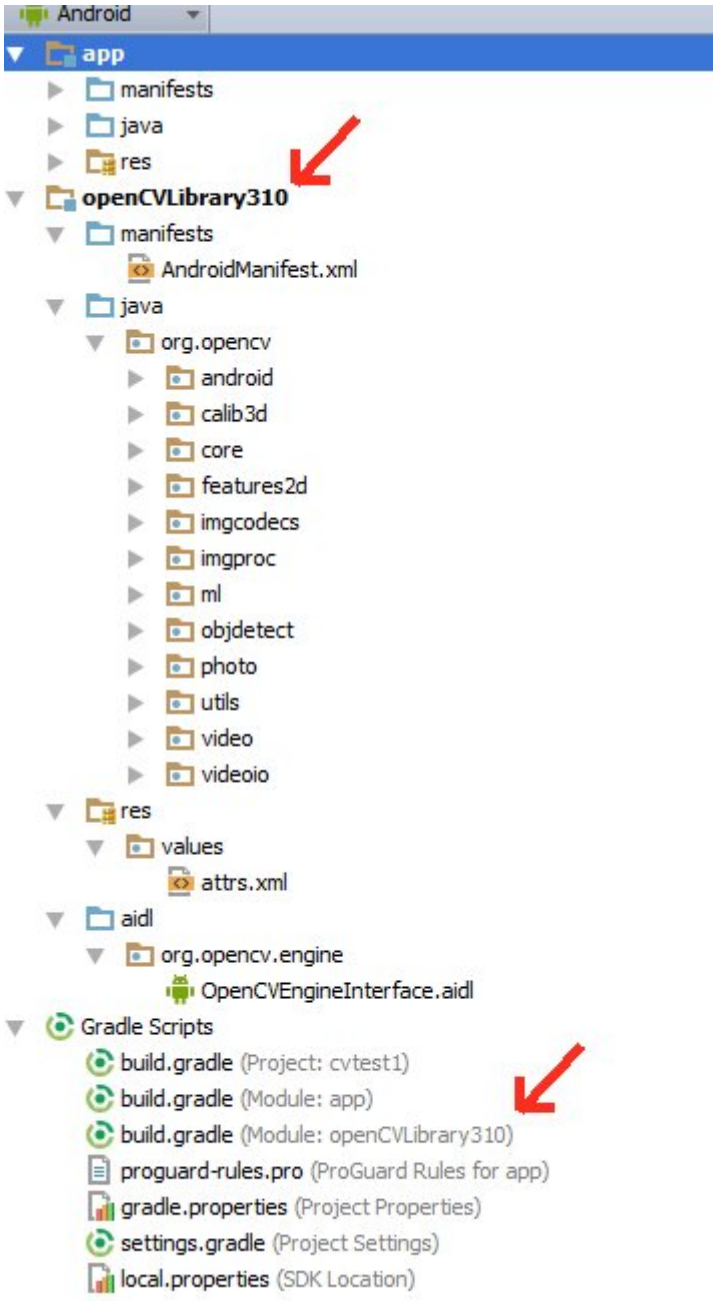
8. Synchronisieren Sie Ihr Projekt erneut und bereinigen / erstellen Sie es anschließend neu. (**Menü: / Build / Clean_Project**) Es sollte ohne Fehler bereinigt und erstellt werden, und es sollten viele Verweise auf **: openCVLibrary310** im Bildschirm **0: Messages** angezeigt werden.



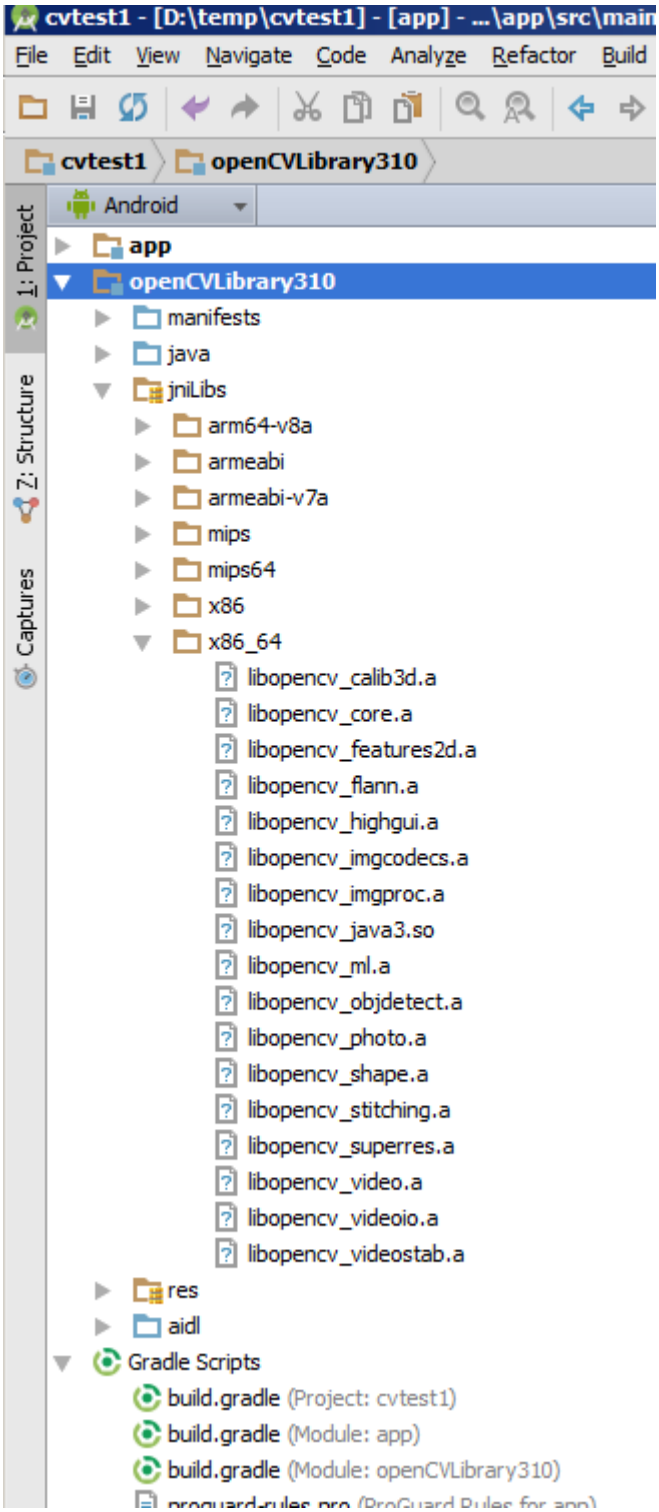
Zu diesem Zeitpunkt sollte das Modul in der Projekthierarchie wie **openCVLibrary310 angezeigt werden** , genau wie die **App** . (Beachten Sie, dass ich in diesem kleinen Dropdown-Menü wieder von der **Projektansicht** zur **Android-Ansicht** gewechselt bin). Sie sollten auch eine zusätzliche **build.gradle** -Datei unter "Gradle Scripts" sehen, aber ich finde die Android Studio-Benutzeroberfläche etwas unangenehm und manchmal tut es das nicht gleich. Versuchen Sie es also erneut, Android Studio neu zu synchronisieren, zu reinigen

und sogar neu zu starten.

Sie sollten das openCVLibrary310-Modul mit allen OpenCV-Funktionen unter Java wie in diesem Screenshot sehen:



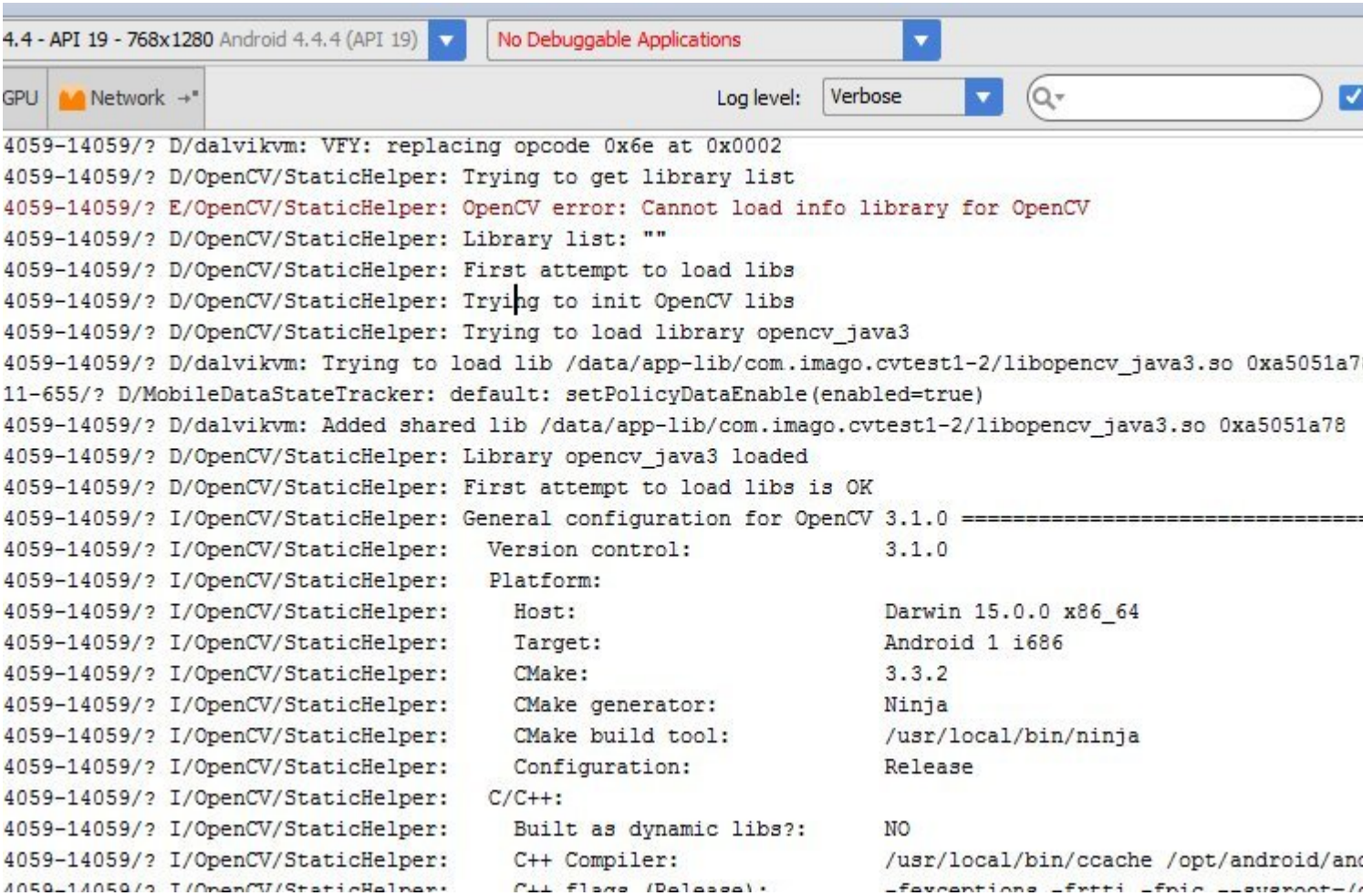
9. Kopieren Sie das **Verzeichnis {unzip-dir} / sdk / native / libs** (und alles darunter) in Ihr Android-Projekt in **cvtest1 / OpenCVLibrary310 / src / main /** und benennen Sie Ihre Kopie von **libs** in **jniLibs um** . Sie sollten jetzt ein **Verzeichnis cvtest1 / OpenCVLibrary310 / src / main / jniLibs haben** . Synchronisieren Sie Ihr Projekt erneut und dieses Verzeichnis sollte jetzt in der Projektansicht unter **openCVLibrary310 angezeigt werden** .



10. Gehen Sie zur onCreate-Methode von *MainActivity.java* und *hängen Sie* diesen Code an:

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

Führen Sie dann Ihre Anwendung aus. Sie sollten solche Zeilen im Android-Monitor sehen:



(Ich weiß nicht, warum diese Zeile mit der Fehlermeldung vorhanden ist)

11. Versuchen Sie nun, tatsächlich OpenCV-Code zu verwenden. Im folgenden Beispiel habe ich eine .jpg-Datei in das Cache-Verzeichnis der cvtest1-Anwendung auf dem Android-Emulator kopiert. Mit dem folgenden Code wird dieses Image geladen, der Erkennungsalgorithmus "canny edge" ausgeführt und die Ergebnisse in eine .png-Datei im selben Verzeichnis zurückgeschrieben.

```
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. Führen Sie Ihre Anwendung aus. Ihr Emulator sollte ein Schwarzweißbild "Rand" erstellen. Sie können den Android-Gerätemonitor verwenden, um die Ausgabe abzurufen oder eine Aktivität zu schreiben, um sie anzuzeigen.

Integrieren Sie OpenCV in Android Studio online lesen:
https://riptutorial.com/de/android/topic/7068/integrieren-sie-opencv-in-android-studio

Kapitel 130: IntentService

Syntax

4. <service android: name = ". UploadS3IntentService" android: exported = "false" />

Bemerkungen

Ein `IntentService` bietet eine einfache Möglichkeit, die Arbeit in einem Hintergrundthread `IntentService` . Es erledigt alles für Sie, wenn Sie Anfragen empfangen, in eine Warteschlange stellen, sich selbst stoppen usw. Es ist auch einfach zu implementieren und daher ideal für zeitaufwendige Vorgänge, die nicht zum Main (UI) -Thread gehören.

Examples

Einen IntentService erstellen

Erstellen Sie zum Erstellen eines IntentService eine Klasse, die `IntentService` , und darin eine Methode, die `onHandleIntent` überschreibt:

```
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.
     }
}
```

Beispielabsichts-Service

Hier ist ein Beispiel für einen `IntentService` , der vorgibt, Bilder im Hintergrund zu laden. Zur Implementierung eines `IntentService` müssen Sie `IntentService` einen Konstruktor bereitstellen, der den `super(String) onHandleIntent(Intent)` , und Sie müssen die `onHandleIntent(Intent)` - Methode implementieren.

```
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
    }
}
```

Um einen `IntentService` zu starten, müssen Sie ihm eine `Intent` senden. Sie können dies beispielsweise aus einer `Activity` tun. Natürlich sind Sie nicht darauf beschränkt. Hier ein Beispiel, wie Sie Ihren neuen `Service` aus einer `Activity` abrufen würden.

```
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
```

Die `IntentService` verarbeitet die Daten von seiner `Intent` s sequentiell, so dass Sie mehrere senden `Intent` s , ohne sich Gedanken , ob sie miteinander kollidieren. Es wird jeweils nur eine `Intent` verarbeitet, der Rest wird in eine Warteschlange gestellt. Wenn alle Jobs abgeschlossen sind, wird der `IntentService` automatisch heruntergefahren.

Grundlegendes IntentService-Beispiel

Die abstrakte Klasse `IntentService` ist eine Basisklasse für Dienste, die im Hintergrund ohne Benutzeroberfläche ausgeführt werden. Um die Benutzeroberfläche zu aktualisieren, müssen wir daher einen Empfänger verwenden, der entweder ein `BroadcastReceiver` oder ein `ResultReceiver` :

- Ein `BroadcastReceiver` sollte verwendet werden, wenn Ihr Dienst mit mehreren Komponenten kommunizieren muss, die auf Kommunikation warten möchten.
- Ein `ResultReceiver` : sollte verwendet werden, wenn Ihr Dienst nur mit der übergeordneten Anwendung (dh Ihrer Anwendung) kommunizieren muss.

Innerhalb des `IntentService` haben wir eine Schlüsselmethode, `onHandleIntent()` , in der wir alle Aktionen ausführen, z.

Wenn Sie `IntentService` eigenen `IntentService` verwenden `IntentService` , müssen Sie ihn wie folgt erweitern:

```
public class YourIntentService extends IntentService {
    public YourIntentService () {
```

```
        super("YourIntentService ");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        // TODO: Write your own code here.
    }
}
```

Das Aufrufen / Starten der Aktivität kann wie folgt durchgeführt werden:

```
Intent i = new Intent(this, YourIntentService.class);
startService(i);  // For the service.
startActivity(i); // For the activity; ignore this for now.
```

Ähnlich wie bei jeder Aktivität können Sie zusätzliche Informationen wie Bündeldaten wie folgt übergeben:

```
Intent passDataIntent = new Intent(this, YourIntentService.class);
msgIntent.putExtra("foo","bar");
startService(passDataIntent);
```

`YourIntentService` jetzt an, dass wir einige Daten an die `YourIntentService` Klasse übergeben haben. Basierend auf diesen Daten kann eine Aktion wie folgt ausgeführt werden:

```
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.
        }
    }
}
```

Der obige Code zeigt auch, wie mit Einschränkungen in der `OnHandleIntent()` Methode `OnHandleIntent()` wird.

IntentService online lesen: https://riptutorial.com/de/android/topic/5319/intentservice

Kapitel 131: Inter-App-UI-Tests mit UIAutomator

Syntax

- Instrumentation getInstrumentation ()
- UIDevice UiDevice.getInstance (Instrumentierungsinstrumentierung)
- boolean UIDevice.pressHome ()
- boolean UIDevice.pressBack ()
- boolean UIDevice.pressRecentApps ()
- void UIDevice.wakeUp ()
- boolean UIDevice.swipe (int startX, int startY, int endX, int endY, int step)
- boolean UIDevice.drag (int startX, int startY, int endX, int endY, int step)
- UIObject2 UIDevice.findObject (By.desc (String contentDesc))
- boolean UIObject2.click ()

Bemerkungen

UIAutomator eignet sich besonders zum Testen von User Storys. Sie haben Probleme, wenn Ansichtselemente weder eine eindeutige *Ressourcen-ID* noch eine *Inhaltsangabe haben* . In den meisten Fällen gibt es eine Möglichkeit, den Test trotzdem abzuschließen, was sehr viel Zeit in Anspruch nimmt. Wenn Sie den Code Ihrer App beeinflussen können, ist UIAutomator möglicherweise Ihr Testwerkzeug.

Examples

Bereiten Sie Ihr Projekt vor und schreiben Sie den ersten UIAutomator-Test

Fügen Sie die erforderlichen Bibliotheken in den Abhängigkeitsbereich des build.gradle Ihres Android-Moduls ein:

```
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'
}
```

Beachten Sie, dass sich die Versionen natürlich in der Zwischenzeit unterscheiden können.

Danach mit den Änderungen synchronisieren.

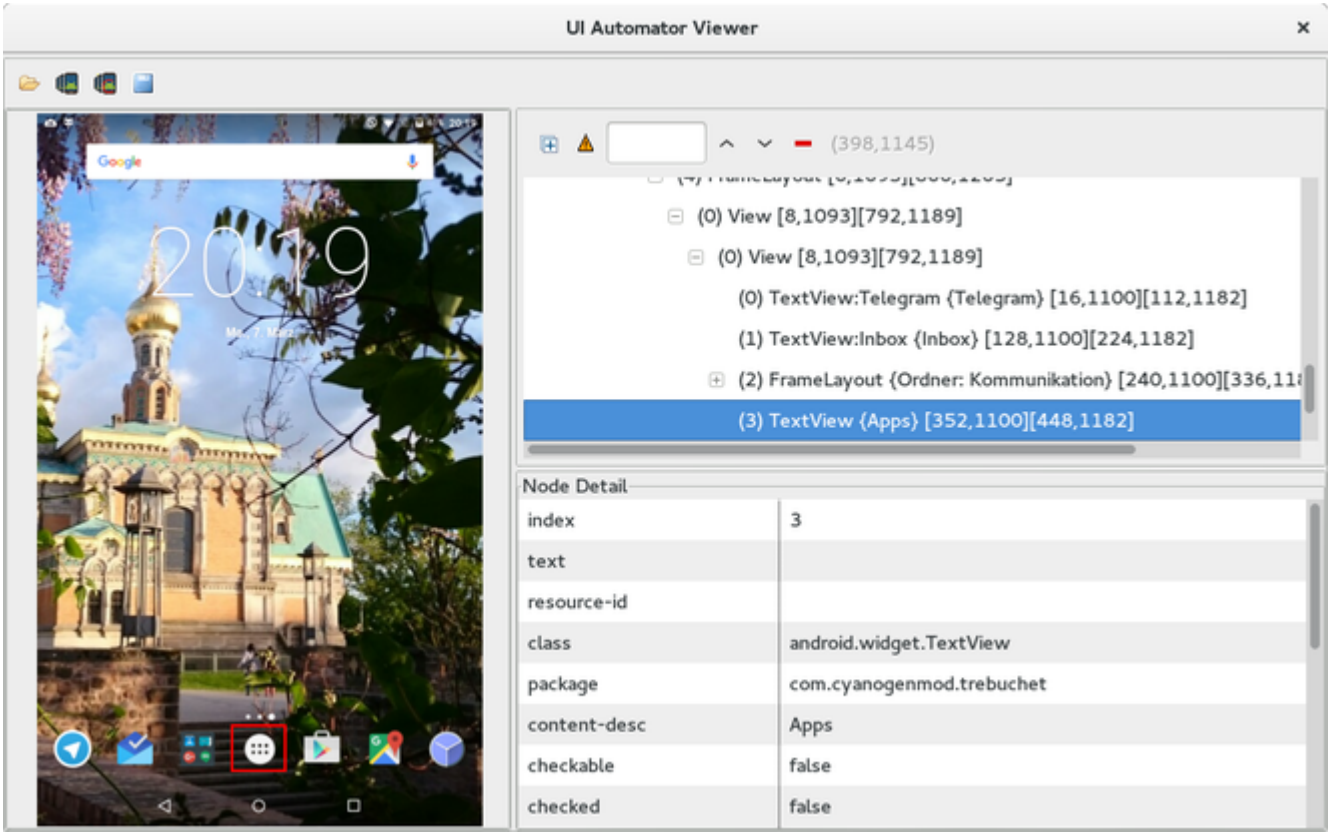
Fügen Sie dann eine neue Java-Klasse im Ordner androidTest hinzu:

```
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();
  }
}
```

Durch einen Rechtsklick auf die Klassenregisterkarte und auf "Ausführen" führt InterAppTest diesen Test aus.

Komplexere Tests mit dem UIAutomatorViewer schreiben

Um komplexere UI-Tests *erstellen zu können,* ist der *UIAutomatorViewer* erforderlich. Das Tool unter */ tools / erstellt* einen Vollbild-Screenshot mit den Layouts der aktuell angezeigten Ansichten. Sehen Sie sich das folgende Bild an, um eine Vorstellung davon zu bekommen, was gezeigt wird:



Für die UI-Tests suchen wir nach *resource-id* , *content-desc* oder etwas anderem, um eine Ansicht

zu identifizieren und in unseren Tests zu verwenden.

Der *uiautomatorviewer* wird über das Terminal ausgeführt.

Wenn wir jetzt zum Beispiel auf die Schaltfläche "Anwendungen" klicken und dann eine App öffnen und herumwischen möchten, kann die Testmethode folgendermaßen aussehen:

```
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)
}
```

Erstellen einer Testsuite von UIAutomator-Tests

UIAutomator-Tests in einer Test-Suite zusammenzufassen, ist eine schnelle Sache:

```
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 {}
```

Führen Sie den Test wie bei einem einzelnen Test aus, indem Sie auf die rechte Maustaste klicken und die Suite ausführen.

Inter-App-UI-Tests mit UIAutomator online lesen:
https://riptutorial.com/de/android/topic/6249/inter-app-ui-tests-mit-uiautomator

Kapitel 132: Internationalisierung und Lokalisierung (I18N und L10N)

Einführung

Internationalisierung (i18n) und Lokalisierung (L10n) werden verwendet, um Software an Unterschiede in den Sprachen, regionalen Unterschieden und Zielgruppen anzupassen.

Internationalisierung: Planungsprozess für die zukünftige Lokalisierung, dh das Softwaredesign so flexibel machen, dass es sich an zukünftige Lokalisierungsbemühungen anpassen und anpassen kann.

Lokalisierung: Der Prozess der Anpassung der Software an eine bestimmte Region / ein bestimmtes Land (einen bestimmten Markt).

Bemerkungen

Um ein Gerät auf Lokalisierung zu testen, können Sie das Gerät oder den Emulator in einem bestimmten Gebietsschema neu starten, indem Sie `adb` wie folgt verwenden:

1. Führen Sie Adb mit dem Befehl: `adb shell`
2. Führen Sie an der Eingabeaufforderung von adb den folgenden Befehl aus: `setprop persist.sys.locale [BCP-47 language tag];stop;sleep 5;start` wobei [BCP-47-Sprachkennzeichen] der hier beschriebene sprachspezifische Code ist: BCP47-Codes

`setprop persist.sys.locale ja-JP;stop;sleep 5;start` die japanische Lokalisierung in der App zu überprüfen, verwenden Sie den Befehl: `setprop persist.sys.locale ja-JP;stop;sleep 5;start`

Examples

Lokalisierung planen: Aktivieren Sie die RTL-Unterstützung in Manifest

Die Unterstützung von RTL (von rechts nach links) ist ein wesentlicher Bestandteil der Planung für i18n und L10n. Im Gegensatz zur englischen Sprache, die von links nach rechts geschrieben wird, werden viele Sprachen wie Arabisch, Japanisch, Hebräisch usw. von rechts nach links geschrieben. Um eine globalere Zielgruppe anzusprechen, empfiehlt es sich, Ihre Layouts so zu planen, dass sie diese Sprache von Anfang an unterstützen, sodass das Hinzufügen von Lokalisierungen später einfacher ist.

Die RTL-Unterstützung kann in einer Android-App aktiviert werden, indem der Tag " `supportsRtl` im `AndroidManifest` wird.

```
<application
    ...
```

```
    android:supportsRtl="true"
    ...>
...
</application>
```

Planen der Lokalisierung: Fügen Sie RTL-Unterstützung in Layouts hinzu

Ab SDK 17 (Android 4.2) wurde die RTL-Unterstützung in Android-Layouts hinzugefügt und ist ein wesentlicher Bestandteil der Lokalisierung. In der Zukunft sollte die `left/right` Notation in Layouts durch die `start/end` Notation ersetzt werden. Wenn Ihr Projekt jedoch einen `minSdk` Wert von weniger als `17` , sollten in Layouts sowohl die `left/right` als auch die `start/end` `minSdk` verwendet werden.

Für relative Layouts sollten `alignParentStart` und `alignParentEnd` wie `alignParentEnd` verwendet werden:

```
<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>
```

Für die Angabe der Schwerkraft und des Layouts der Schwerkraft sollte eine ähnliche Schreibweise verwendet werden, z.

```
    <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"/>
```

Füllungen und Ränder sollten ebenfalls entsprechend angegeben werden:

```
<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"/>
```

Lokalisierungsplanung: Testlayouts für RTL

Gehen Sie folgendermaßen vor, um zu testen, ob die erstellten Layouts RTL-kompatibel sind:

> Gehen Sie zu Einstellungen -> Entwickleroptionen -> Zeichnung -> RTL-Layoutrichtung festlegen

Wenn Sie diese Option aktivieren, muss das Gerät RTL-Gebietsschemas verwenden, und Sie können alle Teile der App für RTL-Unterstützung überprüfen. Beachten Sie, dass Sie bis zu diesem Zeitpunkt keine neuen Gebietsschemas / Sprachunterstützung hinzufügen müssen.

Kodierung für die Lokalisierung: Erstellen von Standardzeichenfolgen und -ressourcen

Der erste Schritt für die Codierung für die Lokalisierung ist das Erstellen von Standardressourcen. Dieser Schritt ist so implizit, dass viele Entwickler nicht einmal darüber nachdenken. Das Erstellen von Standardressourcen ist jedoch wichtig, da das Gerät bei einer nicht unterstützten Ländereinstellung alle Ressourcen aus den Standardordnern lädt. Wenn in den Standardordnern auch nur eine der Ressourcen fehlt, stürzt die App einfach ab.

Der Standardsatz von Zeichenfolgen sollte im angegebenen Ordner am angegebenen Speicherort abgelegt werden:

```
res/values/strings.xml
```

Diese Datei sollte die Zeichenfolgen in der Sprache enthalten, von der die meisten Benutzer der App voraussichtlich sprechen werden.

Standardressourcen für die App sollten auch in den folgenden Ordnern und Speicherorten abgelegt werden:

```
res/drawable/
res/layout/
```

Wenn für Ihre App Ordner wie `anim` oder `xml` erforderlich sind, sollten die Standardressourcen den folgenden Ordnern und Speicherorten hinzugefügt werden:

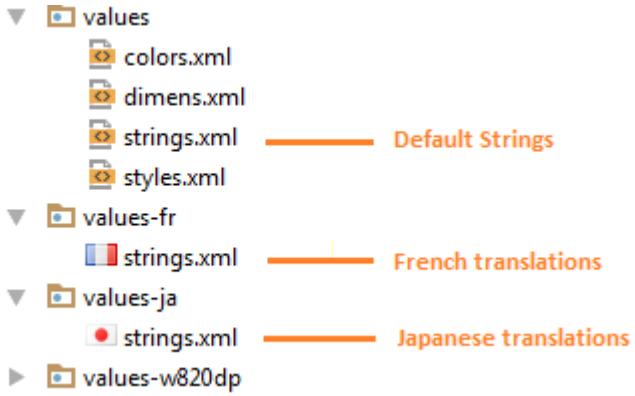
```
res/anim/
res/xml/
res/raw/
```

Kodierung für die Lokalisierung: Bereitstellung alternativer Zeichenketten

Um Übersetzungen in anderen Sprachen (Gebietsschemas) bereitzustellen, müssen wir nach folgender Konvention eine `strings.xml` in einem separaten Ordner erstellen:

```
res/values-<locale>/strings.xml
```

Ein Beispiel für dasselbe ist unten angegeben:



In diesem Beispiel haben wir standardmäßige englische Zeichenfolgen in der Datei `res/values/strings.xml` . Französische Übersetzungen werden im Ordner `res/values-fr/strings.xml` und japanische Übersetzungen im Ordner `res/values-ja/strings.xml` bereitgestellt. `res/values-ja/strings.xml`

Andere Übersetzungen für andere Sprachumgebungen können ebenfalls zur App hinzugefügt werden.

> Eine vollständige Liste der Ländereinstellungscodes finden Sie hier: ISO 639-Codes

Nicht übersetzbare Zeichenketten:

Ihr Projekt enthält möglicherweise bestimmte Zeichenfolgen, die nicht übersetzt werden sollen. In diese Kategorie fallen Zeichenfolgen, die als Schlüssel für SharedPreferences verwendet werden, oder Zeichenfolgen, die als Symbole verwendet werden. Diese Strings sollten nur in dem Standard gespeichert werden `strings.xml` und sollen mit einem markiert `translatable="false"` - Attribut. z.B

```
<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>
```

Dieses Attribut ist wichtig, da Übersetzungen häufig von zweisprachigen Fachleuten ausgeführt werden. Dies würde es den an Übersetzungen beteiligten Personen ermöglichen, Zeichenketten zu identifizieren, die nicht übersetzt werden sollen, wodurch Zeit und Geld gespart werden.

Kodierung für die Lokalisierung: Bereitstellung alternativer Layouts

Das Erstellen sprachspezifischer Layouts ist häufig nicht erforderlich, wenn Sie die richtige `start/end` Endnotation angegeben haben, wie im vorherigen Beispiel beschrieben. Es kann jedoch Situationen geben, in denen die Standardlayouts für bestimmte Sprachen möglicherweise nicht ordnungsgemäß funktionieren. Links-nach-Rechts-Layouts werden möglicherweise nicht für RTL-Sprachen übersetzt. In solchen Fällen müssen die korrekten Layouts angegeben werden.

Um eine vollständige Optimierung für RTL-Layouts zu ermöglichen, können Sie vollständig separate Layoutdateien mit dem `ldrtl` Ressourcenqualifizierer verwenden (`ldrtl` steht für layout-direction-right-to-left}). Zum Beispiel können wir Ihre Standard-Layoutdateien in `res/layout/` und unsere RTL-optimierten Layouts in `res/layout-ldrtl/` .

Das `ldrtl` Qualifikationsmerkmal eignet sich hervorragend für gezeichnete Ressourcen, sodass Sie Grafiken bereitstellen können, die in der der Leserichtung entsprechenden Richtung ausgerichtet sind.

Hier ist ein großartiger Beitrag, der die Rangfolge der `ldrtl` Layouts beschreibt: Sprachspezifische Layouts

Internationalisierung und Lokalisierung (I18N und L10N) online lesen: https://riptutorial.com/de/android/topic/8796/internationalisierung-und-lokalisierung--i18n-und-l10n-

Kapitel 133: Jackson

Einführung

Jackson ist eine vielseitige Java-Bibliothek für die Verarbeitung von JSON. Jackson zielt darauf ab, die bestmögliche Kombination aus schnellen, korrekten, leichten und ergonomischen Eigenschaften für Entwickler zu sein.

Jackson-Funktionen:

Multi-Bearbeitungsmodus und sehr gute Zusammenarbeit

Nicht nur Anmerkungen, sondern auch gemischte Anmerkungen

Volle Unterstützung für generische Typen

Polymorphe Typen unterstützen

Examples

Vollständiges Datenbindungsbeispiel

JSON-Daten

```
{
  "name" : { "first" : "Joe", "last" : "Sixpack" },
  "gender" : "MALE",
  "verified" : false,
  "userImage" : "keliuyue"
}
```

Es sind zwei Java-Zeilen erforderlich, um daraus eine Benutzerinstanz zu machen:

```
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;
    }
}
```

Das Zurückrufen zu JSON ist ähnlich unkompliziert:

```
mapper.writeValue(new File("user-modified.json"), user);
```

Jackson online lesen: https://riptutorial.com/de/android/topic/10878/jackson

Kapitel 134: Java auf Android

Einführung

Android unterstützt alle Java 7-Sprachfunktionen und einen Teil der Java 8-Sprachfunktionen, die je nach Plattformversion variieren. Auf dieser Seite werden die neuen Sprachfunktionen beschrieben, die Sie verwenden können, wie Sie Ihr Projekt ordnungsgemäß konfigurieren und alle bekannten Probleme, auf die Sie stoßen können.

Examples

Java 8-Funktionen mit Retrolambda

Mit Retrolambda können Sie Java 8-Code mit Lambda-Ausdrücken, Methodenreferenzen und try-with-resources-Anweisungen auf Java 7, 6 oder 5 ausführen. Er konvertiert Ihren kompilierten Java 8-Bytecode so, dass er auf einer älteren Java-Laufzeitumgebung ausgeführt werden kann.

Backported-Sprachfunktionen:

- Lambda-Ausdrücke werden zurückportiert, indem sie in anonyme innere Klassen umgewandelt werden. Dies beinhaltet die Optimierung der Verwendung einer Singleton-Instanz für zustandslose Lambda-Ausdrücke, um eine wiederholte Objektzuordnung zu vermeiden. Methodenreferenzen sind im Wesentlichen nur Syntaxzucker für Lambda-Ausdrücke und werden auf dieselbe Weise zurückportiert.
- Try-with-resources-Anweisungen werden `Throwable.addSuppressed` indem Aufrufe an `Throwable.addSuppressed` wenn die Zielcode-Version unter Java 7 liegt. Wenn die unterdrückten Ausnahmen protokolliert und nicht verschluckt werden sollen, erstellen Sie eine Funktionsanforderung, und wir erstellen sie konfigurierbar
- Aufrufe von `Objects.requireNonNull` werden durch Aufrufe von `Object.getClass` wenn die Zielcode-Version unter Java 7 liegt. Die von JDK 9 generierten synthetischen `Objects.requireNonNull` verwenden `Objects.requireNonNull` , während in früheren JDK-Versionen `Object.getClass` .
- Optional auch:
    1. Standardmethoden werden zurückportiert, indem die Standardmethoden als statische Methoden in eine Companion-Klasse (Schnittstellenname + "$") kopiert werden, die Standardmethoden in der Schnittstelle durch abstrakte Methoden ersetzt werden und indem allen Klassen, die diese Schnittstelle implementieren, die erforderlichen Methodenimplementierungen hinzugefügt werden .
    2. Statische Methoden für Schnittstellen werden zurückportiert, indem die statischen Methoden in eine Companion-Klasse (Schnittstellenname + "$") verschoben und alle Methodenaufrufe geändert werden, um den neuen Methodenstandort aufzurufen.

Bekannte Einschränkungen:

- Es werden keine Java 8-APIs zurückportiert
- Das Zurückschreiben von Standardmethoden und statischen Methoden auf Schnittstellen erfordert, dass alle zurückgeschalteten Schnittstellen und alle Klassen, die sie implementieren oder deren statische Methoden aufrufen, zusammen mit einer Ausführung von Retrolambda zurückportiert werden. Mit anderen Worten, Sie müssen immer einen sauberen Build durchführen. Backporting-Standardmethoden funktionieren auch nicht über Modul- oder Abhängigkeitsgrenzen hinweg.
- `invokedynamic` , wenn ein zukünftiger JDK 8-Build die Generierung einer neuen Klasse für jeden `invokedynamic` Aufruf beendet. Retrolambda arbeitet so, dass es den Bytecode erfasst, den `java.lang.invoke.LambdaMetafactory` dynamisch generiert, sodass Optimierungen dieses Mechanismus Retrolambda möglicherweise brechen.

Retrolambda Gradle Plugin erstellt automatisch ein Android-Projekt mit Retrolambda. Die neueste Version finden Sie auf der Freigabeseite .

Verwendungszweck:

1. Laden Sie jdk8 herunter und installieren Sie es
2. Fügen Sie Folgendes zu Ihrem `build.gradle`

```
buildscript {
  repositories {
     mavenCentral()
  }

  dependencies {
     classpath 'me.tatarka:gradle-retrolambda:<latest version>'
  }
}

// Required because retrolambda is on maven central
repositories {
  mavenCentral()
}

apply plugin: 'com.android.application' //or apply plugin: 'java'
apply plugin: 'me.tatarka.retrolambda'

android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

Bekannte Probleme:

- Flusen schlagen bei Java-Dateien mit Lambdas fehl. Die Fusseln von Android verstehen die Java 8-Syntax nicht und versagen lautlos oder lautlos. Es gibt jetzt eine experimentelle Gabel, die das Problem behebt.

- Durch die Verwendung der Google Play-Dienste schlägt Retrolambda fehl. Version 5.0.77 enthält einen mit Retrolambda nicht kompatiblen Bytecode. Dies sollte in neueren Versionen von Play Services behoben werden. Wenn Sie aktualisieren können, sollte dies die bevorzugte Lösung sein. Um dieses Problem zu umgehen, können Sie entweder eine frühere Version wie 4.4.52 verwenden oder `-noverify` zu den jvm-Argumenten hinzufügen.

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

Java auf Android online lesen: https://riptutorial.com/de/android/topic/9223/java-auf-android

Kapitel 135: JCodec

Examples

Fertig machen

Sie können JCodec automatisch mit maven erhalten. Fügen Sie dazu einfach einen Ausschnitt aus Ihrer pom.xml hinzu.

```
<dependency>
    <groupId>org.jcodec</groupId>
    <artifactId>jcodec-javase</artifactId>
    <version>0.1.9</version>
</dependency>
```

Bild vom Film aufnehmen

Ein einzelnes Bild aus einem Film abrufen (unterstützt nur AVC, H.264 in MP4, ISO BMF, Quicktime-Container):

```
int frameNumber = 150;
BufferedImage frame = FrameGrab.getFrame(new File("filename.mp4"), frameNumber);
ImageIO.write(frame, "png", new File("frame_150.png"));
```

Holen einer Sequenz von Bildern aus einem Film (unterstützt nur AVC, H.264 in MP4, ISO BMF, Quicktime-Container):

```
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);
}
```

JCodec online lesen: https://riptutorial.com/de/android/topic/9948/jcodec

Kapitel 136: Jenkins CI-Setup für Android-Projekte

Examples

Schritt für Schritt zum Einrichten von Jenkins für Android

Dies ist eine schrittweise Anleitung zum Einrichten des automatisierten Erstellungsprozesses mit Jenkins CI für Ihre Android-Projekte. Bei den folgenden Schritten wird davon ausgegangen, dass Sie neue Hardware mit einer beliebigen Linux-Variante installiert haben. Es wird auch berücksichtigt, dass Sie möglicherweise eine Remote-Maschine haben.

TEIL I: Erste Einrichtung auf Ihrem Rechner

1. Melden Sie sich über *ssh* an Ihrem Ubuntu-Rechner an:

   ssh benutzername@xxx.xxx.xxx

2. Laden Sie eine Version des Android SDK auf Ihren Computer herunter:

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

3. Entpacken Sie die heruntergeladene *tar* - Datei:

   sudo apt-get install tar
   tar -xvf android-sdk_r24.4.1-linux.tgz

4. Nun müssen Sie Java 8 auf Ihrem Ubuntu-Computer installieren. Dies ist eine Voraussetzung für Android-Builds auf Nougat. Für Jenkins müssen Sie JDK und JRE 7 wie folgt installieren:

   sudo apt-get install python-software-eigenschaften
   sudo add-apt-repository ppa: webupd8team / java
   sudo apt-get update
   apt-get install openjdk-8-jdk

5. Installieren Sie nun Jenkins auf Ihrem Ubuntu-Rechner:

   wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -
   sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary /> /etc/apt/sources.list.d/jenkins.list'
   sudo apt-get update
   sudo apt-get install jenkins

6. Laden Sie die neueste unterstützte Gradle-Version für Ihr Android-Setup herunter:

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

7. Richten Sie Android auf Ihrem Ubuntu-Computer ein. Wechseln Sie zunächst in den *Tools*-Ordner im Android SDK-Ordner, der in Schritt 2 heruntergeladen wurde:

   cd android-sdk-linux / tools **// listet das verfügbare SDK auf**
   android update sdk --no-ui **// Aktualisiert die SDK-Version**
   android list sdk -a | grep "SDK Build-Tools" **// listet verfügbare Build-Tools auf**
   android update sdk -a -u -t 4 **// aktualisiert die Version der Build-Tools auf eine, die in der** vorherigen **Version als 4 aufgeführt ist. cmd.**
   Java aktualisieren

8. Installieren Sie *Git* oder ein anderes VCS auf Ihrem Computer:

   sudo apt-get install git

9. Melden Sie sich jetzt mit Ihrem Internetbrowser bei Jenkins an. `ipAddress:8080` in die Adressleiste ein.

10. Um das Passwort für die erstmalige Anmeldung zu erhalten, überprüfen Sie bitte die entsprechende Datei wie folgt (Sie benötigen zum Zugriff auf diese Datei su).

    cat / var / lib / jenkins / secrets / initialAdminPassword

TEIL II: Richten Sie Jenkins ein, um Android Jobs zu erstellen

1. Gehen Sie nach dem Anmelden zu folgendem Pfad:

   Jenkins> Jenkins verwalten> Globale Werkzeugkonfiguration

2. `JAVA_HOME` an dieser Stelle `JAVA_HOME` mit den folgenden Einträgen hinzu:

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

3. Fügen Sie außerdem die folgenden Werte zu *Git hinzu* und speichern Sie die Umgebungsvariablen:

   Name = Standard
   / usr / bin / git

4. Gehen Sie nun zu folgendem Pfad:

   Jenkins> Jenkins verwalten> Konfiguration

5. `ANDROID_HOME` an dieser Stelle `ANDROID_HOME` zu den "globalen Eigenschaften" hinzu:

Name = ANDROID_HOME
Wert = / home / Benutzername / android-sdk-linux

Teil III: Erstellen Sie einen Jenkins-Job für Ihr Android-Projekt

1. Klicken Sie im Jenkins-Startbildschirm auf *Neues Element* .
2. Fügen Sie einen *Projektnamen* und eine *Beschreibung hinzu* .
3. Wählen Sie auf der Registerkarte *Allgemein die* Option *Erweitert aus* . Wählen *Sie dann Benutzerdefinierten Arbeitsbereich verwenden aus* :

   Verzeichnis / home / user / Code / ProjectFolder
4. Wählen Sie in der Quellcodeverwaltung *Git aus* . Ich benutze *Bitbucket* für dieses Beispiel:

   Repository-URL = https: // Benutzername: Kennwort@bitbucket.org/project/projectname.git
5. Wählen Sie zusätzliche Verhalten für Ihr Repository aus:

   Vor der Kasse reinigen
   Kasse in ein Unterverzeichnis. Lokales Unterverzeichnis für Repo / home / user / Code / ProjectFolder
6. Wählen Sie einen Zweig aus, den Sie erstellen möchten:

   */Meister
7. *Wählen Sie auf der* Registerkarte *Erstellen* die *Option Shell* in *Buildschritt hinzufügen aus* .
8. *Fügen Sie* in der *Execute-Shell* den folgenden Befehl hinzu:

   cd / home / user / Code / ProjectFolder && gradle clean - no-daemon zusammenbauen
9. Wenn Sie Lint für das Projekt ausführen möchten, fügen Sie der *Execute-Shell* einen weiteren Build-Schritt hinzu:

   /home/user/gradle/gradle-2.14.1/bin/gradle fussel

Nun ist Ihr System so eingerichtet, dass Sie Android-Projekte mit Jenkins erstellen können. Dieses Setup erleichtert Ihnen die Freigabe von Builds für QA- und UAT-Teams.

PS: Da Jenkins ein anderer Benutzer auf Ihrem Ubuntu-Computer ist, sollten Sie ihm die Berechtigung zum Erstellen von Ordnern in Ihrem Arbeitsbereich geben, indem Sie den folgenden Befehl ausführen:

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

Jenkins CI-Setup für Android-Projekte online lesen:
https://riptutorial.com/de/android/topic/7830/jenkins-ci-setup-fur-android-projekte

Kapitel 137: JSON in Android mit org.json

Syntax

- **Objekt** : Ein Objekt ist eine ungeordnete Menge von Name / Wert-Paaren. Ein Objekt beginnt mit {(linke Klammer) und endet mit} (rechte Klammer). Jeder Name wird gefolgt von: (Doppelpunkt) und die Name / Wert-Paare werden durch (Komma) getrennt.
- **Array** : Ein Array ist eine geordnete Sammlung von Werten. Ein Array beginnt mit [(linke Klammer) und endet mit] (rechte Klammer). Werte werden durch (Komma) getrennt.
- **Wert** : Ein Wert kann eine Zeichenfolge in doppelten Anführungszeichen oder eine Zahl oder wahr oder falsch oder null oder ein Objekt oder ein Array sein. Diese Strukturen können verschachtelt werden.
- **String:** Ein String ist eine Folge von null oder mehr Unicode - Zeichen in doppelten Anführungszeichen eingewickelt, mit Backslash entkommt. Ein Zeichen wird als einzelne Zeichenfolge dargestellt. Eine Zeichenfolge ist einer C- oder Java-Zeichenfolge sehr ähnlich.
- **Zahl** : Eine Zahl ist einer C- oder Java-Zahl sehr ähnlich, außer dass die Oktal- und Hexadezimalformate nicht verwendet werden.

Bemerkungen

In diesem Thema wird das Paket `org.json` , das im Android SDK enthalten ist.

Examples

Einfaches JSON-Objekt analysieren

Betrachten Sie die folgende JSON-Zeichenfolge:

```
{
  "title": "test",
  "content": "Hello World!!!",
  "year": 2016,
  "names" : [
        "Hannah",
        "David",
        "Steve"
   ]
}
```

Dieses JSON-Objekt kann mit folgendem Code analysiert werden:

```
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());
}
```

Hier ist ein weiteres Beispiel mit einem in JSONObject geschachtelten JSONArray:

```
{
    "books":[
      {
        "title":"Android JSON Parsing",
        "times_sold":186
      }
    ]
}
```

Dies kann mit dem folgenden Code analysiert werden:

```
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");
```

Einfaches JSON-Objekt erstellen

Erstellen Sie das `JSONObject` mithilfe des leeren Konstruktors und fügen Sie Felder mithilfe der Methode `put()` , die überladen ist, sodass sie mit verschiedenen Typen verwendet werden kann:

```
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);
}
```

Die resultierende `JSON` Zeichenfolge sieht folgendermaßen aus:

```
{
```

```
   "name":"test",
   "content":"Hello World!!!1",
   "year":2016,
   "value":3.23,
   "member":true,
   "null_value":null
}
```

Fügen Sie JSONArray zu JSONObject hinzu

```
// 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();
```

Die resultierende JSON-Zeichenfolge sieht folgendermaßen aus:

```
{
   "the_array":[
      "ASDF",
      "QWERTY"
   ]
}
```

Erstellen Sie einen JSON-String mit Nullwert.

Wenn Sie einen JSON-String mit einem Wert von `null` wie folgt erstellen müssen:

```
{
   "name":null
}
```

Dann müssen Sie die spezielle Konstante JSONObject.NULL verwenden .

Funktionsbeispiel:

```
jsonObject.put("name", JSONObject.NULL);
```

Bei der Analyse von Json mit Null-String arbeiten

```
{
    "some_string": null,
    "ather_string": "something"
}
```

Wenn wir diesen Weg verwenden werden:

```
JSONObject json = new JSONObject(jsonStr);
String someString = json.optString("some_string");
```

Wir werden ausgeben:

```
someString = "null";
```

Daher müssen wir diese Problemumgehung bereitstellen:

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

Und dann anrufen:

```
JSONObject json = new JSONObject(jsonStr);
String someString = optNullableString(json, "some_string");
String someString2 = optNullableString(json, "some_string", "");
```

Und wir werden Output wie erwartet haben:

```
someString = null; //not "null"
someString2 = "";
```

JsonReader zum Lesen von JSON aus einem Stream verwenden

`JsonReader`

liest einen JSON-codierten Wert als `JsonReader` .

```
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);
  }
```

Erstellen Sie geschachtelte JSON-Objekte

Um geschachtelte JSON-Objekte zu erstellen, müssen Sie einfach ein JSON-Objekt zu einem anderen hinzufügen:

```
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";
}
```

Jetzt enthält `mainObject` einen Schlüssel namens `claim` mit dem gesamten `requestObject` als Wert.

Umgang mit dynamischen Schlüsseln für die JSON-Antwort

Dies ist ein Beispiel für den Umgang mit dynamischen Schlüsseln für die Antwort. Hier sind `A` und `B` dynamische Schlüssel, es kann alles sein

Antwort

```
{
  "response": [
    {
      "A": [
        {
          "name": "Tango"
        },
        {
          "name": "Ping"
```

```
        }
      ],
      "B": [
        {
          "name": "Jon"
        },
        {
          "name": "Mark"
        }
      ]
    }
  ]
}
```

Java-Code

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

Prüfen Sie, ob Felder in JSON vorhanden sind

Manchmal ist es nützlich zu prüfen, ob ein Feld in Ihrem JSON vorhanden ist oder nicht, um `JSONException` in Ihrem Code zu vermeiden.

Um dies zu erreichen, verwenden Sie das `JSONObject#has(String)` oder die Methode wie im folgenden Beispiel:

Beispiel JSON

```
{
   "name":"James"
}
```

Java-Code

```
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".
```

Aktualisieren der Elemente in der JSON

Beispiel-Json zum Aktualisieren

```
 {
 "student":{"name":"Rahul", "lastname":"sharma"},
 "marks":{"maths":"88"}
 }
```

Um den Elementwert im Json zu aktualisieren, müssen wir den Wert zuweisen und aktualisieren.

```
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);
}
```

aktualisierter Wert

```
{
"student":{"name":"Kumar", "lastname":"sharma"},
"marks":{"maths":"88"}
}
```

JSON in Android mit org.json online lesen: https://riptutorial.com/de/android/topic/106/json-in-android-mit-org-json

Kapitel 138: Kamera 2 API

Parameter

Parameter	Einzelheiten
`CameraCaptureSession`	Eine konfigurierte Aufnahmesitzung für ein `CameraDevice` , die zum `CameraDevice` von Bildern von der Kamera oder zum Wiederaufbereiten von Bildern verwendet wird, die zuvor in derselben Sitzung von der Kamera aufgenommen wurden
`CameraDevice`	Eine Darstellung einer einzelnen Kamera, die mit einem Android-Gerät verbunden ist
`CameraCharacteristics`	Die Eigenschaften, die ein CameraDevice beschreiben. Diese Eigenschaften sind für ein bestimmtes CameraDevice festgelegt und können über die CameraManager-Benutzeroberfläche mit `getCameraCharacteristics(String)` abgefragt werden.
`CameraManager`	Ein System Service Manager zum Erkennen, Charakterisieren und Verbinden mit `CameraDevices` . Sie können eine Instanz dieser Klasse `Context.getSystemService()` indem Sie `Context.getSystemService()` aufrufen.
`CaptureRequest`	Ein unveränderliches Paket von Einstellungen und Ausgaben, das zum Erfassen eines einzelnen Bilds vom Kameragerät benötigt wird. Enthält die Konfiguration für die Erfassungshardware (Sensor, Objektiv, Blitz), die Verarbeitungspipeline, die Steuerungsalgorithmen und die Ausgabepuffer. Enthält auch die Liste der Zieloberflächen, an die Bilddaten für diese Aufnahme gesendet werden sollen. Kann mit einer `CaptureRequest.Builder` Instanz erstellt werden, die durch Aufrufen von `createCaptureRequest(int)` abgerufen wird.
`CaptureResult`	Die Teilmenge der Ergebnisse einer einzelnen Bildaufnahme vom Bildsensor. Enthält eine Teilmenge der endgültigen Konfiguration für die Erfassungshardware (Sensor, Objektiv, Blitzlicht), die Verarbeitungspipeline, die Steuerungsalgorithmen und die Ausgangspuffer. Es wird von einem `CameraDevice` nach der Verarbeitung eines `CaptureRequest`

Bemerkungen

- Camera2-APIs sind in API 21+ (Lollipop und höher) verfügbar.
- Selbst wenn ein Android-Gerät offiziell über 21 ROM-ROMs verfügt, gibt es keine Garantie,

dass Camera2-APIs implementiert werden. Es ist völlig Sache des Herstellers, es zu implementieren oder nicht.
- Bei Camera2 ist Camera ("Camera1") veraltet
- Mit großer Leistung geht eine große Verantwortung einher: Es ist einfacher, diese APIs zu verwenden.
- Denken Sie daran, wenn Sie nur ein Foto in Ihrer App aufnehmen und es einfach erhalten möchten, müssen Sie Camera2 **nicht** implementieren. Sie können die Kamera-App des Geräts über einen Intent öffnen und sie zurückerhalten

Examples

Vorschau der Hauptkamera in einer TextureView

In diesem Fall wird mit API 23 erstellt, sodass auch Berechtigungen behandelt werden.

Sie müssen im Manifest die folgende Berechtigung hinzufügen (wo auch immer die API-Ebene verwendet wird):

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

Wir erstellen gerade eine Aktivität (Camera2Activity.java), die eine `TextureView` mit der Vorschau der Kamera des Geräts füllt.

Die Aktivität, die wir verwenden werden, ist eine typische AppCompatActivity:

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

Attribute (Sie müssen möglicherweise das gesamte Beispiel lesen, um einige davon zu verstehen.)

Die `MAX_PREVIEW_SIZE` von Camera2 API garantiert ist 1920x1080

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

`TextureView.SurfaceTextureListener` verarbeitet mehrere Lebenszyklusereignisse in einer `TextureView` . In diesem Fall hören wir uns diese Ereignisse an. Wenn die SurfaceTexture fertig ist, initialisieren wir die Kamera. Wenn sich die Größe ändert, stellen wir die Vorschau der Kamera entsprechend ein

```
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) {
    }

};
```

Ein `CameraDevice` repräsentiert die Kamera eines physischen Geräts. In diesem Attribut speichern wir die ID des aktuellen `CameraDevice`

```
private String mCameraId;
```

Dies ist die Ansicht (`TextureView`), mit der wir die Vorschau der Kamera "zeichnen"

```
private TextureView mTextureView;
```

Die `CameraCaptureSession` für die `CameraCaptureSession`

```
private CameraCaptureSession mCaptureSession;
```

Ein Verweis auf das geöffnete `CameraDevice`

```
private CameraDevice mCameraDevice;
```

Die `Size` der Kameravorschau.

```
private Size mPreviewSize;
```

`CameraDevice.StateCallback` wird aufgerufen, wenn `CameraDevice` seinen `CameraDevice` ändert

```
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();
    }

};
```

Ein zusätzlicher Thread zum Ausführen von Aufgaben, die die Benutzeroberfläche nicht blockieren sollen

```
private HandlerThread mBackgroundThread;
```

Ein `Handler` zum Ausführen von Aufgaben im Hintergrund

```
private Handler mBackgroundHandler;
```

Ein `ImageReader` , der die Erfassung von Standbildern übernimmt

```
private ImageReader mImageReader;
```

`CaptureRequest.Builder` für die `CaptureRequest.Builder`

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

`CaptureRequest` das von `mPreviewRequestBuilder` generiert wird

```
private CaptureRequest mPreviewRequest;
```

Ein `Semaphore` , um zu verhindern, dass die App beendet wird, bevor die Kamera geschlossen wird.

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

Konstante ID der Berechtigungsanfrage

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

Android-Lebenszyklusmethoden

```
@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 verwandte Methoden

Dies sind Methoden, die die Camera2-APIs verwenden

```
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);
    }
}
```

Schließt die aktuelle Kamera

```
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();
    }
}
```

Legt Mitgliedsvariablen für die Kamera fest

```
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();
    }
}
```

Erstellt eine neue `CameraCaptureSession` für die Kameravorschau

```
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();
    }
}
```

Berechtigungsbezogene Methoden Für 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);
    }
}
```

Hintergrund-Thread / Handler-Methoden

```
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();
    }
}
```

Gebrauchsmethoden

Wählen Sie bei Auswahl der von einer Kamera unterstützten `Size` s die kleinste Größe aus, die mindestens der Gesamtgröße der Texturansicht entspricht und die der maximalen Größe entspricht und deren Seitenverhältnis mit dem angegebenen Wert übereinstimmt. Falls nicht vorhanden, wählen Sie den größten Wert, der höchstens so groß wie die jeweilige maximale Größe ist und dessen Seitenverhältnis mit dem angegebenen Wert übereinstimmt

```
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];
    }
}
```

Diese Methode konfiguriert die erforderliche `Matrix` Umwandlung in `mTextureView`

```
private void configureTransform(int viewWidth, int viewHeight) {
    if (null == mTextureView || null == mPreviewSize) {
        return;
    }
    int rotation = getWindowManager().getDefaultDisplay().getRotation();
    Matrix matrix = new Matrix();
    RectF viewRect = new RectF(0, 0, viewWidth, viewHeight);
    RectF bufferRect = new RectF(0, 0, mPreviewSize.getHeight(), mPreviewSize.getWidth());
    float centerX = viewRect.centerX();
    float centerY = viewRect.centerY();
    if (Surface.ROTATION_90 == rotation || Surface.ROTATION_270 == rotation) {
        bufferRect.offset(centerX - bufferRect.centerX(), centerY - bufferRect.centerY());
        matrix.setRectToRect(viewRect, bufferRect, Matrix.ScaleToFit.FILL);
        float scale = Math.max(
                (float) viewHeight / mPreviewSize.getHeight(),
                (float) viewWidth / mPreviewSize.getWidth());
        matrix.postScale(scale, scale, centerX, centerY);
        matrix.postRotate(90 * (rotation - 2), centerX, centerY);
    } else if (Surface.ROTATION_180 == rotation) {
        matrix.postRotate(180, centerX, centerY);
    }
    mTextureView.setTransform(matrix);
}
```

Diese Methode vergleicht zwei `Size` basierend auf ihren Bereichen.

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

Hier nicht viel zu sehen

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

Kamera 2 API online lesen: https://riptutorial.com/de/android/topic/619/kamera-2-api

Kapitel 139: Kamera und Galerie

Examples

Foto in voller Größe von der Kamera aufnehmen

Um ein Foto aufzunehmen, müssen wir zunächst die erforderlichen Berechtigungen in `AndroidManifest.xml` . Wir benötigen zwei Berechtigungen:

- `Camera` - Kamera-App öffnen Wenn das Attribut `required` auf `true` gesetzt ist, können Sie diese App nicht installieren, wenn Sie keine Hardwarekamera haben.
- `WRITE_EXTERNAL_STORAGE` - Diese Berechtigung ist erforderlich, um eine neue Datei zu erstellen, in der das aufgenommene Foto gespeichert wird.

AndroidManifest.xml

```
<uses-feature android:name="android.hardware.camera"
          android:required="true" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Die Hauptidee bei der Aufnahme eines Fotos in voller Größe von der Kamera ist, dass wir eine neue Datei für das Foto erstellen müssen, bevor die Kamera-App geöffnet und das Foto aufgenommen wird.

```
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();
    }
}
```

Foto machen

Fügen Sie der AndroidManifest-Datei eine Berechtigung zum Zugriff auf die Kamera hinzu:

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

XML-Datei:

```
<?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>
```

Aktivität

```
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;
    }
}
```

So starten Sie die Kamera oder Galerie und speichern das Kameraergebnis im Speicher

Zunächst benötigen Sie `Uri` und temporäre Ordner und Anforderungscodes:

```
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;
```

Dann 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` und `Gallery` Absichten öffnen:

```
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);
    }

}
```

Dann in der `onActivityResult` Methode:

```
@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);
}
```

Sie benötigen diese Berechtigungen in `AndroidManifest.xml` :

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

Und Sie müssen mit Laufzeitberechtigungen wie externem Speicher lesen / schreiben usw. umgehen.

Ich überprüfe die `READ_EXTERNAL_STORAGE` Berechtigung in meiner `openGallery` Methode:

Meine `requestPermission` :

```
protected void requestPermission(final String permission, String rationale, final int
requestCode) {
    if (ActivityCompat.shouldShowRequestPermissionRationale(this, permission)) {
        showAlertDialog(getString(R.string.permission_title_rationale), rationale,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        ActivityCompat.requestPermissions(BasePermissionActivity.this,
                                new String[]{permission}, requestCode);
                    }
                }, getString(android.R.string.ok), null, getString(android.R.string.cancel));
    } else {
        ActivityCompat.requestPermissions(this, new String[]{permission}, requestCode);
    }
}
```

`onRequestPermissionsResult` dann die `onRequestPermissionsResult` Methode:

```
@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` Methode:

```
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();
}
```

Kameraauflösung einstellen

Stellen Sie die hohe Auflösung programmgesteuert ein.

```
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);
```

Decodierungs-Bitmap korrekt gedreht aus dem mit der Absicht abgerufenen URI

```
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;
}
```

Kamera und Galerie online lesen: https://riptutorial.com/de/android/topic/4789/kamera-und-galerie

Kapitel 140: Konten und AccountManager

Examples

Benutzerdefinierte Konten / Authentifizierung

Das folgende Beispiel zeigt eine umfassende Abdeckung der Schlüsselkonzepte und des grundlegenden Skelettaufbaus: -

1. Sammelt Anmeldeinformationen vom Benutzer (normalerweise über einen von Ihnen erstellten Anmeldebildschirm)
2. Authentifiziert die Anmeldeinformationen beim Server (speichert benutzerdefinierte Authentifizierung)
3. Speichert die Anmeldeinformationen auf dem Gerät

Erweitern eines AbstractAccountAuthenticator *(wird hauptsächlich zum Abrufen der Authentifizierung und zum erneuten Authentifizieren verwendet)*

```
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.
  }
}
```

Erstellen Sie einen Dienst *(das Account Manager-Framework stellt über die Dienstschnittstelle eine Verbindung zum erweiterten AbstractAccountAuthenticator her.)*

```
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();
    }
}
```

Authenticator-XML-Konfiguration *(Das Account Manager-Framework erfordert. Dies wird unter Einstellungen -> Konten in Android angezeigt.)*

```
<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" />
```

Änderungen an der AndroidManifest.xml *(Bringen Sie alle oben genannten Konzepte zusammen, um sie programmgesteuert über den AccountManager nutzbar zu machen.)*

```
<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>
```

Das nächste Beispiel enthält Informationen zur Verwendung dieses Setups.

Konten und AccountManager online lesen: https://riptutorial.com/de/android/topic/7003/konten-und-accountmanager

Kapitel 141: Kontext

Einführung

Per Google-Dokumentation: "Schnittstelle zu globalen Informationen über eine Anwendungsumgebung. Sie ermöglicht den Zugriff auf anwendungsspezifische Ressourcen und Klassen sowie Up-Calls für Vorgänge auf Anwendungsebene, z.

Einfacher ausgedrückt: Kontext ist der aktuelle Status Ihrer Anwendung. Sie können damit Informationen zu Objekten bereitstellen, damit diese wissen, was in anderen Teilen Ihrer Anwendung vor sich geht.

Syntax

- `getApplicationContext()`
- `getBaseContext()`
- `getContext()`
- `this`

Bemerkungen

Diese StackOverflow-Seite enthält mehrere ausführliche und gut geschriebene Erklärungen zum Konzept von Kontext:

Was ist der Kontext?

Examples

Grundlegende Beispiele

Standardnutzung in Aktivität:

```
Context context = getApplicationContext();
```

Standardverwendung in Fragment:

```
Context context = getActivity().getApplicationContext();
```

`this` (wenn in einer Klasse, die von Context ausgeht, wie z. B. die Klassen Application, Activity, Service und IntentService)

```
TextView textView = new TextView(this);
```

ein anderes `this` Beispiel:

```
Intent intent = new Intent(this, MainActivity.class);
startActivity(intent);
```

Kontext online lesen: https://riptutorial.com/de/android/topic/9774/kontext

Kapitel 142: Konvertieren Sie den vietnamesischen String in den englischen Android-String

Examples

Beispiel:

```
String myStr = convert("Lê Minh Thoại là người Việt Nam");
```

umgewandelt:

```
"Le Minh Thoai la nguoi Viet Nam"
```

Chuyển chuỗi Tiếng Việt thành chuỗi không dấu

```
  public static String convert(String str) {
    str = str.replaceAll("à|á|ạ|ả|ã|â|ầ|ấ|ậ|ẩ|ẫ|ă|ằ|ắ|ặ|ẳ|ẵ", "a");
    str = str.replaceAll("è|é|ẹ|ẻ|ẽ|ê|ề|ế|ệ|ể|ễ", "e");
    str = str.replaceAll("ì|í|ị|ỉ|ĩ", "i");
    str = str.replaceAll("ò|ó|ọ|ỏ|õ|ô|ồ|ố|ộ|ổ|ỗ|ơ|ờ|ớ|ợ|ở|ỡ", "o");
    str = str.replaceAll("ù|ú|ụ|ủ|ũ|ư|ừ|ứ|ự|ử|ữ", "u");
    str = str.replaceAll("ỳ|ý|ỵ|ỷ|ỹ", "y");
    str = str.replaceAll("đ", "d");

    str = str.replaceAll("À|Á|Ạ|Ả|Ã|Â|Ầ|Ấ|Ậ|Ẩ|Ẫ|Ă|Ằ|Ắ|Ặ|Ẳ|Ẵ", "A");
    str = str.replaceAll("È|É|Ẹ|Ẻ|Ẽ|Ê|Ề|Ế|Ệ|Ể|Ễ", "E");
    str = str.replaceAll("Ì|Í|Ị|Ỉ|Ĩ", "I");
    str = str.replaceAll("Ò|Ó|Ọ|Ỏ|Õ|Ô|Ồ|Ố|Ộ|Ổ|Ỗ|Ơ|Ờ|Ớ|Ợ|Ở|Ỡ", "O");
    str = str.replaceAll("Ù|Ú|Ụ|Ủ|Ũ|Ư|Ừ|Ứ|Ự|Ử|Ữ", "U");
    str = str.replaceAll("Ỳ|Ý|Ỵ|Ỷ|Ỹ", "Y");
    str = str.replaceAll("Đ", "D");
    return str;
}
```

Konvertieren Sie den vietnamesischen String in den englischen Android-String online lesen: https://riptutorial.com/de/android/topic/10946/konvertieren-sie-den-vietnamesischen-string-in-den-englischen-android-string

Kapitel 143: Konvertierung von Sprache in Text

Examples

Sprache zu Text mit Standard-Google-Eingabeaufforderungsdialogfeld

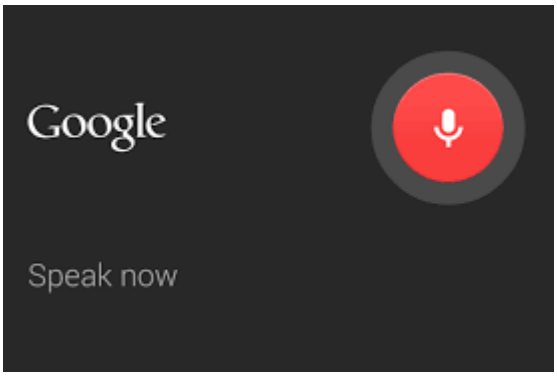
Trigger Rede zur Textübersetzung

```
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();
    }
}
```

Erhalten Sie übersetzte Ergebnisse in onActivityResult

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    switch (requestCode) {
    case REQ_CODE_SPEECH_INPUT: {
        if (resultCode == RESULT_OK && null != data) {

            ArrayList<String> result = data
                    .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
            txtSpeechInput.setText(result.get(0));
        }
        break;
    }

    }
}
```

Ausgabe



Rede zu Text ohne Dialog

Mit dem folgenden Code können Sie eine Sprach-zu-Text-Übersetzung auslösen, ohne einen Dialog anzuzeigen:

```
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);
}
```

Die benutzerdefinierte Listener-Klasse `CustomRecognitionListener` die im obigen Code verwendet wird, wird wie folgt implementiert:

```
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);
    }
}
```

Konvertierung von Sprache in Text online lesen:
https://riptutorial.com/de/android/topic/6252/konvertierung-von-sprache-in-text

Kapitel 144: Lader

Einführung

Loader ist eine gute Wahl, um Speicherverluste zu vermeiden, wenn Sie beim Aufruf der oncreate-Methode Daten in den Hintergrund laden möchten. Wenn wir zum Beispiel Asynctask in der Oncreate-Methode ausführen und den Bildschirm so drehen, dass die Aktivität erneut erstellt wird, wird eine weitere AsyncTask erneut ausgeführt, sodass wahrscheinlich zwei Asyntask parallel laufen und nicht wie ein Loader, der den zuvor ausgeführten Hintergrundprozess fortsetzt.

Parameter

Klasse	Beschreibung
LoaderManager	Eine abstrakte Klasse, die einer Aktivität oder einem Fragment zugeordnet ist, um eine oder mehrere Loader-Instanzen zu verwalten.
LoaderManager.LoaderCallbacks	Eine Rückmeldeschnittstelle, über die ein Client mit dem LoaderManager interagieren kann.
Lader	Eine abstrakte Klasse, die asynchrones Laden von Daten durchführt.
AsyncTaskLoader	Abstraktes Ladeprogramm , das eine AsyncTask für die Arbeit bereitstellt .
CursorLoader	Eine Unterklasse von AsyncTaskLoader, die den ContentResolver abfragt und einen Cursor zurückgibt.

Bemerkungen

In Android 3.0 eingeführt, ermöglichen Lader das asynchrone Laden von Daten in einer Aktivität oder einem Fragment. Lader haben folgende Eigenschaften:

- Sie sind für jede Aktivität und jedes Fragment verfügbar.
- Sie ermöglichen das asynchrone Laden von Daten.
- Sie überwachen die Quelle ihrer Daten und liefern neue Ergebnisse, wenn sich der Inhalt ändert.
- Sie stellen automatisch eine Verbindung zum Cursor des letzten Laders her, wenn sie nach einer Konfigurationsänderung neu erstellt werden. Daher müssen sie ihre Daten nicht erneut abfragen.

Wann sollten Loader nicht verwendet werden

Sie sollten keine Loader verwenden, wenn Sie Hintergrundaufgaben ausführen müssen. Android zerstört Lader zusammen mit den zugehörigen Aktivitäten / Fragmenten. Wenn Sie einige Aufgaben ausführen möchten, die bis zum Abschluss ausgeführt werden müssen, verwenden Sie keine Loader. Sie sollten stattdessen Dienste für diese Art von Sachen verwenden.

Examples

Grundlegender AsyncTaskLoader

`AsyncTaskLoader` ist ein abstrakter `Loader` , der eine `AsyncTask` für die Arbeit `AsyncTask` .

Hier einige grundlegende Implementierungen:

```
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();
    }
}
```

Typischerweise wird `Loader` innerhalb der `onCreate()` Methode der Aktivität oder innerhalb der `onActivityCreated()` des Fragments `onActivityCreated()` . In der Regel implementiert auch activity oder fragment die Schnittstelle `LoaderManager.LoaderCallbacks` :

```
public class MainActivity extends Activity implements LoaderManager.LoaderCallbacks<String> {

    // Unique id for loader
    private static final int LDR_BASIC_ID = 1;

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

        // Initialize loader; Some data can be passed as second param instead of Bundle.Empty
        getLoaderManager().initLoader(LDR_BASIC_ID, Bundle.EMPTY, this);
    }

    @Override
    public Loader<String> onCreateLoader(int id, Bundle args) {
        return new BasicLoader(this);
    }

    @Override
    public void onLoadFinished(Loader<String> loader, String data) {
        Toast.makeText(this, data, Toast.LENGTH_LONG).show();
    }

    @Override
    public void onLoaderReset(Loader<String> loader) {
    }
}
```

In diesem Beispiel wird nach Abschluss des Laders Toast mit Ergebnis angezeigt.

AsyncTaskLoader mit Cache

Es empfiehlt sich, geladene Ergebnisse zwischenzuspeichern, um das mehrfache Laden derselben Daten zu vermeiden.
Um den Cache ungültig zu machen, sollte `onContentChanged()` aufgerufen werden. Wenn der Loader bereits gestartet wurde, wird `forceLoad()` aufgerufen. Andernfalls (wenn der Loader im angehaltenen Zustand ist) kann der Loader die `takeContentChanged()` mit `takeContentChanged()` check nachvollziehen.

Anmerkung: `onContentChanged()` muss vom Haupt-Thread des Prozesses aufgerufen werden.

Javadocs sagt über takeContentChanged ():

> Nehmen Sie das aktuelle Flag, um anzuzeigen, ob sich der Inhalt des Laders geändert hat, während er gestoppt wurde. Wenn ja, wird true zurückgegeben und das Flag wird gelöscht.

```
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
    }
}
```

Neuladen

Um Ihre alten Daten zu `restartLoader()` und den vorhandenen Loader neu zu starten, können Sie die `restartLoader()` Methode verwenden:

```
private void reload() {
    getLoaderManager().reastartLoader(LOADER_ID, Bundle.EMPTY, this);
}
```

Übergeben Sie Parameter mit einem Bundle

Sie können Parameter per Bundle übergeben:

```
Bundle myBundle = new Bundle();
myBundle.putString(MY_KEY, myValue);
```

Rufen Sie den Wert in onCreateLoader ab:

```
@Override
public Loader<String> onCreateLoader(int id, final Bundle args) {
    final String myParam = args.getString(MY_KEY);
    ...
}
```

Lader online lesen: https://riptutorial.com/de/android/topic/4390/lader

Kapitel 145: Laufzeitberechtigungen in API-23 +

Einführung

Android Marshmallow führte das Runtime Permission- Modell ein. Berechtigungen werden in zwei Kategorien unterteilt, nämlich normale und gefährliche Berechtigungen . wo gefährliche Berechtigungen jetzt vom Benutzer zur Laufzeit erteilt werden.

Bemerkungen

Ab sdk 23 benötigt Android Laufzeitberechtigungen für Berechtigungen auf Geräten, auf denen Android 6.0 oder höher ausgeführt wird, innerhalb der als gefährlich eingestuften Berechtigungsgruppen. Gefährliche Berechtigungsgruppen sind Personengruppen, die die Privatsphäre und / oder Sicherheit des Benutzers gefährden.

Im Folgenden finden Sie eine Liste gefährlicher Berechtigungsgruppen:

Gefährliche Berechtigungsgruppen

Berechtigungsgruppe
KALENDER
KAMERA
KONTAKTE
STANDORT
MIKROFON
TELEFON
SENSOREN
SMS
LAGER

Alle Berechtigungen dieser Gruppen erfordern die Verwaltung von Laufzeitberechtigungen für Geräte ab Android 6.0 mit einem Ziel-SDK von 23 oder höher.

Normale Berechtigungen

Im Folgenden finden Sie eine Liste normaler Berechtigungen. Diese werden nicht als gefährlich für die Privatsphäre oder Sicherheit des Benutzers angesehen und erfordern daher keine Laufzeitberechtigungen für SDK 23 und höher.

ACCESS_LOCATION_EXTRA_COMMANDS
ACCESS_NETWORK_STATE
ACCESS_NOTIFICATION_POLICY
ACCESS_WIFI_STATE
BLUETOOTH

BLUETOOTH_ADMIN
BROADCAST_STICKY
CHANGE_NETWORK_STATE
CHANGE_WIFI_MULTICAST_STATE
CHANGE_WIFI_STATE
DISABLE_KEYGUARD
EXPAND_STATUS_BAR
GET_PACKAGE_SIZE
INSTALL_SHORTCUT
INTERNET
KILL_BACKGROUND_PROCESSES
MODIFY_AUDIO_SETTINGS
NFC
READ_SYNC_SETTINGS
READ_SYNC_STATS
RECEIVE_BOOT_COMPLETED
REORDER_TASKS
REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
REQUEST_INSTALL_PACKAGES
WECKER STELLEN
SET_TIME_ZONE
HINTERGRUNDBILD FESTLEGEN
SET_WALLPAPER_HINTS
TRANSMIT_IR
UNINSTALL_SHORTCUT
USE_FINGERPRINT
VIBRIEREN
WAKE_LOCK
WRITE_SYNC_SETTINGS

Examples

Android 6.0 mehrere Berechtigungen

Dieses Beispiel zeigt, wie Berechtigungen zur Laufzeit in Android 6 und höher geprüft werden.

```
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;
        }
    }
}
```

Erzwingen von Berechtigungen in Broadcasts, URI

Sie können eine Berechtigungsprüfung durchführen, wenn Sie eine Absicht an einen registrierten Rundfunkempfänger senden. Die Berechtigungen, die Sie senden, werden mit den unter dem Tag registrierten Berechtigungen abgeglichen. Sie schränken ein, wer Broadcasts an den zugehörigen Empfänger senden darf.

Um eine Broadcast-Anforderung mit Berechtigungen zu senden, geben Sie die Berechtigung als Zeichenfolge im `Context.sendBroadcast(Intent intent, String permission)` jedoch, dass die Empfänger-App diese Berechtigung haben **MUSS** , um Ihre Rundsendung zu empfangen. Der Empfänger sollte zuerst vor dem Absender installiert werden.

Die Methodensignatur lautet:

```
 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");
```

und Sie können in Ihrem Manifest angeben, dass der Broadcast-Sender die angeforderte Berechtigung enthalten muss, die über sendBroadcast gesendet wird:

```
<!--  Your special permission -->
<permission android:name="org.quadcore.mypermission"
   android:label="my_permission"
   android:protectionLevel="dangerous"></permission>
```

Erklären Sie auch die Erlaubnis im Manifest der Anwendung, die diese Sendung empfangen soll:

```
<!--  I use the permission ! -->
<uses-permission android:name="org.quadcore.mypermission"/>
<!-- along with the receiver -->
<receiver android:name="Bcastreceiver" android:exported="true" />
```

Hinweis: Sowohl ein Empfänger als auch ein Sender können eine Berechtigung erfordern. In diesem Fall müssen beide Berechtigungsprüfungen bestanden werden, damit die Absicht an das zugeordnete Ziel übermittelt werden kann. Die App, die die Berechtigung definiert, sollte zuerst installiert werden.

Die vollständige Dokumentation finden Sie hier unter Berechtigungen.

Mehrere Laufzeitberechtigungen aus denselben Berechtigungsgruppen

Im Manifest haben wir vier gefährliche Laufzeitberechtigungen aus zwei Gruppen.

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

In der Aktivität, für die die Berechtigungen erforderlich sind. Beachten Sie, dass bei jeder Aktivität, die Berechtigungen erfordert, nach Berechtigungen gesucht werden muss, da die Berechtigungen widerrufen werden können, während sich die App im Hintergrund befindet und die App dann abstürzt.

```
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();
    }
```

Wir müssen nur für jede dieser Gruppen eine Erlaubnis einholen, und alle anderen Berechtigungen dieser Gruppe werden erteilt, es sei denn, der Benutzer widerruft die Berechtigung.

```
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;
}
```

Hierbei handelt es sich um das Ergebnis, dass der Benutzer Berechtigungen zulässt oder nicht. Wenn in diesem Beispiel keine Berechtigungen zulässig sind, wird die App beendet.

```
@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);
    }
}
```

Weitere Informationen https://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/de

PermissionUtil verwenden

PermissionUtil ist eine einfache und bequeme Methode, um Berechtigungen im Kontext abzufragen. Sie können problemlos `onAllGranted()` , was geschehen soll, wenn alle angeforderten Berechtigungen erteilt wurden (`onAllGranted()`), jede Anforderung abgelehnt wurde (`onAnyDenied()`) oder falls ein rationaler `onRational()` erforderlich ist (`onRational()`).

Überall in Ihrer AppCompatActivity oder Ihrem Fragment, an dem Sie die Erlaubnis des Benutzers anfordern möchten

```
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);
```

Und fügen Sie dies zu `onRequestPermissionsResult`

```
if(mRequestObject!=null){
    mRequestObject.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

Fügen Sie die angeforderte Berechtigung auch zu Ihrer AndroidManifest.xml hinzu

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

Fügen Sie allen berechtigungsbezogenen Code einer abstrakten Basisklasse hinzu und erweitern Sie die Aktivität dieser Basisklasse, um saubereren / wiederverwendbaren Code zu erhalten

```
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);
    }
}
```

Verwendungsbeispiel in der Aktivität

Die Aktivität sollte die oben definierte abstrakte Basisklasse wie folgt erweitern:

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

Laufzeitberechtigungen in API-23 + online lesen:
https://riptutorial.com/de/android/topic/1525/laufzeitberechtigungen-in-api-23-plus

Kapitel 146: Layouts

Einführung

Ein Layout definiert die visuelle Struktur für eine Benutzeroberfläche, z. B. eine Aktivität oder ein Widget.

Ein Layout wird in XML deklariert, einschließlich der darin enthaltenen Bildschirmelemente. Der Anwendung kann Code hinzugefügt werden, um den Status der Bildschirmobjekte zur Laufzeit zu ändern, einschließlich der in XML deklarierten.

Syntax

- android: gravity = "oben | unten | links | rechts | center_vertical | fill_vertical | center_horizontal | fill_horizontal | center | fill | clip_vertical | clip_horizontal | start | end"
- android: layout_gravity = "oben | unten | links | rechts | center_vertical | fill_vertical | center_horizontal | fill_horizontal | center | fill | clip_vertical | clip_horizontal | start | end"

Bemerkungen

LayoutParams- und Layout_-Attribute

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

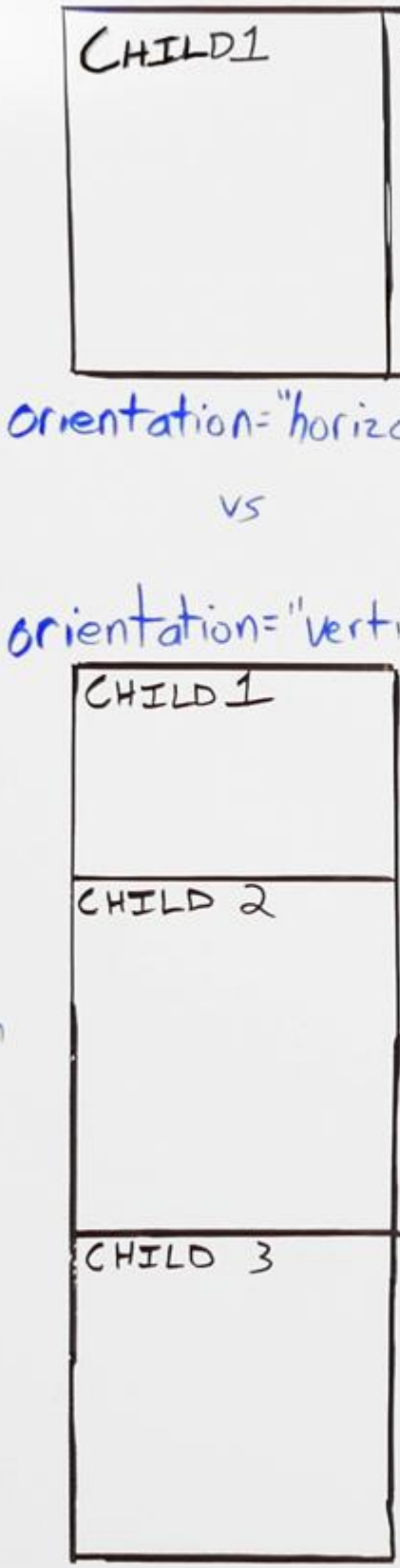
layout_alignBaseline

layout_centerInParent

+AbsoluteLayout NO

Please don't

Linear



Layout zwei Layout-Durchgänge erforderlich, damit das Rendering ordnungsgemäß ausgeführt werden kann. Bei komplexen Ansichtshierarchien kann dies erhebliche Auswirkungen auf die Leistung haben. Durch das Verschachteln von `RelativeLayouts` wird dieses Problem noch verschlimmert, da jedes `RelativeLayout` die Anzahl der Layoutdurchläufe erhöht.

Examples

LinearLayout

Das LinearLayout ist eine `ViewGroup` , die ihre untergeordneten `ViewGroup` in einer einzelnen Spalte oder einer einzelnen Zeile `ViewGroup` . Die Ausrichtung kann durch Aufrufen der Methode `setOrientation()` oder mithilfe des XML-Attributs `android:orientation` .

1. **Vertikale Ausrichtung** : `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>
```

Hier ist ein Screenshot, wie das aussehen wird:

2. **Horizontale Ausrichtung** : `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" />
```

Das `LinearLayout` unterstützt auch das Zuweisen einer Gewichtung für einzelne Kinder mit dem Attribut `android:layout_weight` .

RelativesLayout

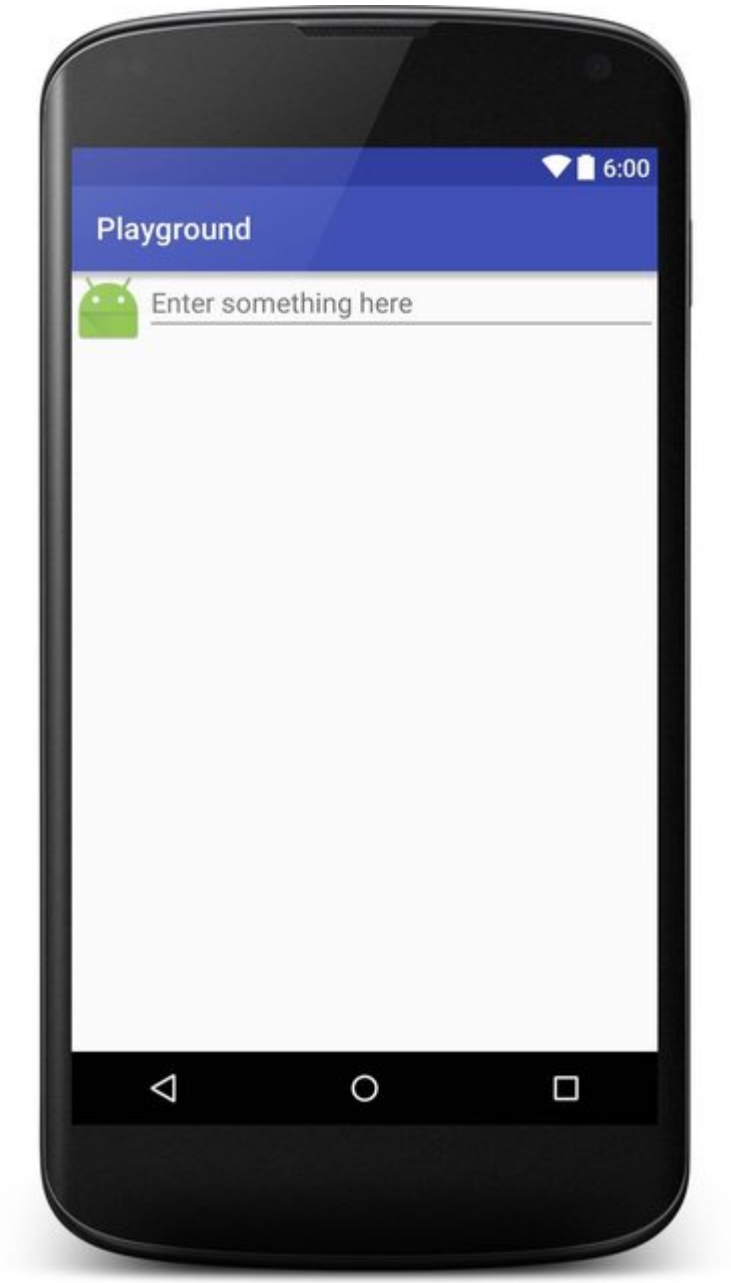
`RelativeLayout` ist eine `ViewGroup` , die `ViewGroup` Ansichten in relativen Positionen anzeigt.

Standardmäßig werden alle untergeordneten Ansichten oben links im Layout gezeichnet. Sie müssen daher die Position jeder Ansicht unter Verwendung der verschiedenen Layout-Eigenschaften definieren, die in `RelativeLayout.LayoutParams` verfügbar sind. Der Wert für jede Layout-Eigenschaft ist entweder ein Boolescher Wert, um eine Layoutposition relativ zum übergeordneten RelativeLayout zu aktivieren, oder eine ID, die auf eine andere Ansicht im Layout verweist, gegenüber der die Ansicht positioniert werden soll.

Beispiel:

```
<?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>
```

Hier ist ein Screenshot, wie das aussehen wird:



Schwerkraft und Layout Schwerkraft

android: layout_gravity

- `android:layout_gravity` wird verwendet, um die Position eines Elements in seinem übergeordneten Element `android:layout_gravity` (z. B. eine `android:layout_gravity` `View` in einem `Layout`).
- Unterstützt von LinearLayout und FrameLayout

android: schwerkraft

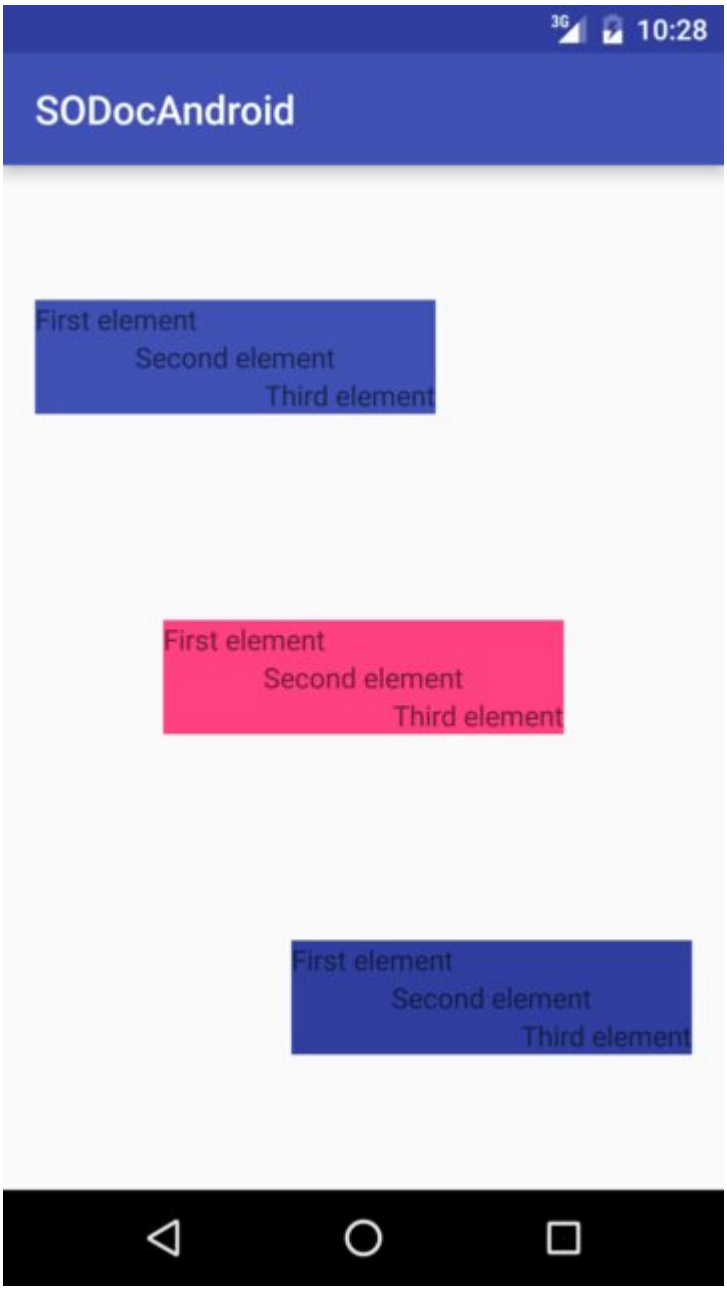
- `android:gravity` wird die Position des Inhalts innerhalb eines Elements (z. B. eines Textes in einer `TextView`) festgelegt.

```
<?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>
```

Was wird wie folgt gerendert:

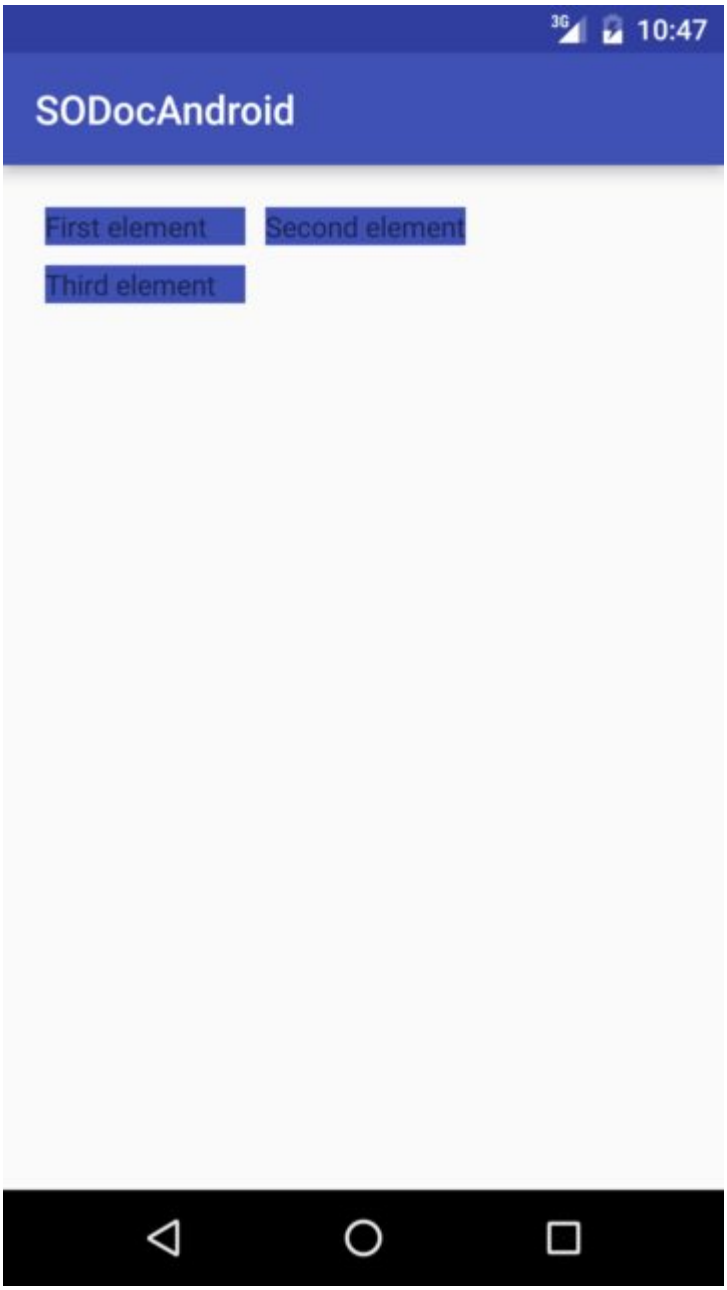


Gitterstruktur

GridLayout ist, wie der Name schon sagt, ein Layout zum Anordnen von Ansichten in einem Raster. Ein GridLayout teilt sich in Spalten und Zeilen auf. Wie Sie im folgenden Beispiel sehen können, wird die Anzahl der Spalten und / oder Zeilen durch die Eigenschaften `columnCount` und `rowCount` . Durch das Hinzufügen von Ansichten zu diesem Layout wird die erste Ansicht der ersten Spalte, die zweite Ansicht der zweiten Spalte und die dritte Ansicht der ersten Spalte der zweiten Zeile hinzugefügt.

```
<?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>
```



Prozentuale Layouts

2.3

Die Percent Support Library bietet `PercentFrameLayout` und `PercentRelativeLayout` , zwei Ansichtsgruppen, die eine einfache Möglichkeit `PercentRelativeLayout` , Ansichtsdimensionen **und -ränder** als **Prozentsatz** der Gesamtgröße anzugeben.

Sie können die Percent Support Library verwenden, indem Sie Folgendes zu Ihren Abhängigkeiten hinzufügen.

```
compile 'com.android.support:percent:25.3.1'
```

Wenn Sie eine Ansicht anzeigen möchten, die den Bildschirm horizontal, aber nur die Hälfte des Bildschirms vertikal ausfüllt, würden Sie Folgendes tun.

```
<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>
```

Sie können die Prozentsätze auch in einer separaten XML-Datei mit folgendem Code definieren:

```
<fraction name="margin_start_percent">25%</fraction>
```

Und verweisen Sie sie in Ihren Layouts mit `@fraction/margin_start_percent` .

Sie enthalten auch die Möglichkeit, ein benutzerdefiniertes **Seitenverhältnis** über die `app:layout_aspectRatio` .
Auf diese Weise können Sie nur eine einzige Bemaßung festlegen, z. B. nur die Breite. Die Höhe wird automatisch basierend auf dem von Ihnen definierten Seitenverhältnis bestimmt, ob es 4: 3 oder 16: 9 oder sogar ein Quadrat 1: 1 ist Seitenverhältnis.

Zum Beispiel:

```
 <ImageView
    app:layout_widthPercent="100%"
    app:layout_aspectRatio="178%"
    android:scaleType="centerCrop"
    android:src="@drawable/header_background"/>
```

FrameLayout

`FrameLayout` dient dazu, einen Bereich auf dem Bildschirm zu sperren, um ein einzelnes Element anzuzeigen. Sie können jedoch einem FrameLayout mehrere untergeordnete Objekte hinzufügen und deren Position innerhalb des FrameLayout steuern, indem Sie jedem untergeordneten Element die Schwerkraft zuweisen, indem Sie das Attribut android: layout_gravity verwenden .

Im Allgemeinen wird `FrameLayout` verwendet, um eine einzelne `FrameLayout` Ansicht zu `FrameLayout` . Übliche Anwendungsfälle sind das Erstellen von Platzhaltern für das Aufblasen von `Fragments` in der `Activity` , das Überlappen von Ansichten oder das Anwenden des Vordergrunds auf die Ansichten.

Beispiel:

```
<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>
```

Es wird so aussehen:

KoordinatorLayout

2.3

Das `CoordinatorLayout` ist ein Container, der `FrameLayout` etwas ähnelt, jedoch mit zusätzlichen

Funktionen in der offiziellen Dokumentation als `FrameLayout` mit Super-Power `FrameLayout` .

Durch Anhängen eines `CoordinatorLayout.Behavior` an ein direktes untergeordnetes Element von CoordinatorLayout können Sie Berührungsereignisse, Fensterinsets, Messungen, Layout und verschachteltes Scrollen abfangen.

Um sie verwenden zu können, müssen Sie zunächst eine Abhängigkeit für die Unterstützungsbibliothek in Ihrer Gradle-Datei hinzufügen:

```
compile 'com.android.support:design:25.3.1'
```

Die Nummer der neuesten Version der Bibliothek finden Sie hier

Ein praktischer Anwendungsfall des `CoordinatorLayout` ist das Erstellen einer Ansicht mit einem `FloatingActionButton` . In diesem speziellen Fall erstellen wir eine `RecyclerView` mit einem `SwipeRefreshLayout` und einem `FloatingActionButton` darüber. So können Sie das machen:

```
<?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>
```

Beachten Sie, wie der FloatingActionButton mit der `app:layout_anchor="@id/coord_layout"` im CoordinatorLayout verankert wird `app:layout_anchor="@id/coord_layout"`

CoordinatorLayout-Scrolling-Verhalten

2.3-2.3.2

Ein einschließendes `CoordinatorLayout` kann verwendet werden, um Material Design-Scrolling-Effekte zu erzielen, wenn innere Layouts verwendet werden, die Nested Scrolling unterstützen, beispielsweise `NestedScrollView` oder `RecyclerView` .

Für dieses Beispiel:

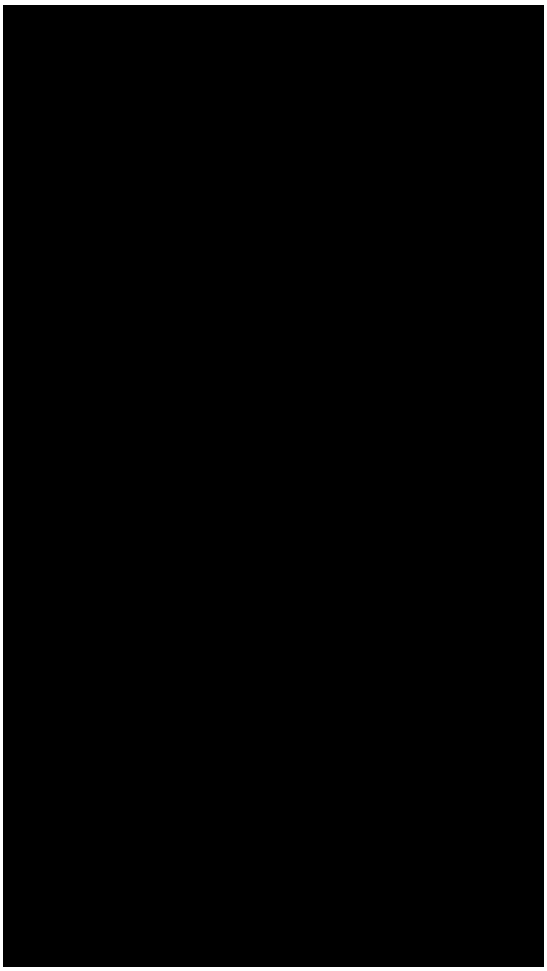
- `app:layout_scrollFlags="scroll|enterAlways"` wird in den Eigenschaften der `app:layout_scrollFlags="scroll|enterAlways"` verwendet
- `app:layout_behavior="@string/appbar_scrolling_view_behavior"` wird in den ViewPager-Eigenschaften verwendet
- In den ViewPager-Fragmenten wird eine RecyclerView verwendet

Hier ist die Layout-XML-Datei, die in einer Aktivität verwendet wird:

```
<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>
```

Ergebnis:



Gewicht anzeigen

Eines der am häufigsten verwendeten Attribute für LinearLayout ist die Gewichtung der untergeordneten Ansichten. Die Gewichtung legt fest, wie viel Platz eine Ansicht im Vergleich zu anderen Ansichten in einem LinearLayout verbraucht.

Die Gewichtung wird verwendet, wenn Sie einer Komponente im Vergleich zu anderen Komponenten einen bestimmten Bildschirmbereich zuweisen möchten.

Schlüsseleigenschaften :

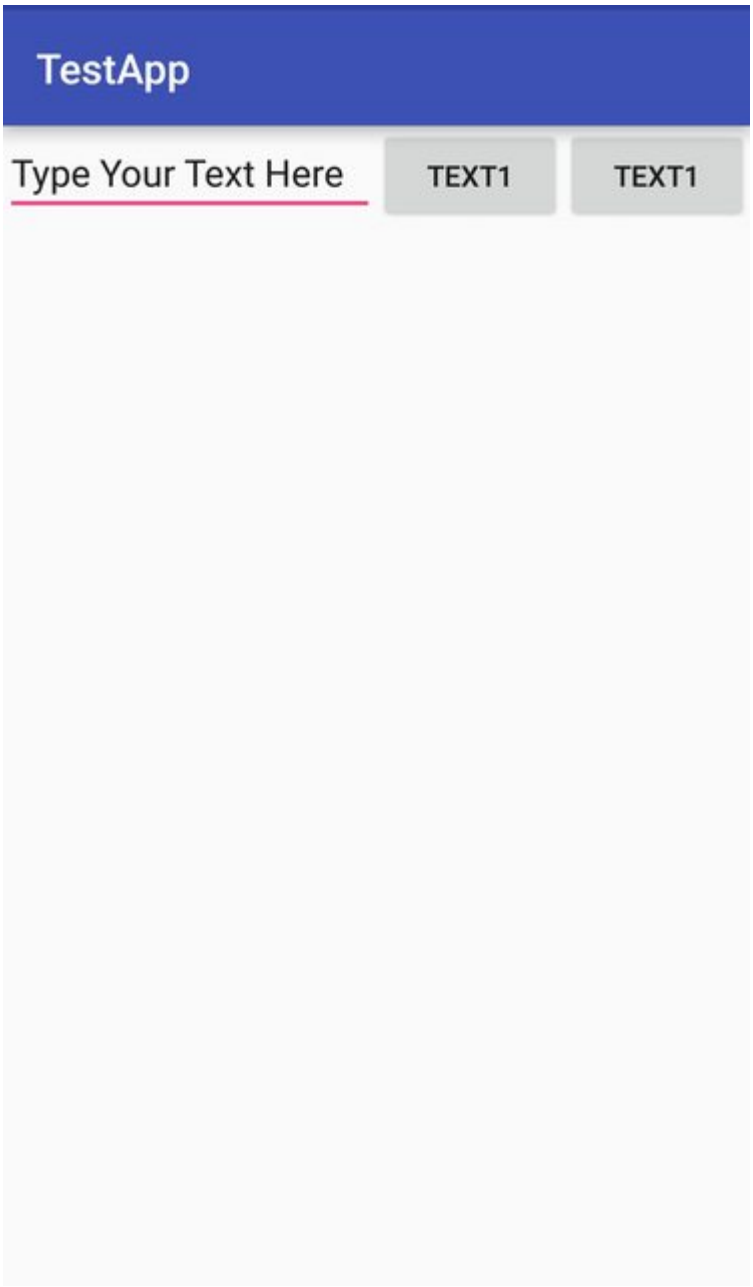
- `weightSum` ist die Gesamtsumme der Gewichtungen aller untergeordneten Ansichten. Wenn Sie die `weightSum` nicht angeben, berechnet das System die Summe aller Gewichtungen alleine.

- `layout_weight` gibt die Menge an Speicherplatz an, die das Widget insgesamt `layout_weight` .

Code:

```
<?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>
```

Die Ausgabe ist:



Auch wenn die Größe des Geräts größer ist, nimmt der EditText 2/4 der Bildschirmfläche ein. Daher ist das Aussehen Ihrer App auf allen Bildschirmen konsistent.

Hinweis: Hier wird die `layout_width` auf `0dp` `layout_width` gehalten, da der `0dp` horizontal geteilt wird. Wenn die Widgets vertikal `layout_height` werden sollen, wird `0dp` auf `0dp` . Dies geschieht, um die Effizienz des Codes zu erhöhen, da das System zur Laufzeit nicht versucht, die Breite bzw. Höhe zu berechnen, da dies vom Gewicht verwaltet wird. Wenn Sie stattdessen `wrap_content` verwenden, `wrap_content` das System zuerst, die Breite / Höhe zu berechnen, bevor Sie das Gewichtsattribut anwenden, wodurch ein weiterer Berechnungszyklus ausgelöst wird.

LinearLayout programmgesteuert erstellen

Hierarchie

```
- LinearLayout(horizontal)
   - ImageView
```

```
  - LinearLayout(vertical)
    - TextView
    - TextView
```

Code

```
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

Jede einzelne `ViewGroup` (z. B. `LinearLayout` , `RelativeLayout` , `CoordinatorLayout` usw.) muss Informationen über die Eigenschaften ihrer Kinder speichern. `ViewGroup` die `ViewGroup` in der `ViewGroup` . Diese Informationen werden in Objekten der Wrapper-Klasse `ViewGroup.LayoutParams` .

`ViewGroups` verwendet Unterklassen der `ViewGroup.LayoutParams` Klasse, um spezifische Parameter für einen bestimmten `ViewGroups` `ViewGroup.LayoutParams` .

ZB für

- `LinearLayout` es `LinearLayout.LayoutParams`
- `RelativeLayout` es `RelativeLayout.LayoutParams`
- `CoordinatorLayout` Es ist `CoordinatorLayout.LayoutParams`
- ...

Die meisten `ViewGroups` die Möglichkeit, `margins` für ihre Kinder `ViewGroup.LayoutParams` , sodass sie `ViewGroup.LayoutParams` direkt subclassieren, sondern `ViewGroup.MarginLayoutParams` (die selbst eine Subklasse von `ViewGroup.LayoutParams`).

`LayoutParams` in XML

`LayoutParams` Objekte werden basierend auf der aufgeblasenen Layout- `xml` Datei erstellt.

```
<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>
```

Alle Parameter, die mit `layout_` beginnen, `layout_` , wie das **umschließende** Layout funktionieren soll. Wenn das Layout aufgeblasen wird, werden diese Parameter in ein richtiges `LayoutParams` Objekt eingeschlossen, das später vom `Layout` verwendet wird, um eine bestimmte `View` in der `ViewGroup` richtig zu positionieren. Andere Attribute einer `View` sind direkt auf die `View` und werden von der `View` selbst verarbeitet.

Für `TextView` :

- `layout_width` , `layout_height` und `layout_gravity` werden in einem `LinearLayout.LayoutParams` Objekt gespeichert und vom `LinearLayout`
- `gravity` , `text` und `textColor` werden vom `TextView` selbst verwendet

Für `ImageView` :

- `layout_width` , `layout_height` und `layout_weight` werden in einem `LinearLayout.LayoutParams` Objekt gespeichert und vom `LinearLayout`
- `background` , `scaleType` und `src` werden von `ImageView` selbst verwendet

`LayoutParams` Objekt `LayoutParams`

`getLayoutParams` ist eine `View's` Methode, mit der ein aktuelles `LayoutParams` Objekt abgerufen werden kann.

Da das `LayoutParams` Objekt direkt mit dem **umschließenden** zusammenhängt `ViewGroup` , gibt diese Methode einen Wert ungleich Null nur dann , wenn `View` auf das angebracht ist `ViewGroup` . Sie müssen bedenken, dass dieses Objekt möglicherweise nicht immer vorhanden ist. Insbesondere sollten Sie nicht darauf angewiesen sein, dass es sich innerhalb `View's` Konstruktors von `View's` .

```
public class ExampleView extends View {

    public ExampleView(Context context) {
        super(context);
        setupView(context);
    }

    public ExampleView(Context context, AttributeSet attrs) {
        super(context, attrs);
        setupView(context);
    }

    public ExampleView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        setupView(context);
    }

    private void setupView(Context context) {
        if (getLayoutParams().height == 50){  // DO NOT DO THIS!
                                              // This might produce NullPointerException
            doSomething();
        }
    }

    //...
}
```

Wenn Sie sich auf das `LayoutParams` Objekt verlassen `LayoutParams` , sollten `onAttachedToWindow` stattdessen die Methode `onAttachedToWindow` verwenden.

```
public class ExampleView extends View {

    public ExampleView(Context context) {
        super(context);
    }

    public ExampleView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public ExampleView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    @Override
    protected void onAttachedToWindow() {
        super.onAttachedToWindow();
        if (getLayoutParams().height == 50) { // getLayoutParams() will NOT return null here
            doSomething();
        }
    }

    //...
}
```

Casting des `LayoutParams` Objekts

Möglicherweise müssen Sie Funktionen verwenden, die für eine bestimmte `ViewGroup` spezifisch

sind (z. B. möchten Sie die Regeln eines `RelativeLayout` programmgesteuert ändern). Zu diesem Zweck müssen Sie wissen, wie das `ViewGroup.LayoutParams` Objekt ordnungsgemäß `ViewGroup.LayoutParams` wird.

Dies kann etwas verwirrend sein, wenn ein `LayoutParams` Objekt für eine `LayoutParams View ViewGroup` , bei der es sich tatsächlich um eine andere `ViewGroup` .

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/outer_layout"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical">

    <FrameLayout
        android:id="@+id/inner_layout"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_gravity="right"/>

</LinearLayout>
```

WICHTIG: Der Typ des `LayoutParams` Objekts hängt direkt mit dem Typ der **ENCLOSING-** `ViewGroup` .

***Falsches** Casting* :

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
FrameLayout.LayoutParams par = (FrameLayout.LayoutParams) innerLayout.getLayoutParams();
                                        // INCORRECT! This will produce ClassCastException
```

***Korrektes** Gießen* :

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
LinearLayout.LayoutParams par = (LinearLayout.LayoutParams) innerLayout.getLayoutParams();
                                       // CORRECT! the enclosing layout is a LinearLayout
```

Layouts online lesen: https://riptutorial.com/de/android/topic/94/layouts

Kapitel 147: Leakcanary

Einführung

Leak Canary ist eine Android- und Java-Bibliothek, die zur Erkennung von Leckagen in der Anwendung verwendet wird

Bemerkungen

Sie können das Beispiel im Link unten sehen

https://github.com/square/leakcanary

Examples

Ein Leck Canary in Android-Anwendung implementieren

In Ihrem *build.gradle* müssen Sie die folgenden Abhängigkeiten hinzufügen:

```
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'
```

In Ihrer `Application` Klasse müssen Sie den folgenden Code in `onCreate()` :

```
LeakCanary.install(this);
```

Das ist alles, was Sie für *LeakCanary* tun *müssen* . Es zeigt automatisch Benachrichtigungen an, wenn ein Fehler in Ihrem Build *auftritt* .

Leakcanary online lesen: https://riptutorial.com/de/android/topic/10041/leakcanary

Kapitel 148: Leinwandzeichnung mit SurfaceView

Bemerkungen

Es ist wichtig, das Grundkonzept der Oberflächenansicht zu verstehen, bevor Sie sie verwenden:

- Es ist im Grunde nur ein Loch im aktuellen Fenster
- Native UI kann darüber platziert werden
- Das Zeichnen erfolgt mit einem dedizierten Thread ohne Benutzeroberfläche
- Das Zeichnen ist nicht hardwarebeschleunigt
- Verwendet zwei Puffer: Einer wird aktuell angezeigt, einer wird zum Zeichnen verwendet.
- `unlockCanvasAndPost()` tauscht die Puffer aus.

Deadlocks können leicht auftreten, wenn die `lockCanvas()` und `unlockCanvasAndPost()` nicht in der richtigen Reihenfolge aufgerufen werden.

Examples

SurfaceView mit Zeichnungsfaden

In diesem Beispiel wird beschrieben, wie Sie eine SurfaceView mit einem dedizierten Zeichnungsthread erstellen. Diese Implementierung behandelt auch Randfälle wie fertigungsspezifische Probleme sowie das Starten / Stoppen des Threads, um CPU-Zeit zu sparen.

```
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");
    }
}
```

Dieses Layout enthält nur die benutzerdefinierte SurfaceView und maximiert sie auf die Bildschirmgröße.

```
<?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>
```

Die Aktivität, die SurfaceView verwendet, ist für das Starten und Stoppen des Zeichnungsthreads verantwortlich. Dieser Ansatz spart Batterie, da die Zeichnung angehalten wird, sobald sich die Aktivität im Hintergrund befindet.

```
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();
    }
}
```

Leinwandzeichnung mit SurfaceView online lesen:
https://riptutorial.com/de/android/topic/3754/leinwandzeichnung-mit-surfaceview

Kapitel 149: Leistungsoptimierung

Einführung

Die Leistung Ihrer Apps ist ein entscheidendes Element der Benutzererfahrung. Vermeiden Sie schlechte Performance-Muster, z. B. Arbeiten am UI-Thread, und lernen Sie, schnell und reaktionsschnelle Apps zu schreiben.

Examples

Speichern Sie View-Lookups mit dem ViewHolder-Muster

Insbesondere in einer `ListView` können Sie auf Leistungsprobleme `ListView` , wenn Sie zu viele `findViewById()` Aufrufe während des Bildlaufs `findViewById()` . Mit dem `ViewHolder` Muster können Sie diese `ListView` speichern und die `ListView` Leistung verbessern.

Wenn Ihr Listenelement ein einzelnes `TextView` enthält, erstellen Sie eine `ViewHolder` Klasse, um die Instanz zu speichern:

```
static class ViewHolder {
    TextView myTextView;
}
```

`ViewHolder` Erstellen Ihres Listenelements ein `ViewHolder` Objekt an das Listenelement an:

```
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;
}
```

Wenn Sie dieses Muster verwenden, wird `findViewById()` nur aufgerufen, wenn eine neue `View` erstellt wird und die `ListView` Ihre Ansichten viel effizienter recyceln kann.

Leistungsoptimierung online lesen:
https://riptutorial.com/de/android/topic/8711/leistungsoptimierung

Kapitel 150: Listenansicht

Einführung

ListView ist eine Ansichtsgruppe, die mehrere Elemente aus einer Datenquelle wie einem Array oder einer Datenbank zusammenfasst und in einer scrollbaren Liste anzeigt. Daten werden mithilfe einer Adapterklasse an die Listendarstellung gebunden.

Bemerkungen

`ListView` ist eine Ansichtsgruppe, in der eine Liste scrollbarer Elemente `ListView` wird.
Die Listenelemente werden automatisch mit einem `Adapter` in die Liste eingefügt, der Inhalt aus einer Quelle wie einem Array oder einer Datenbankabfrage abruft und jedes Elementergebnis in eine Ansicht konvertiert, die in der Liste enthalten ist.

Wenn der Inhalt Ihres Layouts dynamisch oder nicht vordefiniert ist, können Sie ein Layout verwenden, das die `AdapterView` Unterklassen `AdapterView` , um das Layout zur Laufzeit mit Ansichten zu `AdapterView` . Eine Unterklasse der `AdapterView` Klasse verwendet einen `Adapter` , um Daten an das Layout zu binden.

Bevor Sie die `ListView` , sollten Sie auch die `RecyclerView` Beispiele überprüfen.

Examples

Filtern mit CursorAdapter

```
// Get the reference to your ListView
ListView listResults = (ListView) findViewById(R.id.listResults);

// Set its adapter
listResults.setAdapter(adapter);

// Enable filtering in ListView
listResults.setTextFilterEnabled(true);

// Prepare your adapter for filtering
adapter.setFilterQueryProvider(new FilterQueryProvider() {
    @Override
    public Cursor runQuery(CharSequence constraint) {

        // in real life, do something more secure than concatenation
        // but it will depend on your schema
        // This is the query that will run on filtering
        String query = "SELECT _ID as _id, name FROM MYTABLE "
                       + "where name like '%" + constraint + "%' "
                       + "ORDER BY NAME ASC";
        return db.rawQuery(query, null);
     }
});
```

`EditText` , Ihre Abfrage wird jedes Mal ausgeführt, wenn der Benutzer einen `EditText` :

```
    EditText queryText = (EditText) findViewById(R.id.textQuery);
    queryText.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(final CharSequence s, final int start, final int count,
final int after) {

        }

        @Override
        public void onTextChanged(final CharSequence s, final int start, final int before,
final int count) {
            // This is the filter in action
            adapter.getFilter().filter(s.toString());
            // Don't forget to notify the adapter
            adapter.notifyDataSetChanged();
        }

        @Override
        public void afterTextChanged(final Editable s) {

        }
    });
```

Benutzerdefinierter ArrayAdapter

Standardmäßig erstellt die ArrayAdapter-Klasse eine Ansicht für jedes Arrayelement, indem Sie für jedes Element `toString()` aufrufen und den Inhalt in eine TextView einfügen.

Um eine komplexe Ansicht für jedes Element zu erstellen (z. B. wenn Sie für jedes Array-Element eine ImageView erstellen möchten), erweitern Sie die ArrayAdapter-Klasse, und überschreiben Sie die `getView()` Methode, um den für jedes Element `getView()` zurückzugeben.

Zum Beispiel:

```
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);
         }
    }
}
```

Eine grundlegende ListView mit einem ArrayAdapter

Standardmäßig erstellt der `ArrayAdapter` eine Ansicht für jedes `ArrayAdapter` indem Sie für jedes Element `toString()` aufrufen und den Inhalt in eine `TextView` .

Beispiel:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, myStringArray);
```

Dabei ist `android.R.layout.simple_list_item_1` das Layout, das eine `TextView` für jeden String im Array enthält.

Dann rufen `setAdapter()` einfach `setAdapter()` in Ihrer `ListView` :

```
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

Wenn Sie für die Array-Anzeige etwas anderes als TextViews verwenden möchten, z. B.

ImageViews, oder außer `toString()` Ergebnisse mit Daten füllen, überschreiben Sie `getView(int, View, ViewGroup)` , um den gewünschten Ansichtstyp zurückzugeben. Überprüfen Sie dieses Beispiel .

Listenansicht online lesen: https://riptutorial.com/de/android/topic/4226/listenansicht

Kapitel 151: Lokalisiertes Datum / Uhrzeit in Android

Bemerkungen

Es wird empfohlen, Methoden der DateUtils- Klasse zu verwenden, um Datumsangaben zu formatieren, die sich auf das Gebietsschema beziehen, dh die Benutzervorlieben berücksichtigen (z. B. Uhrzeitformate für 12h / 24h). Diese Methoden eignen sich am besten für Datumsangaben, die dem Benutzer angezeigt werden.

Für vollständig angepasste Datumsdarstellungen wird die Verwendung der SimpleDateFormat- Klasse empfohlen, da damit alle Datumselemente vollständig gesteuert werden können.

Examples

Benutzerdefiniertes lokalisiertes Datumsformat mit DateUtils.formatDateTime ()

Mit DateUtils.formatDateTime () können Sie eine Uhrzeit angeben. Basierend auf den von Ihnen bereitgestellten Flags wird eine lokalisierte Datetime-Zeichenfolge erstellt. Mit den Flags können Sie angeben, ob bestimmte Elemente (wie der Wochentag) eingeschlossen werden sollen.

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

formatDateTime () kümmert sich automatisch um die richtigen Datumsformate.

Standardformat für Datum und Uhrzeit in Android

Datum formatieren:

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

Datum und Uhrzeit formatieren Datum ist im Kurzformat, Zeit ist im Langformat:

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

Datum und Uhrzeit vollständig angepasst

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

Häufig verwendete Muster:

- HH: Stunde (0-23)
- hh: stunde (1-12)
- a: AM / PM-Markierung
- mm: Minute (0-59)
- ss: zweitens
- Tag: Tag im Monat (1-31)
- MM: Monat
- JJJJ: Jahr

Lokalisiertes Datum / Uhrzeit in Android online lesen:
https://riptutorial.com/de/android/topic/6057/lokalisiertes-datum---uhrzeit-in-android

Kapitel 152: Lokalisierung mit Ressourcen in Android

Examples

Währung

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

Hinzufügen von Übersetzungen zu Ihrer Android-App

Sie müssen für jede neue Sprache eine andere Datei " `strings.xml` erstellen.

1. Klicken Sie mit der rechten Maustaste auf den Ordner *res*
2. Wählen Sie *Neu* → *Ressourcendatei Werte*
3. Wählen Sie aus den verfügbaren Qualifikationsmerkmalen ein Gebietsschema aus
4. Klicken Sie auf die Schaltfläche *Weiter* (>>)
5. Wähle eine Sprache
6. Benennen Sie die Datei *strings.xml*

strings.xml

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

strings.xml (hi)

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

Sprache programmgesteuert einstellen:

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

Die obige Funktion ändert die Textfelder, auf die in *strings.xml* verwiesen wird. Angenommen, Sie haben die folgenden zwei Textansichten:

```
<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"/>
```

Nach dem Ändern des Gebietsschemas werden die Sprachzeichenfolgen mit den IDs `app_name` und `hello` entsprechend geändert.

Typ der Ressourcenverzeichnisse unter dem Ordner "res"

Bei der Lokalisierung sind verschiedene Arten von Ressourcen erforderlich, von denen jede in der Android-Projektstruktur eine eigene Heimat hat. Nachfolgend sind die verschiedenen Verzeichnisse aufgeführt, die wir im Verzeichnis `\res` ablegen können. Die in jedem dieser Verzeichnisse platzierten Ressourcentypen werden in der folgenden Tabelle erläutert:

Verzeichnis	Ressourcentyp
Animator/	XML-Dateien, die Eigenschaftsanimationen definieren.
anim /	XML-Dateien, die Tween-Animationen definieren. (Eigenschaftsanimationen können auch in diesem Verzeichnis gespeichert werden. Das Animator / -Verzeichnis wird jedoch für Eigenschaftsanimationen bevorzugt, um zwischen den beiden Typen zu unterscheiden.)
Farbe/	XML-Dateien, die eine Zustandsliste von Farben definieren. Siehe Farbzustandslistenressource
zeichnend /	„Bitmap - Dateien (.png, .9.png, JPG, GIF) oder XML - Dateien , die in die folgenden ziehbar Ressource Subtypen kompiliert werden: `Bitmap files – Nine-Patches (re-sizable bitmaps) – State lists – Shapes – Animation drawables – Other drawables – "` - `Bitmap files – Nine-Patches (re-sizable bitmaps) – State lists – Shapes – Animation drawables – Other drawables – "`

Verzeichnis	Ressourcentyp
Mipmap /	Zeichnungsfähige Dateien für unterschiedliche Symbol-Startsymbole. Weitere Informationen zum Verwalten von Startersymbolen mit Mipmap / Ordnern finden Sie unter Verwalten von Projekten - Übersicht.
Layout/	XML-Dateien, die ein Benutzeroberflächenlayout definieren. Siehe Layoutressource.
Speisekarte/	XML-Dateien, die Anwendungsmenüs definieren, z. B. ein Optionsmenü, ein Kontextmenü oder ein Untermenü. Siehe Menüressource.
roh/	Beliebige Dateien, die in ihrer Rohform gespeichert werden sollen. Um diese Ressourcen mit einem rohen InputStream zu öffnen, rufen Sie Resources.openRawResource () mit der Ressourcen-ID auf. Dies ist R.raw.filename.
	Wenn Sie jedoch auf die ursprünglichen Dateinamen und die Dateiehierarchie zugreifen müssen, sollten Sie möglicherweise einige Ressourcen im Verzeichnis assets / speichern (statt ofres / raw /). Dateien in Assets / erhalten keine Ressourcen-ID. Sie können sie daher nur mit AssetManager lesen.
Werte/	XML-Dateien, die einfache Werte enthalten, z. B. Zeichenfolgen, Ganzzahlen und Farben sowie Stile und Designs
xml /	Beliebige XML-Dateien, die zur Laufzeit durch Aufrufen von Resources.getXML () gelesen werden können. Hier müssen verschiedene XML-Konfigurationsdateien gespeichert werden, beispielsweise eine durchsuchbare Konfiguration.

Konfigurationstypen und Qualifikationsnamen für jeden Ordner im Verzeichnis "res"

Für jedes Ressourcenverzeichnis unter dem Ordner `res` (in dem obigen Beispiel aufgeführt) können verschiedene Variationen der enthaltenen Ressourcen in einem gleichnamigen Verzeichnis mit Suffix und unterschiedlichen `qualifier-values` für jeden `configuration-type` .

Beispiel für Variationen von `` Verzeichnis mit unterschiedlichen Qualifier-Werten, die häufig in unseren Android-Projekten angezeigt werden:

- zeichnend /
- drawable-de /
- Zeichenbar-fr-rCA /
- drawable-de-port /
- drawable-en-notouch-12key /
- drawable-port-ldpi /
- drawable-port-notouch-12key /

Vollständige Liste aller verschiedenen Konfigurationstypen und ihrer Qualifiziererwerte für Android-Ressourcen:

Aufbau	Qualifikationswerte
MCC und MNC	Beispiele:
	mcc310
	mcc310-mnc004
	mcc208-mnc00
	usw.
Sprache und Region	Beispiele:
	en
	fr
	en-rUS
	fr-rFR
	fr-rCA
Layoutrichtung	ldrtl
	ldltr
kleinsteWeite	swdp
	Beispiele:
	sw320dp
	sw600dp
	sw720dp
Verfügbare Breite	wdp
	w720dp
	w1024dp
Verfügbare Höhe	HDP
	h720dp

Aufbau	Qualifikationswerte
	h1024dp
Bildschirmgröße	klein
	normal
	groß
	xgroß
Bildschirmaspekt	lange
	nicht lange
Runder Bildschirm	runden
	nicht rund
Bildschirmausrichtung	Hafen
	Land
UI-Modus	Auto
	Schreibtisch
	Fernsehen
	Geräteuhr
Nacht-Modus	Nacht-
	nicht Nacht
Bildschirmpixeldichte (dpi)	ldpi
	mdpi
	hdpi
	xhdpi
	xxhdpi
	xxxhdpi
	nodpi
	tvdpi

Aufbau	Qualifikationswerte
	anydpi
Touchscreen-Typ	Keine Berührung
	Finger
Verfügbarkeit der Tastatur	keysexposed
	keyshidden
	keyssoft
Primäre Texteingabemethode	nokeys
	QWERTY
	12key
Verfügbarkeit der Navigationstasten	Kirchenschiff ausgesetzt
	navhidden
Primäre berührungslose Navigationsmethode	nonav
	dpad
	Trackball
	Rad
Plattformversion (API-Ebene)	Beispiele:
	v3
	v4
	v7

Ändern Sie das Gebietsschema der Android-Anwendung programmgesteuert

In den obigen Beispielen erfahren Sie, wie Sie Anwendungsressourcen lokalisieren. Im folgenden Beispiel wird erläutert, wie das Anwendungsgebietsschema in der Anwendung und nicht vom Gerät aus geändert wird. Um nur das Anwendungsgebietsschema zu ändern, können Sie das Gebietsschema util verwenden.

```
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();
    }
}
```

Initialisieren Sie das von Ihnen bevorzugte Gebietsschema aus der Anwendungsklasse.

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

Sie müssen auch eine Basisaktivität erstellen und diese Aktivität auf alle anderen Aktivitäten ausdehnen, sodass Sie das Gebietsschema der Anwendung nur an einer Stelle wie folgt ändern können:

```
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);
    }

}
```

Anmerkung: Initialisieren Sie das Gebietsschema immer im Konstruktor.

Jetzt können Sie LocalizationActivity wie folgt verwenden.

```
public class MainActivity extends LocalizationActivity {
```

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

    }
}
```

> Hinweis: Wenn Sie locale Anwendungs programmatisch zu ändern, müssen Sie Ihre Aktivitäten neu zu starten , um den Effekt von locale Änderung zu ergreifen , um richtig für diese Lösung , Sie und die Verwendung locale von gemeinsamen Präferenzen auf app Startup zu arbeiten `android:name=".LocaleApp"` In Sie Manifest.xml.

Manchmal werden Sie aufgefordert, den Release-Build zu erstellen. Um dieses Problem zu lösen, folgen Sie den nachstehenden Optionen.

Zuerst:

Wenn Sie die Übersetzung nur für einige Zeichenfolgen deaktivieren möchten, fügen Sie das folgende Attribut zu der Standardzeichenfolge string.xml hinzu

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

Zweite:

Ignoriere alle fehlenden Übersetzungen aus der Ressourcendatei. Füge folgendes Attribut hinzu. Es ist das Ignore-Attribut des Namespaces der Tools in deiner Strings-Datei:

```
<?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>
```

Dritte:

Eine andere Möglichkeit, nicht übersetzbare Zeichenfolgen zu deaktivieren

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

Wenn Sie über viele Ressourcen verfügen, die nicht übersetzt werden sollten, können Sie sie in einer Datei namens donottranslate.xml platzieren, und lint berücksichtigt alle nicht übersetzbaren Ressourcen.

Vierte:

Sie können auch ein Gebietsschema in der Ressourcendatei hinzufügen

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

Sie können auch die fehlende Übersetzungsprüfung für fusseln in app / build.gradle deaktivieren

```
lintOptions {

        disable 'MissingTranslation'
    }
```

Lokalisierung mit Ressourcen in Android online lesen:
https://riptutorial.com/de/android/topic/3345/lokalisierung-mit-ressourcen-in-android

Kapitel 153: Looper

Einführung

Ein `Looper` ist eine Android-Klasse, mit der eine Nachrichtenschleife für einen Thread ausgeführt wird, der normalerweise keine zugeordnet ist.

Der häufigste `Looper` in Android ist der Main-Loop, der auch als Main-Thread bezeichnet wird. Diese Instanz ist für eine Anwendung eindeutig und kann mit `Looper.getMainLooper()` statisch abgerufen werden.

Wenn ein `Looper` dem aktuellen Thread zugeordnet ist, kann er mit `Looper.myLooper()` abgerufen werden.

Examples

Erstellen Sie ein einfaches LooperThread

Ein typisches Beispiel für die Implementierung eines `Looper` Threads, das in der offiziellen Dokumentation angegeben ist, verwendet `Looper.prepare()` und `Looper.loop()` und `Looper.loop()` einen `Handler` der Schleife zwischen diesen Aufrufen zu.

```
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();
    }
}
```

Führen Sie eine Schleife mit einem HandlerThread aus

Ein `HandlerThread` kann verwendet werden, um einen Thread mit einem `Looper` zu starten. Dieser Looper kann dann verwendet werden, um einen `Handler` für die Kommunikation damit zu erstellen.

```
HandlerThread thread = new HandlerThread("thread-name");
thread.start();
Handler handler = new Handler(thread.getLooper());
```

Looper online lesen: https://riptutorial.com/de/android/topic/10593/looper

Kapitel 154: LruCache

Bemerkungen

Sie sollten Lru Cache in Anwendungen verwenden, in denen sich wiederholendes Laden von Ressourcen das reibungslose App-Verhalten beeinträchtigen würde. Zum Beispiel eine Fotogalerie mit großen Thumbnails (128x128).

Seien Sie immer vorsichtig mit der Größe des Lru-Cache, da die App möglicherweise zu hoch eingestellt wird.

Nachdem der Lru-Cache nicht mehr nützlich ist, sollten Sie keine Verweise darauf speichern, damit der Garbage Collector ihn aus dem Speicher entfernen kann.

Um optimale Leistung zu erzielen, sollten Sie Ressourcen wie Bitmaps mit den bewährten Methoden laden, beispielsweise eine geeignete inSampleSize auswählen, bevor Sie sie dem Lru-Cache hinzufügen.

Examples

Cache initialisieren

Der Lru-Cache speichert alle hinzugefügten Ressourcen (Werte) für den schnellen Zugriff, bis ein Speicherlimit erreicht ist. In diesem Fall wird die weniger verwendete Ressource (Wert) zum Speichern der neuen gelöscht.

Um den Lru-Cache zu initialisieren, müssen Sie einen maximalen Speicherwert angeben. Dieser Wert hängt von Ihren Anwendungsanforderungen ab und davon, wie wichtig die Ressource ist, um eine reibungslose App-Nutzung zu gewährleisten. Ein empfohlener Wert für eine Bildergalerie wäre beispielsweise 1/8 Ihres maximal verfügbaren Speichers.

Beachten Sie auch, dass der Lru-Cache auf Schlüsselwertbasis arbeitet. Im folgenden Beispiel ist der Schlüssel ein `String` und der Wert ist eine `Bitmap` :

```
int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);
int cacheSize = maxMemory / 8;

LruCache<String, Bitmap> = memoryCache = new LruCache<String, Bitmap>(cacheSize) {
    protected int sizeOf(String key, Bitmap bitmap) {
        return bitmap.getByteCount();
    }
};
```

Hinzufügen einer Bitmap (Ressource) zum Cache

Um dem Cache eine Ressource hinzuzufügen, müssen Sie einen Schlüssel und die Ressource angeben. Stellen Sie zunächst sicher, dass sich der Wert nicht bereits im Cache befindet

```
public void addResourceToMemoryCache(String key, Bitmap resource) {
    if (memoryCache.get(key) == null)
        memoryCache.put(key, resource);
}
```

Bitmap (Resouce) aus dem Cache abrufen

Um eine Ressource aus dem Cache zu erhalten, übergeben Sie einfach den Schlüssel Ihrer Ressource (in diesem Beispiel String).

```
public Bitmap getResourceFromMemoryCache(String key) {
    memoryCache.get(key);
}
```

LruCache online lesen: https://riptutorial.com/de/android/topic/7709/lrucache

Kapitel 155: Material Design

Einführung

Material Design ist ein umfassender Leitfaden für visuelles, Bewegungs- und Interaktionsdesign für Plattformen und Geräte.

Bemerkungen

Siehe auch den ursprünglichen Android-Blogeintrag, der die Design Support Library vorstellt

Offizielle Dokumentation

https://developer.android.com/design/material/index.html

Richtlinien für Materialdesign

https://material.io/richtlinien

Andere Designressourcen und Bibliotheken

https://design.google.com/resources/

Examples

Wenden Sie ein AppCompat-Design an

Die AppCompat-Unterstützungsbibliothek enthält Designs zum Erstellen von Apps mit der Material Design-Spezifikation . Ein Theme mit einem übergeordneten `Theme.AppCompat` von `Theme.AppCompat` ist auch für eine Aktivität erforderlich, um `AppCompatActivity` zu erweitern.

Der erste Schritt besteht darin, die Farbpalette Ihres Themas so anzupassen, dass Ihre App automatisch eingefärbt wird.
In Ihrer `res/styles.xml` App `res/styles.xml` Sie `res/styles.xml` definieren:

```
<!-- 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>
```

Anstelle von `Theme.AppCompat` , das einen dunklen Hintergrund hat, können Sie auch `Theme.AppCompat.Light` oder `Theme.AppCompat.Light.DarkActionBar` .

Sie können das Design mit Ihren eigenen Farben anpassen. Eine gute Auswahl finden Sie im Farbdiagramm für die Materialdesignspezifikation und in der Materialpalette . Die "500" -Farben sind eine gute Wahl für Primärfarben (in diesem Beispiel blau 500). Wählen Sie "700" für den dunklen Farbton. und eine Nuance aus einem anderen Farbton als Akzentfarbe. Die Primärfarbe wird für die Symbolleiste Ihrer App und für deren Eintrag in der Übersicht (letzte Apps) verwendet, die dunkle Variante zum Einblenden der Statusleiste und die Akzentfarbe, um einige Steuerelemente hervorzuheben.

Nachdem Sie dieses `AndroidManifest.xml` , wenden Sie es in der `AndroidManifest.xml` auf Ihre App an und wenden Sie das `AndroidManifest.xml` auch auf eine bestimmte Aktivität an. Dies ist nützlich, um ein `AppTheme.NoActionBar` anzuwenden, mit dem Sie nicht standardmäßige Symbolleisten-Konfigurationen implementieren können.

```
<application android:theme="@style/AppTheme"
    ...>
    <activity
        android:name=".MainActivity"
        android:theme="@style/AppTheme" />
</application>
```

Sie können Designs auch mit Hilfe von `android:theme` und einem `ThemeOverlay` Theme auf einzelne Ansichten `ThemeOverlay` . Zum Beispiel mit einer `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" />
```

oder ein `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>
```

Eine Symbolleiste hinzufügen

Eine `Toolbar` ist eine Verallgemeinerung von `ActionBar` zur Verwendung in Anwendungslayouts. Während eine `ActionBar` traditionell zum undurchsichtigen Fensterdekor einer `Activity's` das vom Framework gesteuert wird, kann eine `Toolbar` auf einer beliebigen Ebene der Verschachtelung innerhalb einer Ansichtshierarchie platziert werden. Es kann hinzugefügt werden, indem Sie die

folgenden Schritte ausführen:

1. **Stellen** Sie sicher, dass die folgende Abhängigkeit zur **build.gradle-** Datei Ihres Moduls (z. B. der App) unter Abhängigkeiten **hinzugefügt** wird:

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

2. `ActionBar` für Ihre App auf eines fest, für das es **keine** `ActionBar` . Bearbeiten Sie **dazu die** Datei **styles.xml** unter `res/values` und legen Sie ein `Theme.AppCompat` .
   In diesem Beispiel verwenden wir `Theme.AppCompat.NoActionBar` als übergeordnetes `AppTheme` Ihrer `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>
```

   Sie können auch `Theme.AppCompat.Light.NoActionBar` oder `Theme.AppCompat.DayNight.NoActionBar` oder ein anderes `Theme.AppCompat.DayNight.NoActionBar` , das nicht von Natur aus über eine `ActionBar`

3. Fügen Sie die `Toolbar` Ihrem Aktivitätslayout hinzu:

```
<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"/>
```

   Unter der `Toolbar` können Sie den Rest Ihres Layouts hinzufügen.

4. In Ihrer `Activity` , stellen Sie die `Toolbar` als `ActionBar` für diese `Activity` . Wenn Sie die *appcompat*- Bibliothek und eine `AppCompatActivity` , würden Sie die Methode `setSupportActionBar()` verwenden:

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

    final Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    //...
}
```

Nach den oben genannten Schritten können Sie die Verwendung `getSupportActionBar()` Methode , um die zu manipulieren `Toolbar` , die als gesetzt `ActionBar` .

Sie können den Titel beispielsweise wie folgt einstellen:

```
getSupportActionBar().setTitle("Activity Title");
```

Sie können beispielsweise auch die Titel- und Hintergrundfarbe wie folgt einstellen:

```
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)));
```

FloatingActionButton (FAB) hinzufügen

Im Materialentwurf repräsentiert eine schwebende Aktionsschaltfläche die Hauptaktion in einer Aktivität.
Sie werden durch ein eingekreistes Symbol über der Benutzeroberfläche unterschieden und weisen Bewegungsverhalten auf, zu denen Morphing, Starten und Übertragen eines Ankerpunkts gehören.

Stellen Sie sicher, dass die folgende Abhängigkeit zur build.gradle-Datei Ihrer App unter Abhängigkeiten hinzugefügt wird:

```
compile 'com.android.support:design:25.3.1'
```

`FloatingActionButton` Sie nun der Layoutdatei den `FloatingActionButton` :

```
<android.support.design.widget.FloatingActionButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_margin="16dp"
      android:src="@drawable/some_icon"/>
```

Dabei verweist das Attribut `src` auf das Symbol, das für die gleitende Aktion verwendet werden soll.
Das Ergebnis sollte in etwa so aussehen (vorausgesetzt, Ihre Akzentfarbe ist Material Pink):

Standardmäßig wird die Hintergrundfarbe Ihres `FloatingActionButton` auf die Akzentfarbe Ihres Themas eingestellt. Beachten Sie außerdem, dass ein `FloatingActionButton` einen Rand um ihn herum erfordert, damit er ordnungsgemäß funktioniert. Die empfohlene Marge für die Unterseite beträgt `16dp` für Telefone und `24dp` für Tablets.

Hier sind Eigenschaften, die Sie verwenden können, um den `FloatingActionButton` weiter anzupassen (vorausgesetzt, `xmlns:app="http://schemas.android.com/apk/res-auto` ist als

Namespace oben in Ihrem Layout deklariert):

- `app:fabSize` : Kann auf `normal` oder `mini` , um zwischen einer normalen oder einer kleineren Version zu wechseln.
- `app:rippleColor` : Legt die Farbe des Ripple-Effekts Ihres `FloatingActionButton` . Kann eine Farbressource oder ein Hex-String sein.
- `app:elevation` : Kann ein String, eine Ganzzahl, ein Boolean-Wert, ein Farbwert, ein Gleitkommawert und ein Dimensionswert sein.
- `app:useCompatPadding` : Compat-Auffüllung aktivieren. Möglicherweise ein boolescher Wert, z. B. " `true` oder " `false` . Setzen Sie diese Option auf `true` , um Compat-Padding auf api-21 und höher zu verwenden, um ein konsistentes Aussehen mit älteren API-Ebenen zu erhalten.

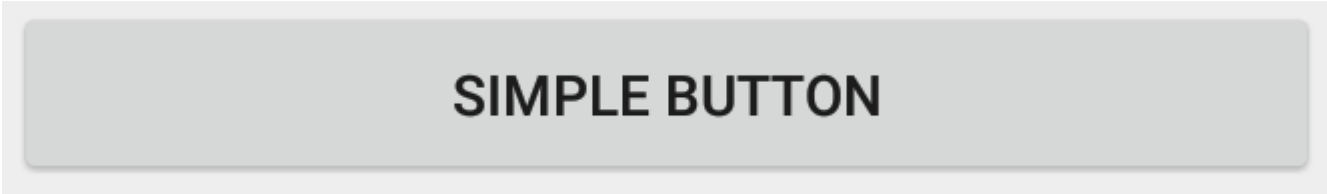
Weitere Beispiele zu FAB finden Sie hier .

Knöpfe im Material Design

Die AppCompat Support Library definiert mehrere nützliche Stile für Schaltflächen , von denen jeder einen Basis `Widget.AppCompat.Button` Stil erweitert, der standardmäßig auf alle Schaltflächen angewendet wird, wenn Sie ein `AppCompat` . Dieser Stil stellt sicher, dass alle Schaltflächen standardmäßig gemäß der Material Design-Spezifikation gleich aussehen.

In diesem Fall ist die Akzentfarbe rosa.

1. Einfacher Button: `@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. Farbige `@style/Widget.AppCompat.Button.Colored` : `@style/Widget.AppCompat.Button.Colored` Der Stil `Widget.AppCompat.Button.Colored` erweitert den Stil `Widget.AppCompat.Button` und wendet automatisch die **Akzentfarbe an, die** Sie in Ihrem App- `Widget.AppCompat.Button` ausgewählt haben.


```
<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"/>
```

Wenn Sie die Hintergrundfarbe anpassen möchten , ohne die Akzentfarbe in Ihrem *Hauptthema* ändern können Sie ein *eigenes Thema* (zur Verlängerung des erstellen `ThemeOverlay` Thema) für Ihren `Button` und es zu den wichtigsten Taste zuweisen `android:theme` Attribut:

```
<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"/>
```

Definieren Sie das `res/values/themes.xml` in `res/values/themes.xml` :

```
 <style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
      <item name="colorAccent">@color/my_color</item>
 </style>
```

3. Randloser Button: `@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. Randloser farbiger Button: `@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"/>
```

So verwenden Sie TextInputLayout

`build.gradle` Sie sicher, dass die folgende Abhängigkeit zur `build.gradle` Datei Ihrer App unter Abhängigkeiten hinzugefügt wird:

```
compile 'com.android.support:design:25.3.1'
```

Zeigt den Hinweis aus einem EditText als Floating-Label an, wenn ein Wert eingegeben wird.

```
<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>
```

Zur Anzeige des Augesymbols für die Kennwortanzeige mit TextInputLayout können wir den folgenden Code verwenden:

```
<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>
```

Dabei sind `app:passwordToggleEnabled="true"` und `android:inputType="textPassword"` -Parameter erforderlich.

`app` sollte den Namespace `xmlns:app="http://schemas.android.com/apk/res-auto"`

Weitere Details und Beispiele finden Sie im jeweiligen Thema .

TabLayout hinzufügen

TabLayout bietet ein horizontales Layout für die Anzeige von Registerkarten und wird üblicherweise in Verbindung mit einem ViewPager verwendet .

`build.gradle` Sie sicher, dass die folgende Abhängigkeit zur `build.gradle` Datei Ihrer App unter Abhängigkeiten hinzugefügt wird:

```
compile 'com.android.support:design:25.3.1'
```

Jetzt können Sie einem TabLayout in Ihrem Layout mithilfe der TabItem- Klasse Elemente hinzufügen.

Zum Beispiel:

```
<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>
```

Fügen Sie einen `OnTabSelectedListener` , der benachrichtigt wird, wenn eine Registerkarte im `TabLayout` ausgewählt / nicht ausgewählt / erneut ausgewählt wird:

```
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) {

    }
});
```

Tabs können auch programmgesteuert zum `TabLayout` hinzugefügt / daraus entfernt werden.

```
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` verfügt über zwei Modi: fest und scrollbar.

```
tabLayout.setTabMode(TabLayout.MODE_FIXED);
tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);
```

Diese können auch in XML angewendet werden:

```
<android.support.design.widget.TabLayout
    android:id="@+id/tabLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:tabMode="fixed|scrollable" />
```

Hinweis: Die `TabLayout` Modi schließen sich gegenseitig aus, dh es kann jeweils nur einer aktiv sein.

Die Anzeigefarbe des Registers ist die Akzentfarbe, die für Ihr Material Design-Thema definiert ist.

Sie können diese Farbe überschreiben, indem Sie einen benutzerdefinierten Stil in `styles.xml` und diesen Stil dann auf Ihr TabLayout anwenden:

```
<style name="MyCustomTabLayoutStyle" parent="Widget.Design.TabLayout">
    <item name="tabIndicatorColor">@color/your_color</item>
</style>
```

Dann können Sie den Stil mit der Ansicht auf die Ansicht anwenden:

```
<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

Der Ripple-Touch-Effekt wurde mit Materialdesign in Android 5.0 (API-Ebene 21) eingeführt. Die Animation wird von der neuen RippleDrawable- Klasse implementiert .

> Zeichenbar, das einen Welleneffekt als Reaktion auf Zustandsänderungen zeigt. Die Verankerungsposition der Welligkeit für einen bestimmten Zustand kann durch Aufrufen von `setHotspot(float x, float y)` mit der entsprechenden `setHotspot(float x, float y)` angegeben werden.

5,0

Im Allgemeinen **funktioniert der** Ripple-Effekt für **reguläre Schaltflächen standardmäßig** in API 21 und höher. Für andere berührbare Ansichten kann dies durch Angabe folgender Parameter erreicht werden:

```
android:background="?android:attr/selectableItemBackground">
```

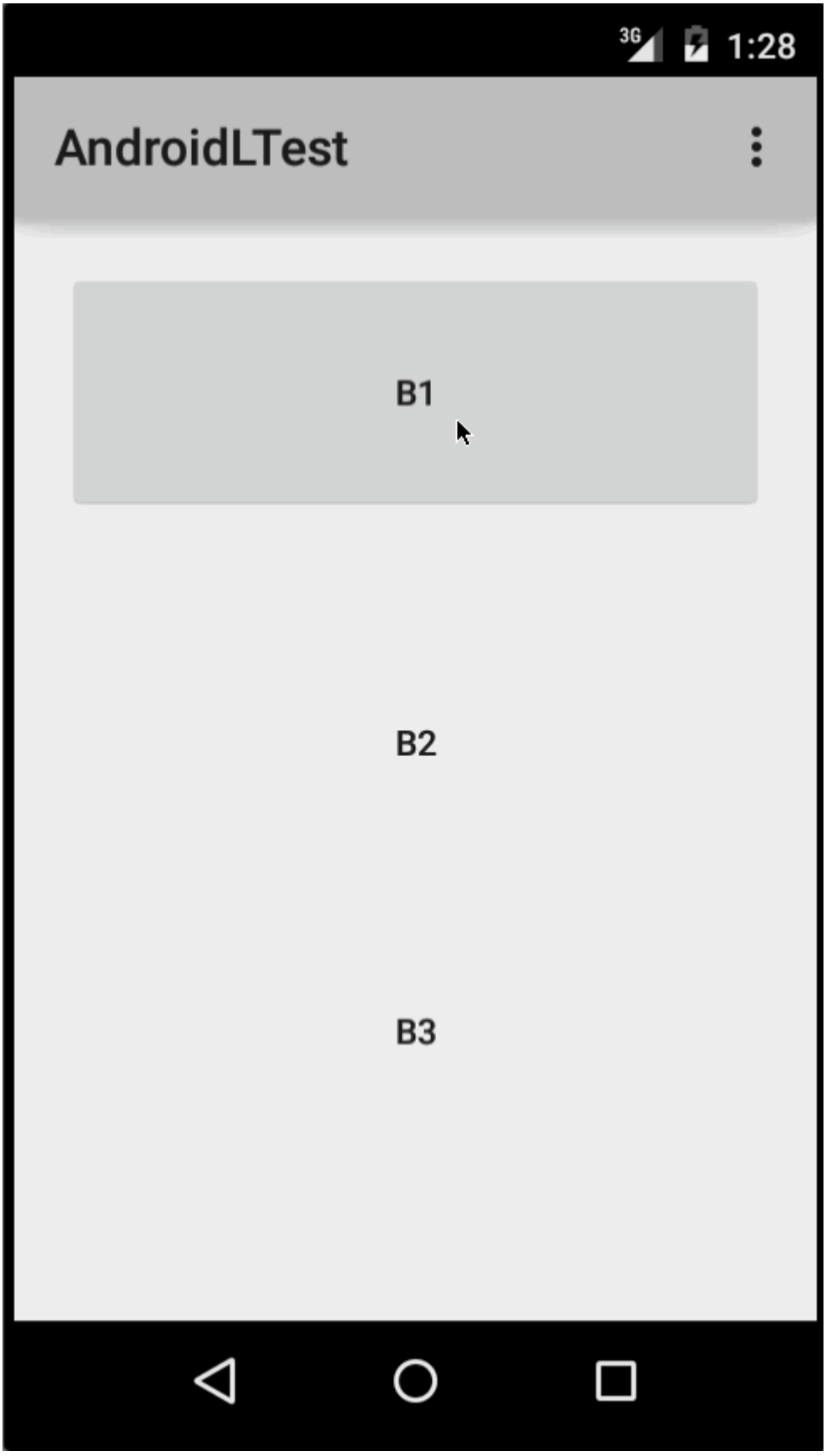
für Wellen in der Ansicht oder:

```
android:background="?android:attr/selectableItemBackgroundBorderless"
```

für Wellen, die über die Grenzen der Sicht hinausgehen.

Zum Beispiel in dem Bild unten

- B1 ist eine Schaltfläche, die keinen Hintergrund hat.
- B2 ist mit `android:background="android:attr/selectableItemBackground"`
- B3 ist mit `android:background="android:attr/selectableItemBackgroundBorderless"`



(Bild mit freundlicher Genehmigung: http://blog.csdn.net/a396901990/article/details/40187203)

Sie können dasselbe im Code erreichen, indem Sie:

```
int[] attrs = new int[]{R.attr.selectableItemBackground};
TypedArray typedArray = getActivity().obtainStyledAttributes(attrs);
```

```
int backgroundResource = typedArray.getResourceId(0, 0);
myView.setBackgroundResource(backgroundResource);
```

Wellen können auch mit dem Attribut `android:foreground` auf dieselbe Weise wie oben in einer Ansicht hinzugefügt werden. Wie der Name vermuten lässt, wird die Welligkeit, falls die Welligkeit zum Vordergrund hinzugefügt wird, über jeder hinzugefügten Ansicht `ImageView` (z. B. `ImageView` , ein `LinearLayout` mit mehreren Ansichten usw.).

Wenn Sie den Ripple-Effekt in eine Ansicht anpassen möchten, müssen Sie eine neue `XML` Datei innerhalb des zeichnbaren Verzeichnisses erstellen.

Hier einige Beispiele:

Beispiel 1 : Eine unbeschränkte Welligkeit

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#ffff0000" />
```

Beispiel 2 : Welligkeit mit Maske und Hintergrundfarbe

```
<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>
```

Wenn für eine `view` *bereits* ein Hintergrund mit einer `shape` , `corners` und anderen Tags festgelegt wurde, verwenden Sie eine Maskenebene, um eine Welligkeit hinzuzufügen `mask layer` und legen Sie die Welligkeit als Hintergrund der Ansicht fest.

Beispiel :

```
 <?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>
```

Beispiel 3 : Kräuseln Sie eine zu zeichnende Ressource

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#ff0000ff">
    <item android:drawable="@drawable/my_drawable" />
</ripple>
```

Verwendung: Um Ihre Ripple-XML-Datei an eine Ansicht anzuhängen, legen Sie sie als Hintergrund wie folgt fest (vorausgesetzt, Ihre Ripple-Datei hat den Namen `my_ripple.xml`):

```
<View
    android:id="@+id/myViewId"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/my_ripple" />
```

Wähler:

Die Welligkeit ziehbar kann auch anstelle von Farbzustandsliste Selektoren verwendet werden , wenn Ihr Ziel Version v21 oder höher (man kann auch die Welligkeit Wähler in der ist `drawable-v21` - Ordner):

```
<!-- /drawable/button.xml: -->
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true" android:drawable="@drawable/button_pressed"/>
    <item android:drawable="@drawable/button_normal"/>
</selector>

<!--/drawable-v21/button.xml:-->
<?xml version="1.0" encoding="utf-8"?>
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="?android:colorControlHighlight">
   <item android:drawable="@drawable/button_normal" />
</ripple>
```

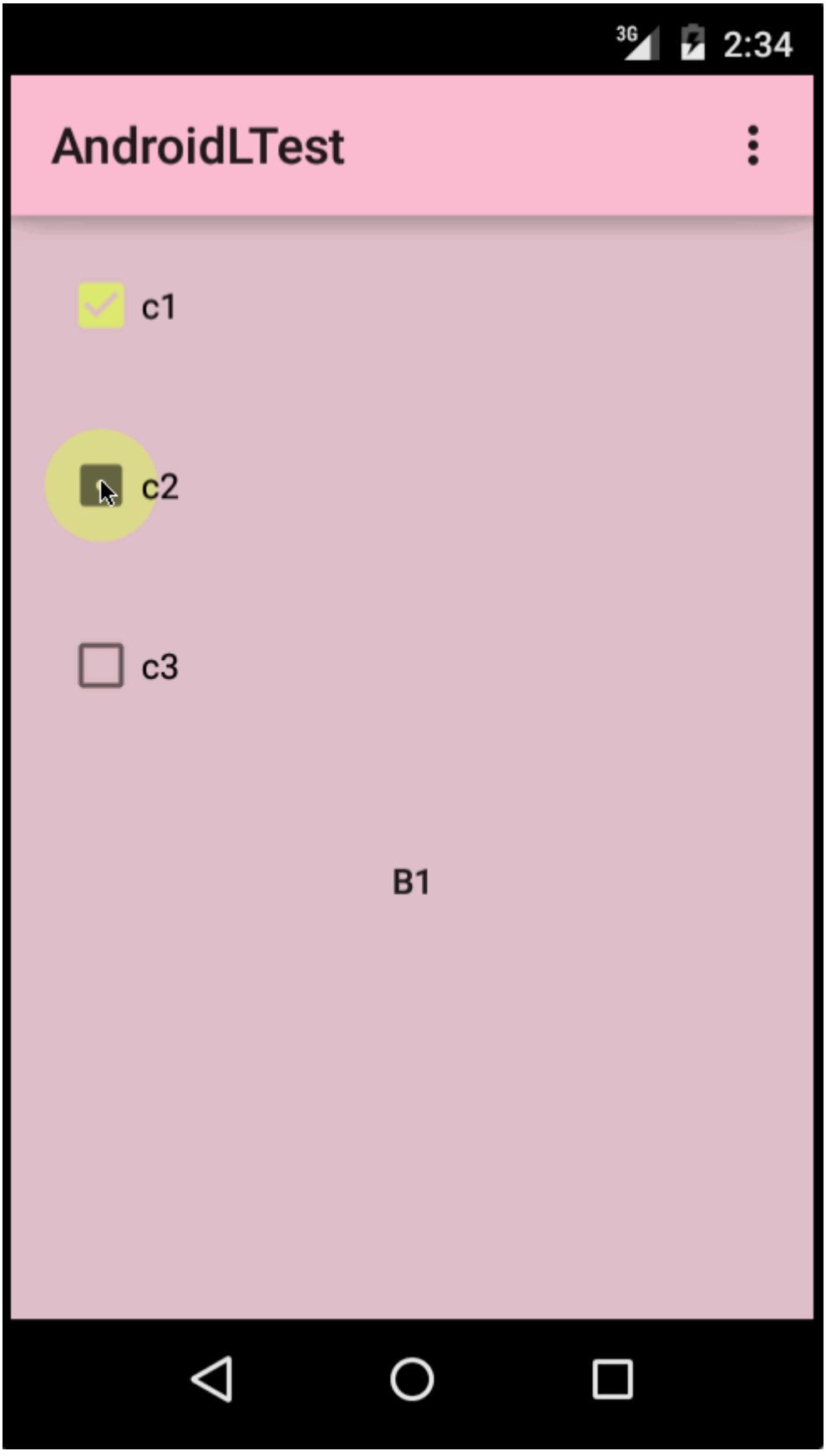
In diesem Fall wäre die Farbe des Standardstatus Ihrer Ansicht weiß und der gedrückte Status würde die Welligkeit anzeigen, die gezeichnet werden kann.

Hinweis: Wenn Sie `?android:colorControlHighlight` erhält die `?android:colorControlHighlight` die gleiche Farbe wie die in Ihrer App integrierten Wellen.

Um nur die `android:colorControlHighlight` zu ändern, können Sie die Farbe `android:colorControlHighlight` in Ihrem `android:colorControlHighlight` wie `android:colorControlHighlight` anpassen:

```
<?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>
```

und dann verwenden Sie dieses Thema in Ihren Aktivitäten usw. Der Effekt wäre wie das Bild unten:



(Bild mit freundlicher Genehmigung: http://blog.csdn.net/a396901990/article/details/40187203)

Fügen Sie eine Navigationsleiste hinzu

Navigations-Schubladen werden verwendet, um zu den Zielen der obersten Ebene in einer App zu navigieren.

`build.gradle` Sie sicher, dass Sie in Ihrer `build.gradle` Datei unter Abhängigkeiten eine Design-Support-Bibliothek hinzugefügt haben:

```
 dependencies {
    // ...
    compile 'com.android.support:design:25.3.1'
}
```

`DrawerLayout` als Nächstes das `DrawerLayout` und die `NavigationView` in Ihre XML-Layout-Ressourcendatei ein.
Das `DrawerLayout` ist nur ein `DrawerLayout` Container, mit dem die `NavigationView` , die eigentliche Navigationsleiste, von links oder rechts aus dem Bildschirm `DrawerLayout` . Hinweis: Für mobile Geräte beträgt die Standardgröße der Schublade 320 dB.

```
<!-- 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>
```

Wenn Sie möchten, erstellen Sie jetzt eine **Header-Datei** , die als oberste Ebene Ihrer Navigationsleiste dient. Dies wird verwendet, um der Schublade ein eleganteres Aussehen zu verleihen.

```
<!-- 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>
```

Es wird im `NavigationView` Tag in der `app:headerLayout="@layout/drawer_header"` -Attribut.
Diese `app:headerLayout` füllt das angegebene Layout automatisch in die Kopfzeile auf. Dies kann alternativ zur Laufzeit erfolgen mit:

```
// 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);
```

Um Ihre Navigationsleiste automatisch mit Materialdesign-kompatiblen Navigationselementen zu füllen, erstellen Sie eine Menüdatei und fügen Sie nach Bedarf Elemente hinzu. Hinweis: Symbole für Elemente sind zwar nicht erforderlich, werden jedoch in der Material Design-Spezifikation vorgeschlagen .
Es wird im `NavigationView` Tag in der `app:menu="@menu/navigation_menu" attribute` referenziert `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>
```

Um Elemente in Gruppen zu unterteilen, fügen Sie sie mit einem `android:title` Attribut in ein `<menu>` ein, das in einem anderen `<item>` `<menu>` verschachtelt ist, oder umschließen Sie sie mit dem `<group>` -Tag.

Nachdem das Layout fertig ist, fahren Sie mit dem `Activity` :

```
  // Find the navigation view
  NavigationView navigationView = (NavigationView) findViewById(R.id.navigation_drawer);
  navigationView.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
      @Override
      public boolean onNavigationItemSelected(MenuItem item) {
          // Get item ID to determine what to do on user click
          int itemId = item.getItemId();
          // Respond to Navigation Drawer selections with a new Intent
          startActivity(new Intent(this, OtherActivity.class));
          return true;
      }
  });

  DrawerLayout drawer = (DrawerLayout) findViewById(R.id.navigation_drawer_layout);
  // Necessary for automatically animated navigation drawer upon open and close
  ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawer, "Open navigation
drawer", "Close navigation drawer");
  // The two Strings are not displayed to the user, but be sure to put them into a separate
strings.xml file.
  drawer.addDrawerListener(toggle);
  toogle.syncState();
```

Sie können jetzt in der Kopfansicht der `NavigationView`

```
View headerView = navigationView.getHeaderView();
TextView headerTextView = (TextView) headerview.findViewById(R.id.header_text_view);
ImageView headerImageView = (ImageView) headerview.findViewById(R.id.header_image);
// Set navigation header text
headerTextView.setText("User name");
// Set navigation header image
headerImageView.setImageResource(R.drawable.header_image);
```

Die Header - Ansicht verhält sich wie jeder andere `View` , so , wenn Sie verwenden `findViewById()` und einige andere hinzufügen `View` s zum Layout - Datei, können Sie die Eigenschaften von irgendetwas in ihm festlegen.

Weitere Details und Beispiele finden Sie im jeweiligen Thema .

Unterblätter in der Design Support Library

Die unteren Blätter werden vom unteren Rand des Bildschirms nach oben verschoben, um mehr Inhalt anzuzeigen.
Sie wurden der Android Support Library in Version 25.1.0 hinzugefügt und unterstützen vor allem die Versionen.

Stellen Sie sicher, dass die folgende Abhängigkeit zur build.gradle-Datei Ihrer App unter Abhängigkeiten hinzugefügt wird:

```
compile 'com.android.support:design:25.3.1'
```

Persistente untere Blätter

Sie können ein beständiges Bottom-Sheet erreichen, indem Sie ein `BottomSheetBehavior` an eine `BottomSheetBehavior` Ansicht eines `CoordinatorLayout` anhängen:

```
<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>
```

Dann können Sie in Ihrem Code eine Referenz erstellen mit:

```
// The View with the BottomSheetBehavior
View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

Sie können den Status Ihres BottomSheetBehavior mithilfe der setState () -Methode festlegen :

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

Sie können einen dieser Zustände verwenden:

- `STATE_COLLAPSED` : Dieser `STATE_COLLAPSED` Status ist die Standardeinstellung und zeigt nur einen Teil des Layouts am unteren Rand. Die Höhe kann mit dem `app:behavior_peekHeight` Attribut (Standardeinstellung 0) gesteuert werden.

- `STATE_EXPANDED` : Der vollständig erweiterte Zustand des untersten Blattes, wobei entweder

das gesamte unterste Blatt sichtbar ist (wenn seine Höhe geringer als das enthaltene `CoordinatorLayout`) oder das gesamte `CoordinatorLayout` gefüllt ist

- `STATE_HIDDEN` : Standardmäßig deaktiviert (und mit dem Attribut `app:behavior_hideable` aktiviert). `STATE_HIDDEN` können Benutzer das untere Blatt nach unten streichen, um das untere Blatt vollständig auszublenden

Um das BottomSheet beim Klicken einer Ansicht Ihrer Wahl zu öffnen oder zu schließen, A Button, sagen wir, hier ist, wie Sie das Blattverhalten und die Aktualisierungsansicht umschalten können.

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

Das Verhalten von `BottomSheet` bietet jedoch auch eine Funktion, mit der Benutzer mit einer DRAG-Bewegung mit dem Wischen nach oben oder unten interagieren können. In einem solchen Fall können wir die abhängige Ansicht (wie die Schaltfläche oben) möglicherweise nicht aktualisieren, wenn sich der Blattstatus geändert hat. Aus diesem `BottomSheetCallback` Sie Rückrufe von `BottomSheetCallback` erhalten. Daher können Sie `BottomSheetCallback` hinzufügen, um User Swipe-Ereignisse zu überwachen:

```
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
   }
 });
```

Und wenn Sie möchten, dass Ihr unteres Blatt nur im COLLAPSED- und EXPANDED-Modus angezeigt wird, schalten Sie es um und verwenden Sie niemals die Option HIDE:

```
mBottomSheetBehavior2.setHideable(false);
```

Unterblatt DialogFragment

Sie können auch ein BottomSheetDialogFragment anstelle einer Ansicht im unteren Arbeitsblatt anzeigen. Dazu müssen Sie zunächst eine neue Klasse erstellen, die BottomSheetDialogFragment erweitert.

Innerhalb der `setupDialog()` -Methode können Sie eine neue Layoutdatei aufblasen und das BottomSheetBehavior der Containeransicht in Ihrer Aktivität abrufen. Sobald Sie das Verhalten haben, können Sie einen BottomSheetCallback erstellen und damit verknüpfen, um das Fragment zu verwerfen, wenn das Blatt ausgeblendet ist.

```
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);
        }
    }
}
```

Schließlich können Sie show () für eine Instanz Ihres Fragments aufrufen, um es im unteren Blatt anzuzeigen.

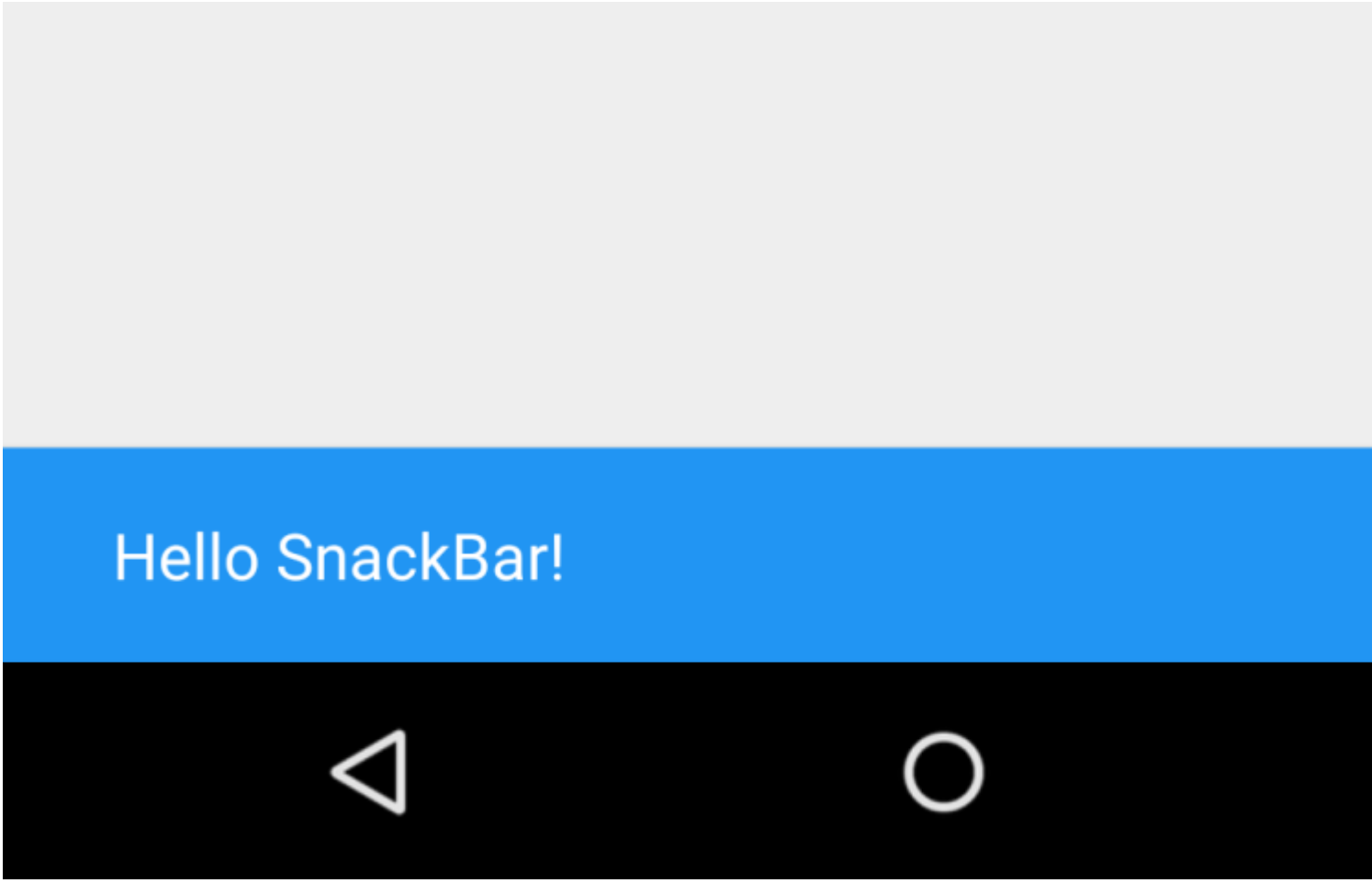
```
BottomSheetDialogFragment bottomSheetDialogFragment = new BottomSheetDialogFragmentExample();
bottomSheetDialogFragment.show(getSupportFragmentManager(),
bottomSheetDialogFragment.getTag());
```

Weitere Details finden Sie im jeweiligen Thema

Fügen Sie eine Snackbar hinzu

Eines der Hauptmerkmale in Material Design ist das Hinzufügen einer `Snackbar` , die theoretisch den vorherigen `Toast` . Wie in der Android-Dokumentation:

> Snackbars enthalten eine einzelne Textzeile, die sich direkt auf die ausgeführte Operation bezieht. Sie können eine Textaktion enthalten, jedoch keine Symbole. Toasts werden hauptsächlich für die Systembenachrichtigung verwendet. Sie werden auch am unteren Rand des Bildschirms angezeigt, dürfen jedoch nicht vom Bildschirm weggezogen werden.



Toasts können in Android weiterhin für die Anzeige von Nachrichten für Benutzer verwendet werden. Wenn Sie sich jedoch für die Verwendung des Materialdesigns in Ihrer App entschieden haben, wird empfohlen, eine Snackbar zu verwenden. Anstatt als Überlagerung auf Ihrem Bildschirm angezeigt zu werden, erscheint eine `Snackbar` von unten.

So wird es gemacht:

```
Snackbar snackbar = Snackbar
        .make(coordinatorLayout, "Here is your new Snackbar", Snackbar.LENGTH_LONG);
snackbar.show();
```

In `Snackbar` auf die Zeitdauer, in der die `Snackbar` , haben wir die Optionen, die denen eines `Toast` ähnlich sind, oder wir können eine benutzerdefinierte Dauer in Millisekunden einstellen:

- `LENGTH_SHORT`
- `LENGTH_LONG`
- `LENGTH_INDEFINITE`
- `setDuration()` (seit Version `22.2.1`)

Sie können Ihrer `Snackbar` auch dynamische Funktionen hinzufügen, z. B. `ActionCallback` oder benutzerdefinierte Farben. Beachten Sie beim Anpassen einer `Snackbar` jedoch die Design-Richtlinie von Android.

Die Implementierung der `Snackbar` hat jedoch eine Einschränkung. Das übergeordnete Layout der Ansicht, in der Sie eine `Snackbar` implementieren `Snackbar` , muss ein `CoordinatorLayout` . Dies ist so, dass das eigentliche Popup von unten gemacht werden kann.

So definieren Sie ein `CoordinatorLayout` in Ihrer Layout-XML-Datei:

```
<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>
```

Das `CoordinatorLayout` muss dann in der `onCreate` Methode Ihrer Activity definiert und dann beim Erstellen der `Snackbar` selbst verwendet werden.

Weitere Informationen zur `Snackbar` finden Sie in der offiziellen Dokumentation oder in dem jeweiligen Thema in der Dokumentation.

Material Design online lesen: https://riptutorial.com/de/android/topic/124/material-design

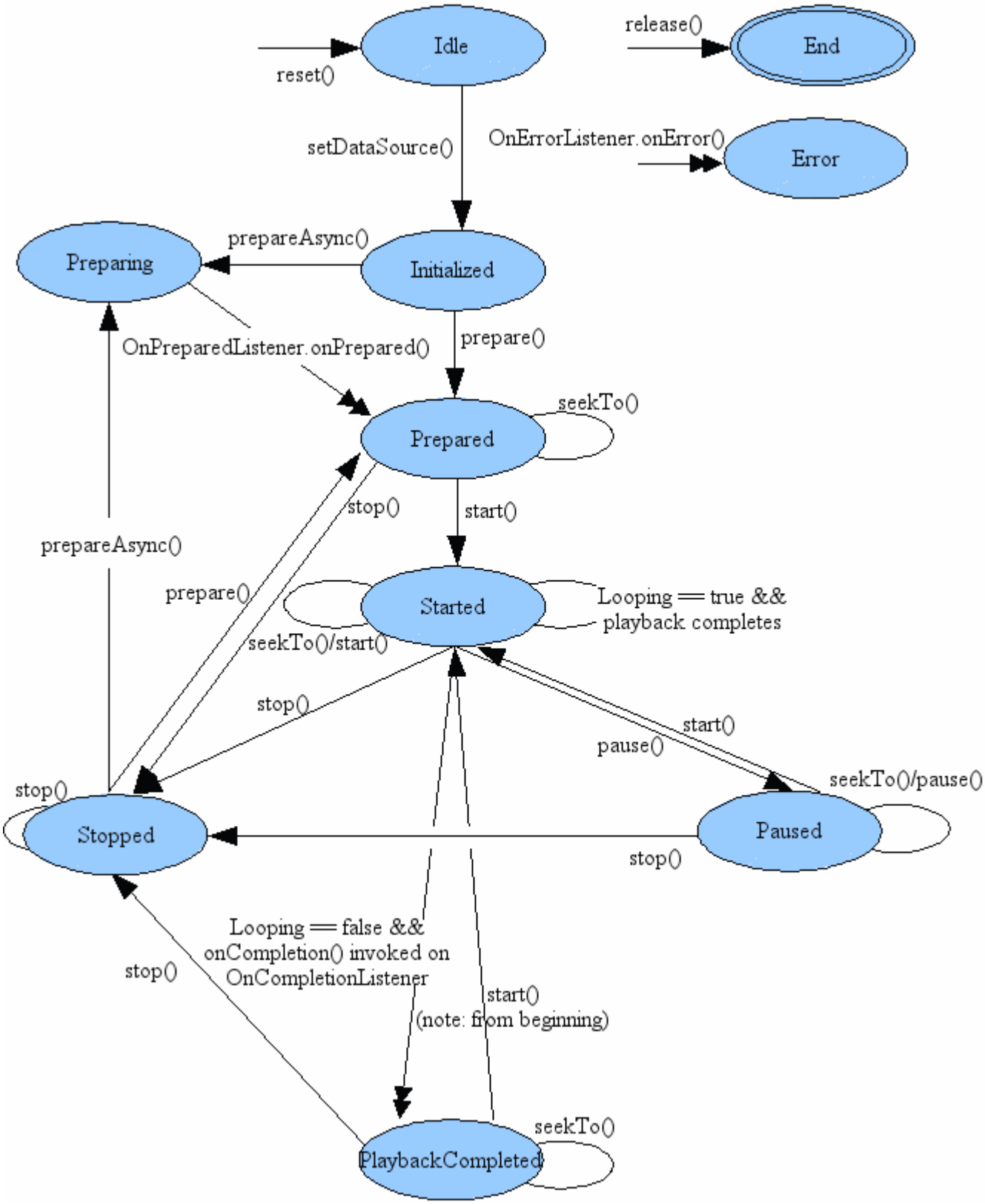
Kapitel 156: Media Player

Syntax

- void setAudioStreamType (int streamtype)
- void setDataSource (Kontext-Kontext, Uri-Uri)
- ungültig vorbereiten ()
- void prepareAsync ()
- void start ()
- void stop ()

Bemerkungen

Die Verwendung des `MediaPlayer` basiert hauptsächlich auf dem Zustandsdiagramm:



Das heißt, um Audio / Video abzuspielen, muss eine bestimmte Reihenfolge von Aktionen festgelegt werden. Es gibt auch an, welche Aktionen in welchem Zustand durchgeführt werden können .

Der MediaPlayer-API fehlt die Flexibilität (Hinzufügen benutzerdefinierter Decoder- und Rendering-Logik) und es fehlt die Unterstützung für Dynamic Adaptive Streaming über HTTP (DASH) und SmoothStreaming. Sehen Sie sich dazu den ExoPlayer an .

Examples

Grundlegendes Schaffen und Spielen

Mit der MediaPlayer-Klasse können Sie die Wiedergabe von Audio- / Videodateien und Streams steuern.

Die Erstellung eines MediaPlayer-Objekts kann auf drei Arten erfolgen:

1. Medien aus lokaler Ressource

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.resource);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

2. Von der lokalen URI (erhalten von ContentResolver)

```
Uri myUri = ....; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

3. Von externer URL

```
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();
```

Asynchrone Vorbereitung

Der `MediaPlayer$prepare()` ist ein blockierender Aufruf und friert die Benutzeroberfläche ein, bis die Ausführung abgeschlossen ist. Um dieses Problem zu lösen, kann der `MediaPlayer$prepareAsync()` verwendet werden.

```
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
```

Bei synchronen Vorgängen werden Fehler normalerweise mit einer Ausnahme oder einem Fehlercode gemeldet. Wenn Sie jedoch asynchrone Ressourcen verwenden, sollten Sie sicherstellen, dass Ihre Anwendung über Fehler entsprechend benachrichtigt wird. Für

MediaPlayer

```
mMediaPlayer.setOnErrorListener(new MediaPlayer.OnErrorListener(){
        @Override
        public boolean onError(MediaPlayer mp, int what, int extra) {
            // ... react appropriately ...
            // The MediaPlayer has moved to the Error state, must be reset!
            // Then return true if the error has been handled
        }
});
```

Systemklingeltöne abrufen

Dieses Beispiel zeigt, wie Sie die URIs von Systemklingeltönen abrufen (`RingtoneManager.TYPE_RINGTONE`):

```
private List<Uri> loadLocalRingtonesUris() {
        List<Uri> alarms = new ArrayList<>();
        try {
            RingtoneManager ringtoneMgr = new RingtoneManager(getActivity());
            ringtoneMgr.setType(RingtoneManager.TYPE_RINGTONE);

            Cursor alarmsCursor = ringtoneMgr.getCursor();
            int alarmsCount = alarmsCursor.getCount();
            if (alarmsCount == 0 && !alarmsCursor.moveToFirst()) {
                alarmsCursor.close();
                return null;
            }

            while (!alarmsCursor.isAfterLast() && alarmsCursor.moveToNext()) {
                int currentPosition = alarmsCursor.getPosition();
                alarms.add(ringtoneMgr.getRingtoneUri(currentPosition));
            }

        } catch (Exception ex) {
            ex.printStackTrace();
        }

        return alarms;
    }
```

Die Liste hängt von der Art der angeforderten Klingeltöne ab. Die Möglichkeiten sind:

- `RingtoneManager.TYPE_RINGTONE`
- `RingtoneManager.TYPE_NOTIFICATION`
- `RingtoneManager.TYPE_ALARM`
- `RingtoneManager.TYPE_ALL = TYPE_RINGTONE | TYPE_NOTIFICATION | TYPE_ALARM`

Um die Klingeltöne als `android.media.Ringtone` muss jeder `Uri` vom `RingtoneManager` aufgelöst werden:

```
android.media.Ringtone osRingtone = RingtoneManager.getRingtone(context, uri);
```

Um den Ton abzuspielen, verwenden Sie die Methode:

```
public void setDataSource(Context context, Uri uri)
```

von `android.media.MediaPlayer` . `MediaPlayer` muss gemäß dem Zustandsdiagramm initialisiert und vorbereitet werden

Systemlautstärke abrufen und einstellen

Audiostream-Typen

Es gibt verschiedene Profile von Klingelton-Streams. Jeder von ihnen hat ein anderes Volumen.

Jedes Beispiel hier ist für `AudioManager.STREAM_RING` Stream-Typ `AudioManager.STREAM_RING` geschrieben. Dies ist jedoch nicht der einzige. Die verfügbaren Stream-Typen sind:

- `STREAM_ALARM`
- `STREAM_DTMF`
- `STREAM_MUSIC`
- `STREAM_NOTIFICATION`
- `STREAM_RING`
- `STREAM_SYSTEM`
- `STREAM_VOICE_CALL`

Lautstärke einstellen

Um das Volumen eines bestimmten Profils zu erhalten, rufen Sie an:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int currentVolume = audioManager.getStreamVolume(AudioManager.STREAM_RING);
```

Dieser Wert ist sehr wenig nützlich, wenn der Maximalwert für den Stream nicht bekannt ist:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int streamMaxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM_RING);
```

Das Verhältnis dieser beiden Werte ergibt ein relatives Volumen (0 <Volumen <1):

```
float volume = ((float) currentVolume) / streamMaxVolume
```

Lautstärke um einen Schritt einstellen

Um das Volume für den Stream um eine Stufe zu erhöhen, rufen Sie Folgendes auf:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_RAISE, 0);
```

Um das Volume für den Stream um einen Schritt zu verringern, rufen Sie Folgendes auf:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_LOWER, 0);
```

Festlegen, dass MediaPlayer einen bestimmten Stream-Typ verwendet

Dazu gibt es eine Hilfsfunktion aus der `MediaPlayer` Klasse.
Rufen `void setAudioStreamType(int streamtype)` einfach `void setAudioStreamType(int streamtype)` :

```
MediaPlayer mMedia = new MediaPlayer();
mMedia.setAudioStreamType(AudioManager.STREAM_RING);
```

Media Player mit Pufferfortschritt und Wiedergabeposition

```
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) {
```

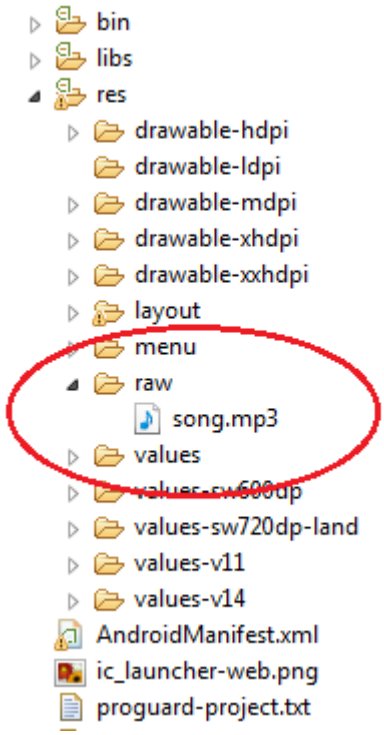
```
                observer.stop();
                progress_bar.setProgress(mp.getCurrentPosition());
                // TODO Auto-generated method stub
                mediaPlayer.stop();
                mediaPlayer.reset();
            }
        });
        mediaPlayer.setOnBufferingUpdateListener(new 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>
```

Importieren Sie Audio in Androidstudio und spielen Sie es ab

Dies ist ein Beispiel, wie Sie eine Audiodatei abspielen können, die Sie bereits auf Ihrem PC / Laptop haben. Erstellen Sie zunächst ein neues Verzeichnis unter res und benennen Sie es als roh



Kopieren Sie das Audio, das Sie abspielen möchten, in diesen Ordner. Es kann sich um eine MP3- oder WAV-Datei handeln.

Wenn Sie zum Beispiel auf den Knopf klicken, möchten Sie diesen Sound wiedergeben. So wird es gemacht

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

Dadurch wird der Song nur einmal abgespielt, wenn auf die Schaltfläche geklickt wird. Wenn Sie den Song bei jedem Klick erneut abspielen möchten, klicken Sie auf den folgenden Code

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

Media Player online lesen: https://riptutorial.com/de/android/topic/1851/media-player

Kapitel 157: MediaSession

Syntax

- void mediaSessionCompat.setFlags (Int Flags)
- void mediaSessionCompat.setMediaButtonReceiver (PendingIntent mbr)
- void mediaSessionCompat.setCallback (Rückruf von MediaSessionCompat.Callback)
- void mediaSessionCompat.setActive (boolean aktiv)
- MediaSessionCompat.Token mediaSessionCompat.getSessionToken ()
- void mediaSessionCompat.release ()
- void mediaSessionCompat.setPlaybackState (PlaybackStateCompat-Status)
- void mediaSessionCompat.setMetadata (Metadaten von MediaMetadataCompat)

Bemerkungen

Verwenden Sie für die beste Praxis die **medienkompatible** Bibliothek. Die Bibliothek sorgt für Abwärtskompatibilität, indem sie Mediensitzungsmethoden in die entsprechenden Methoden auf älteren Plattformversionen konvertiert, sofern verfügbar.

Examples

Schaltflächenereignisse empfangen und bearbeiten

In diesem Beispiel wird ein `MediaSession` Objekt erstellt, wenn ein `Service` gestartet wird. Das `MediaSession` Objekt wird freigegeben, wenn der `Service` zerstört wird:

```
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();
    }
}
```

Die folgende Methode instanziiert und konfiguriert die `MediaSession` der `MediaSession` Schaltfläche:

```
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);
}
```

Die folgende Methode sendet Metadaten (in einer `HashMap` gespeichert) über A2DP an das Gerät:

```
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);
}
```

Die folgende Methode legt den `PlaybackState` . Außerdem wird festgelegt, auf welche `MediaSession` die `MediaSession` reagiert:

```
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);
}
```

MediaSession online lesen: https://riptutorial.com/de/android/topic/6250/mediasession

Kapitel 158: MediaStore

Examples

Rufen Sie Audio- / MP3-Dateien aus einem bestimmten Ordner des Geräts ab oder rufen Sie alle Dateien ab

Fügen Sie dem Manifest Ihres Projekts zunächst die folgenden Berechtigungen hinzu, um den Zugriff auf den Gerätespeicher zu ermöglichen:

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

Erstellen Sie dann die Datei *AudioModel.class* und *fügen Sie* die folgende Modellklasse hinzu, um *Listenelemente abzurufen* und *festzulegen* :

```
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;
    }
}
```

Verwenden Sie anschließend die folgende Methode, um alle MP3-Dateien aus einem Ordner Ihres Geräts oder alle Dateien Ihres Geräts zu lesen:

```
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;
}
```

Der obige Code gibt eine Liste aller MP3-Dateien mit Name, Pfad, Interpret und Album der Musik zurück. Weitere Informationen finden Sie in der Dokumentation zu Media.Store.Audio .

Um Dateien eines bestimmten Ordners zu lesen, verwenden Sie die folgende Abfrage (Sie müssen den Ordnernamen ersetzen):

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    MediaStore.Audio.Media.DATA + " like ? ",
    new String[]{"%yourFolderName%"}, // Put your device folder / file location here.
    null);
```

Wenn Sie alle Dateien von Ihrem Gerät abrufen möchten, verwenden Sie die folgende Abfrage:

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    null,
    null,
    null);
```

Hinweis: Vergessen Sie nicht, Speicherzugriffsberechtigungen zu aktivieren.

Jetzt müssen Sie nur noch die obige Methode aufrufen, um die MP3-Dateien zu erhalten:

```
getAllAudioFromDevice(this);
```

Beispiel mit Aktivität

```
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;
    }
}
```

MediaStore online lesen: https://riptutorial.com/de/android/topic/7136/mediastore

Kapitel 159: Moshi

Einführung

Moshi ist eine moderne JSON-Bibliothek für Android und Java. Es macht es einfach, JSON in Java-Objekte und Java wieder in JSON zu parsen.

Bemerkungen

Vergessen Sie nicht, lesen Sie immer die README !

Examples

JSON in Java

```
String json = ...;

Moshi moshi = new Moshi.Builder().build();
JsonAdapter<BlackjackHand> jsonAdapter = moshi.adapter(BlackjackHand.class);

BlackjackHand blackjackHand = jsonAdapter.fromJson(json);
System.out.println(blackjackHand);
```

Java-Objekte als JSON serialisieren

```
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);
```

Eingebaute Typadapter

Moshi bietet integrierte Unterstützung zum Lesen und Schreiben von Java-Kerndatentypen:

- Primitive (int, float, char ...) und ihre geschachtelten Gegenstücke (Integer, Float, Character ...).
- Arrays
- Sammlungen
- Listen
- Sets
- Maps Strings Enums

Es unterstützt Ihre Modellklassen, indem Sie sie Feld für Feld ausschreiben. Im obigen Beispiel verwendet Moshi diese Klassen:

```
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"
    }
  ]
}
```

Moshi online lesen: https://riptutorial.com/de/android/topic/8744/moshi

Kapitel 160: Multidex- und Dex-Methodenlimit

Einführung

DEX bedeutet ausführbare Bytecode-Dateien von Android-Apps (APK) in Form von Dalvik Executable-Dateien (DEX-Dateien), die den kompilierten Code enthalten, der zum Ausführen Ihrer App verwendet wird.

Die Dalvik Executable-Spezifikation begrenzt die Gesamtzahl der Methoden, die in einer einzelnen DEX-Datei referenziert werden können, auf 65.536 (64 KB). Dies schließt Android-Framework-Methoden, Bibliotheksmethoden und Methoden in Ihrem eigenen Code ein.

Um dieses Limit zu überwinden, müssen Sie den App-Erstellungsprozess so konfigurieren, dass er mehr als eine DEX-Datei generiert, die als Multidex bezeichnet wird.

Bemerkungen

Was ist dex

Dex ist der Name des Dateiformats und der Kodierung, für die Android Java-Code kompiliert wird. In früheren Android-Versionen wurden `dex` Binärdateien direkt in einer virtuellen Maschine namens Dalvik geladen und ausgeführt. Neuere Versionen von Android verwenden die Android Runtime (ART), die `dex` Dateien als Zwischendarstellung behandelt und vor dem Ausführen der Anwendung weitere Kompilierungen `dex` .

Dex ist ein sehr altes Dateiformat in Bezug auf die Lebensdauer von Smartphones und wurde für Geräte entwickelt, deren Hauptspeicher in Dutzenden Megabytes gemessen wurde. Die gestalterischen Einschränkungen dieser Tage sind bis heute geblieben.

Das Problem:

Das `dex` Dateiformat kodiert eine Begrenzung für die Anzahl der Methoden, die in einer einzelnen Binärdatei referenziert werden können. Da der Teil des Dateiformats, in dem die Anzahl der Verweise gespeichert ist, zwei Byte lang ist, beträgt die maximale Anzahl der Methodenverweise `0xFFFF` oder 65535. Wenn eine Anwendung mehr als diese Anzahl von Methodenverweisen enthält, wird die Kompilierung fehlschlagen.

Was Sie dagegen tun können:

Google hat dieses Problem umgangen, genannt Multidex. Es enthält Kompilierungszeit- und Laufzeitkomponenten. Wie der Name schon sagt, wird der Code bei der Kompilierung in eine oder mehrere `dex` Dateien aufgeteilt. Zur Laufzeit wird dem Standard- `ClassLoader` wie Klassen aus diesen Dateien `ClassLoader` werden.

Dieser Ansatz funktioniert gut mit neueren Geräten, weist jedoch einige erhebliche Nachteile auf. Dies kann die Startzeit für Anwendungen erheblich verlängern und auf älteren Geräten kann es zu Fehlern bei der `Application Not Responding` führen.

Multidex sollte zwar wirksam sein, jedoch möglichst vermieden werden.

Wie man das Limit vermeidet:

Bevor Sie Ihre App für die Verwendung von Methodenreferenzen mit 64 KB oder mehr konfigurieren, sollten Sie die Gesamtzahl der durch Ihren App-Code aufgerufenen Referenzen reduzieren, einschließlich der durch Ihren App-Code oder die enthaltenen Bibliotheken definierten Methoden. Mit den folgenden Strategien können Sie vermeiden, die Dex-Referenzgrenze zu erreichen:

- **Überprüfen Sie die direkten und transitiven Abhängigkeiten Ihrer App** - Stellen Sie sicher, dass die **Abhängigkeit** einer großen Bibliothek, die Sie in Ihre App aufnehmen, auf eine Weise verwendet wird, die die Menge des der Anwendung hinzugefügten Codes überwiegt. Ein häufiges Anti-Pattern ist das Einschließen einer sehr großen Bibliothek, da einige Hilfsmethoden nützlich waren. Durch das Reduzieren der Abhängigkeiten Ihres App-Codes können Sie häufig die Dex-Referenzgrenze vermeiden.
- **Ungenutzten Code mit ProGuard entfernen** - Konfigurieren Sie die ProGuard-Einstellungen für Ihre App, um ProGuard auszuführen, und stellen Sie sicher, dass die Verkleinerung für Release-Builds aktiviert ist. Durch das Aktivieren der Verkleinerung wird sichergestellt, dass Sie nicht verwendeten Code mit Ihren APKs versenden.

Der erste Punkt erfordert Sorgfalt und Disziplin des Entwicklers. Bei der Einbindung von Bibliotheken von Drittanbietern muss die Größe der Bibliothek berücksichtigt werden. Zum Beispiel sind zwei beliebte JSON-Bibliotheken Jackson und Gson. Funktionell sind sie sich ziemlich ähnlich, aber Gson sieht in Android einen stärkeren Einsatz. Ein Grund ist, dass Jackson rund 9.000 Methoden wiegt, während Gson 1.900 beiträgt.

Es gibt verschiedene Tools, mit denen Entwickler die Größe ihrer Anwendung nachverfolgen können:

- dexcount-gradle-plugin meldet die Anzahl der Methodenverweise in Ihrem APK oder AAR in jedem Build
- dex-method-count ist ein Befehlszeilentool, das die Anzahl der Methodenverweise in einer APK zählt
- www.methodscount.com ist ein Webservice, der die Methodenreferenzen in jeder APK zählt, die Sie hochladen.

Examples

Multidex durch direkte Verwendung von MultiDexApplication

Verwenden Sie diese Option, wenn Sie keine `Application` benötigen.

Dies ist die einfachste Option, aber auf diese Weise können Sie keine eigene `Application` bereitstellen. Wenn eine `Application` benötigt wird, müssen Sie zu einer der anderen Optionen wechseln.

`android.support.multidex.MultiDexApplication` für diese Option einfach den vollqualifizierten Klassennamen `android.support.multidex.MultiDexApplication` für die Eigenschaft `android:name` des `application` in AndroidManifest.xml an:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.multidex.myapplication">
    <application
        ...
        android:name="android.support.multidex.MultiDexApplication">
        ...
    </application>
</manifest>
```

Multidex durch Erweiterung der Anwendung

Verwenden Sie diese Option, wenn für Ihr Projekt eine `Application` erforderlich ist.

Geben Sie diese `Application` mithilfe der `android:name` -Eigenschaft in der Manifestdatei innerhalb des `application` .

`attachBaseContext()` in der `Application` die `attachBaseContext()` Methode außer Kraft, und rufen `MultiDex.install()` in dieser Methode `MultiDex.install()` :

```
package com.example;

import android.app.Application;
import android.content.Context;

/**
 * Extended application that support multidex
 */
public class MyApplication extends Application {

    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        MultiDex.install(this);
    }
}
```

Stellen Sie sicher, dass die `Application` im `application` Ihrer AndroidManifest.xml angegeben ist:

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

Aktivieren von Multidex

Um eine Multidex-Konfiguration zu ermöglichen, benötigen Sie:

- um die Gradle Build-Konfiguration zu ändern
- Verwenden Sie eine `MultiDexApplication` oder aktivieren Sie die MultiDex in Ihrer `Application` Klasse

Gradle Konfiguration

In `app/build.gradle` fügen Sie diese Teile hinzu:

```
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'
}
```

Aktivieren Sie MultiDex in Ihrer Anwendung

Fahren Sie dann mit einer von drei Optionen fort:

- Multidex durch Erweiterung der Anwendung
- Multidex durch Erweiterung von MultiDexApplication
- Multidex durch direkte Verwendung von MultiDexApplication

Wenn diese Konfigurationseinstellungen einer App hinzugefügt werden, erstellen die Android-Build-Tools nach Bedarf einen primären Dex (classes.dex) und Unterstützung (classes2.dex, classes3.dex).
Das Build-System packt sie dann zur Verteilung in eine APK-Datei.

Zählmethodenreferenzen für jedes Build (Dexcount Gradle Plugin)

Das dexcount-Plugin zählt Methoden und Klassenressourcen nach einem erfolgreichen Build.

Fügen Sie das Plugin im `app/build.gradle` :

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

buildscript {
    repositories {
        mavenCentral() // or jcenter()
    }

    dependencies {
        classpath 'com.getkeepsafe.dexcount:dexcount-gradle-plugin:0.5.5'
    }
}
```

Wenden Sie das Plugin in der Datei `app/build.gradle` :

```
apply plugin: 'com.getkeepsafe.dexcount'
```

Suchen Sie nach den vom Plugin generierten Ausgabedaten:

../app/build/outputs/dexcount

Besonders nützlich ist das .html-Diagramm in:

../app/build/outputs/dexcount/debugChart/index.html

Multidex durch Erweiterung von MultiDexApplication

Dies ist sehr ähnlich der Verwendung einer `Application` und dem Überschreiben der `attachBaseContext()` Methode.

Bei Verwendung dieser Methode müssen Sie `attachBaseContext()` jedoch nicht überschreiben, da dies bereits in der `MultiDexApplication` Superklasse geschieht.

Erweitern Sie `MultiDexApplication` anstelle von `Application` :

```
package com.example;

import android.support.multidex.MultiDexApplication;
import android.content.Context;

/**
 * Extended MultiDexApplication
 */
public class MyApplication extends MultiDexApplication {

     // No need to override attachBaseContext()

     //.........
}
```

Fügen Sie diese Klasse genau zu Ihrer AndroidManifest.xml hinzu, als ob Sie Application erweitern würden:

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

Multidex- und Dex-Methodenlimit online lesen:
https://riptutorial.com/de/android/topic/1887/multidex--und-dex-methodenlimit

Kapitel 161: MVP-Architektur

Einführung

In diesem Thema wird die MVP- Architektur (Model-View-Presenter) von Android mit verschiedenen Beispielen beschrieben.

Bemerkungen

Es gibt viele Möglichkeiten, eine Android-App zu erstellen. Aber nicht alle sind testbar und ermöglichen es uns, unseren Code so zu strukturieren, dass die App leicht zu testen ist. Die Schlüsselidee einer testbaren Architektur besteht darin, Teile der Anwendung voneinander zu trennen, wodurch sie einfacher zu warten, zu erweitern und zu testen sind.

MVP-Definition

Modell

In einer Anwendung mit einer guten mehrschichtigen Architektur wäre dieses Modell nur das Gateway zur Domänenschicht oder Geschäftslogik. Sehen Sie es als den Anbieter der Daten, die wir in der Ansicht anzeigen möchten.

Aussicht

Die Ansicht, die normalerweise von einer `Activity` oder einem `Fragment` implementiert wird, enthält einen Verweis auf den *Präsentator* . Das einzige, was die Ansicht tun wird, ist, eine Methode aus dem Presenter aufzurufen, wenn eine Schnittstellenaktion vorliegt.

Moderator

Der Moderator ist dafür verantwortlich, als Mittler zwischen Ansicht und Modell zu fungieren. Er ruft Daten aus dem Modell ab und gibt sie formatiert an die Ansicht zurück. Im Gegensatz zum typischen MVC entscheidet es auch, was passiert, wenn Sie mit der Ansicht interagieren.

* Definitionen aus Antonio Leivas Artikel.

Empfohlene App-Struktur (nicht erforderlich)

Die App sollte nach Paket *pro Feature* strukturiert sein. Dies verbessert die Lesbarkeit und moduliert die App so, dass Teile davon unabhängig voneinander geändert werden können. Jede Schlüsselfunktion der App befindet sich in einem eigenen Java-Paket.

Examples

Anmeldebeispiel für das Model View Presenter (MVP) -Muster

Lassen Sie uns MVP in Aktion mit einem einfachen Anmeldebildschirm sehen. Es gibt zwei `Button` - eine für Anmeldeaktionen und eine weitere für einen Registrierungsbildschirm. zwei `EditText` - `EditText` - eine für die E-Mail und die andere für das Passwort.

LoginFragment (Die Ansicht)

```
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 (Der Moderator)

```
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 (Das Modell)

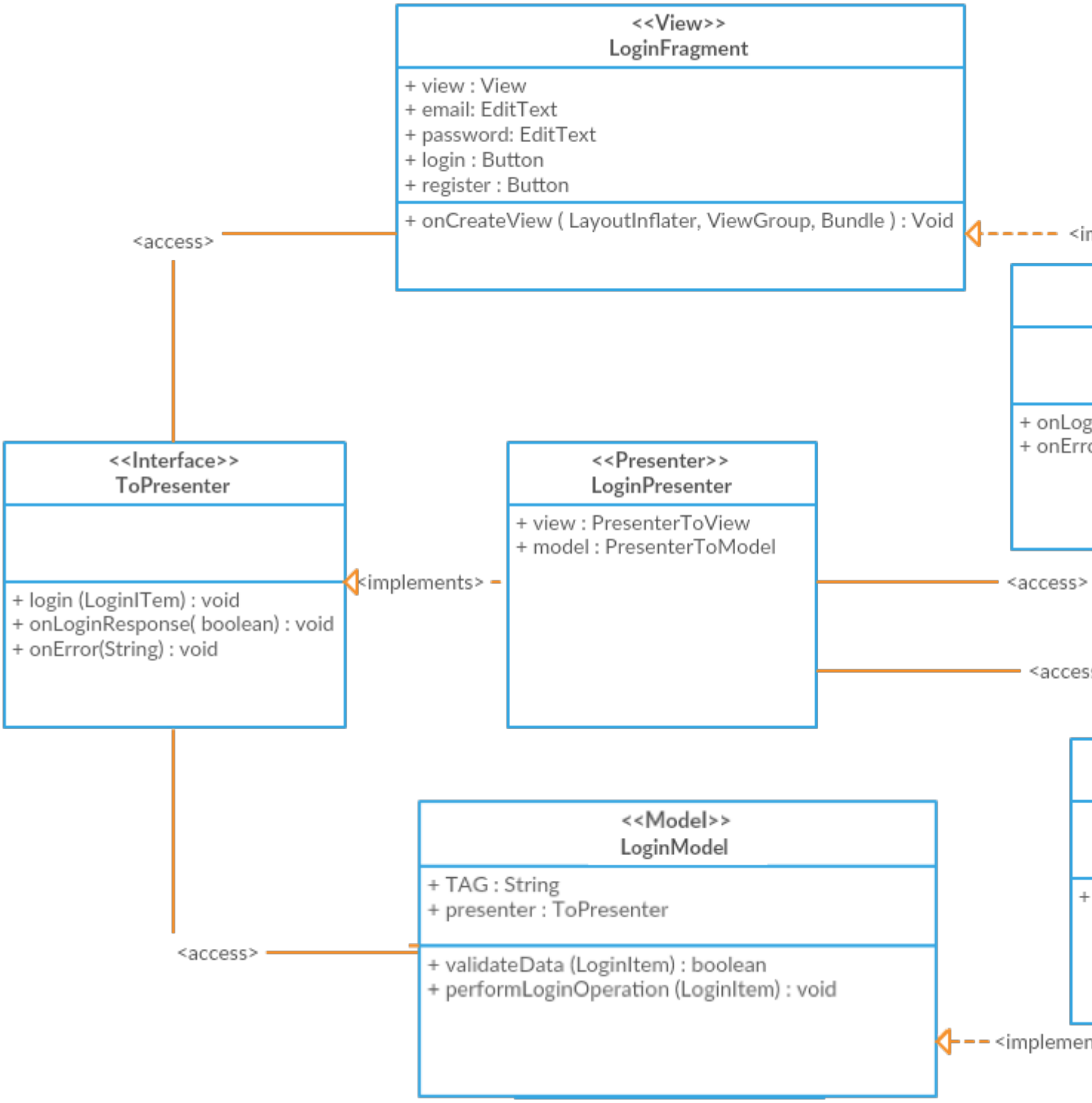
```
public class LoginModel implements LoginContract.PresenterToModel,
```

```
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);
    }
}
```

Klassen Diagramm

Lassen Sie uns die Aktion in Form eines Klassendiagramms sehen.

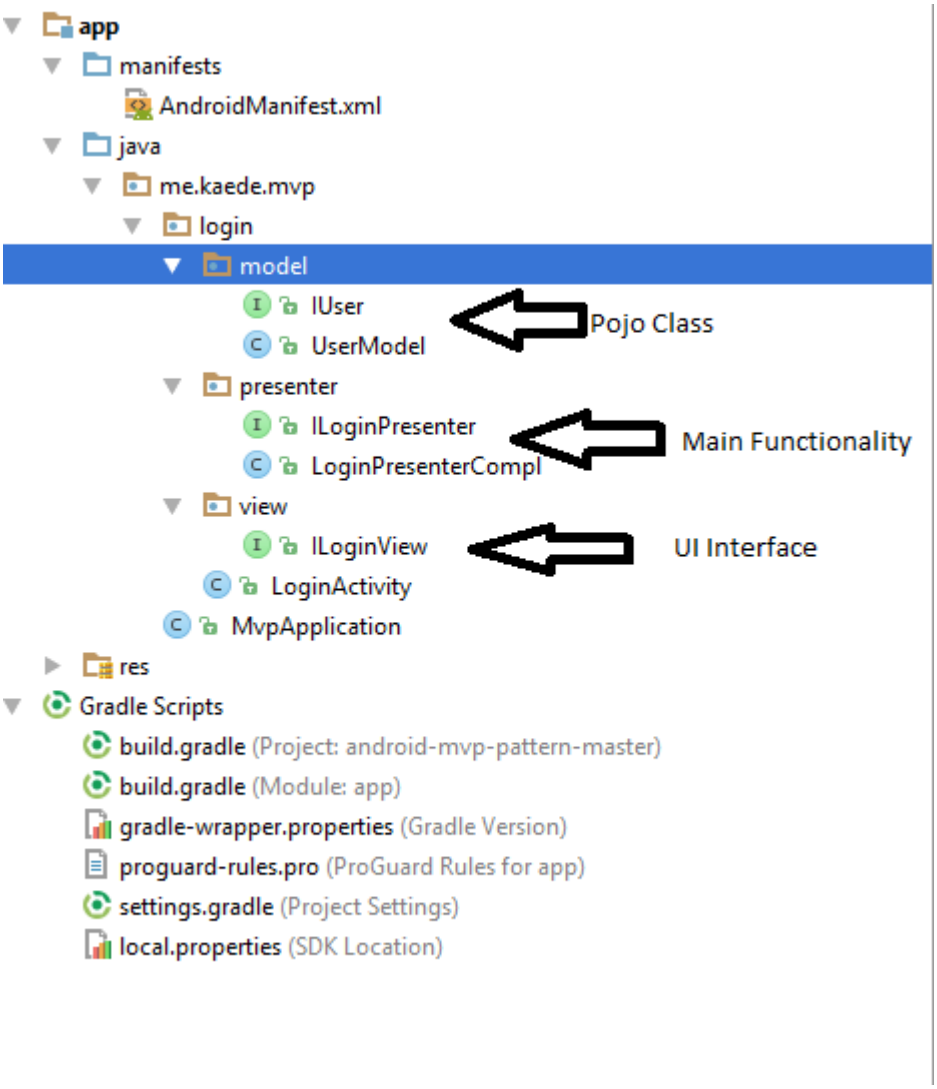


Anmerkungen:

- In diesem Beispiel wird Volley für die Netzwerkkommunikation verwendet. Diese Bibliothek ist jedoch für MVP nicht erforderlich
- `UrlUtils` ist eine Klasse, die alle Links für meine API-Endpunkte enthält
- `ResponseErrorListener.ErrorListener` ist eine `interface` , die auf Fehler in `ErrorResponse` , `implements` Volleys `Response.ErrorListener implements` . Diese Klassen sind hier nicht enthalten, da sie nicht direkt Teil dieses Beispiels sind

Einfaches Login-Beispiel in MVP

Erforderliche Paketstruktur



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>
```

Aktivitätsklasse 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);
    }
}
```

Erstellen einer ILoginView-Schnittstelle

Erstellen Sie eine `ILoginView` Schnittstelle für Update-Informationen aus dem Presenter-Ordner "Ordner" wie folgt:

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

Erstellen einer ILoginPresenter-Schnittstelle

Erstellen Sie eine `ILoginPresenter` Schnittstelle, um mit `LoginActivity` (Views) zu kommunizieren, und erstellen Sie die `LoginPresenterCompl` Klasse zum `LoginPresenterCompl` der `LoginPresenterCompl` und zum `LoginPresenterCompl` an die Aktivität. Die `LoginPresenterCompl` Klasse implementiert die `ILoginPresenter` Schnittstelle:

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");
    }
}
```

Ein UserModel erstellen

Erstellen Sie ein `UserModel` das einer `UserModel` Klasse für `LoginActivity` . Erstellen Sie eine `IUser` Schnittstelle für Pojo-Validierungen:

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;
}
```

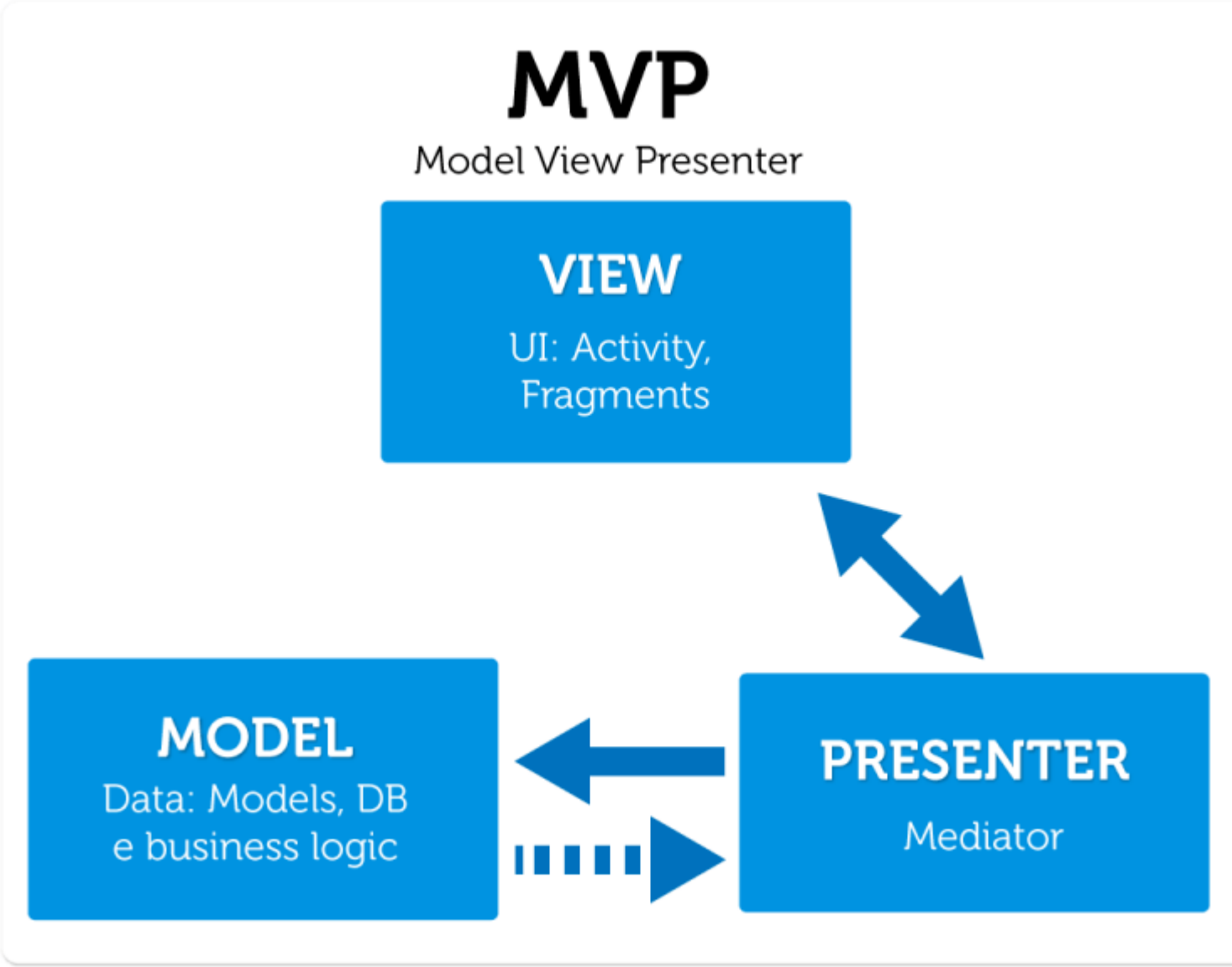
IUser.class

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

    String getPasswd();

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

MVP

Ein Model-View-Presenter (MVP) ist eine Ableitung des MVC-Architekturmusters (Model-View-Controller). Es wird hauptsächlich zum Erstellen von Benutzeroberflächen verwendet und bietet die folgenden Vorteile:

- Ansichten sind mehr von Modellen getrennt. Der Presenter ist der Vermittler zwischen Model und View.
- Es ist einfacher, Komponententests zu erstellen.
- Im Allgemeinen gibt es eine Eins-zu-Eins-Zuordnung zwischen Ansicht und Präsentator, mit der Möglichkeit, mehrere Präsentatoren für komplexe Ansichten zu verwenden.



MVP-Architektur online lesen: https://riptutorial.com/de/android/topic/4615/mvp-architektur

Kapitel 162: MVVM (Architektur)

Bemerkungen

Syntax-Macken mit DataBinding

Beim Binden einer viewModel-Funktion an eine Eigenschaft in xml werden bestimmte Funktionspräfixe wie `get` oder `is` gelöscht. Z.B. `ViewModel::getFormattedText` auf dem ViewModel wird zu `@{viewModel.formattedText}` wenn es an eine Eigenschaft in XML gebunden wird. Ähnlich bei `ViewModel::isContentVisible` -> `@{viewModel.contentVisible}` (Java Bean-Notation)

Die generierten Bindungsklassen wie `ActivityMainBinding` sind nach der XML-Datei benannt, für die sie Bindungen erstellen, nicht nach der Java-Klasse.

Kundenspezifische Bindungen

In der activity_main.xml habe ich das textColor-Attribut für die `app` und nicht den `android` Namespace festgelegt. Warum das? Weil für das Attribut `textColor` ein benutzerdefinierter Setter definiert ist, der eine vom ViewModel `textColor` ColorRes-Ressourcen-ID in eine tatsächliche Farbe auflöst.

```
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);
    }

}
```

Einzelheiten zur Funktionsweise von DataBinding Library: Custom Setters finden Sie hier

Warten Sie ... es gibt Logik in Ihrer XML !!!?

Sie könnten argumentieren, dass die Dinge, die ich in XML für `android:visibility` und `app:textColor` falsche / Anti-Muster im MVVM-Kontext sind, da in meiner Ansicht Ansichtslogik `app:textColor` . Ich würde jedoch argumentieren, dass es für mich wichtiger ist, Android-Abhängigkeiten zu Testzwecken aus meinem ViewModel herauszuhalten.

Was macht `app:textColor` wirklich? Es löst nur einen Ressourcenzeiger auf die tatsächlich zugeordnete Farbe auf. Das ViewModel entscheidet also immer noch, welche Farbe aufgrund bestimmter Bedingungen angezeigt wird.

Bei `android:visibility` empfinde ich aufgrund der Benennung der Methode als richtig, den ternären Operator hier zu verwenden. Aufgrund der Namen `isLoadingVisible` und `isContentVisible` besteht kein Zweifel daran, in was jedes Ergebnis in der Ansicht aufgelöst werden soll. Ich habe also das Gefühl, dass es eher einen Befehl ausführt, der vom ViewModel gegeben wird, als die View-Logik.

Andererseits stimme ich zu, dass `viewModel.isLoading ? View.VISIBLE : View.GONE` wäre eine schlechte Sache, weil Sie Annahmen in der Ansicht machen, was dieser Zustand für die Ansicht bedeutet.

Nützliches Material

Die folgenden Ressourcen haben mir sehr geholfen, dieses Konzept zu verstehen:

- Jeremy Likness - Model-View-ViewModel (MVVM) erklärt (C #) (08.2010)
- Shamlia Shukkur - Grundlagen des MVVM-Entwurfsmusters (C #) (03.2013)
- Frode Nilsen - Android Databinding: Auf Wiedersehen Presenter, Hallo ViewModel! (07.2015)
- Joe Birch - Mit Android auf MVVM (09.2015)
- Florina Muntenescu - Android Architekturmuster Teil 3: Modellansicht- AnsichtModell (10.2016)

Examples

MVVM-Beispiel mit DataBinding-Bibliothek

Der springende Punkt von MVVM besteht darin, Layer mit Logik von der Ansichtsebene zu trennen.

Auf Android können wir die DataBinding-Bibliothek verwenden , um uns dabei zu helfen und die meisten unserer Logikeinheitstests zu testen, ohne sich über Android-Abhängigkeiten Gedanken zu machen.

In diesem Beispiel zeige ich die zentralen Komponenten für eine dumme einfache App, die Folgendes ausführt:

- Beim Start fälschen Sie einen Netzwerkanruf und zeigen einen ladenden Spinner
- Zeigen Sie eine Ansicht mit einem Klickzähler TextView, einer Nachricht TextView und einer Schaltfläche zum Erhöhen des Zählers an
- Klicken Sie auf die Schaltfläche "Zähler aktualisieren" und aktualisieren Sie die Zählerfarbe und den Nachrichtentext, wenn der Zähler eine bestimmte Anzahl erreicht

Beginnen wir mit der Ansichtsebene:

`activity_main.xml`

:

Wenn Sie mit der Funktionsweise von DataBinding nicht vertraut sind, sollten Sie sich wahrscheinlich zehn Minuten Zeit nehmen , um sich damit vertraut zu machen. Wie Sie sehen, sind alle Felder, die Sie normalerweise mit Setters aktualisieren würden, an Funktionen der viewModel-Variablen gebunden.

Wenn Sie eine Frage zu `android:visibility` oder `app:textColor` Eigenschaften haben, überprüfen Sie den Abschnitt "Anmerkungen".

```
<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>
```

Als nächstes die Modellebene. Hier habe ich:

- zwei Felder, die den Status der App darstellen
- Getter, um die Anzahl der Klicks und den Erregungszustand zu lesen
- eine Methode, um meine Klickanzahl zu erhöhen
- eine Methode zum Wiederherstellen eines früheren Zustands (wichtig für Orientierungsänderungen)

Auch definiere ich hier einen "Aufregungszustand", der von der Anzahl der Klicks abhängig ist. Dies wird später verwendet, um Farbe und Nachricht in der Ansicht zu aktualisieren.

Es ist wichtig zu beachten, dass im Modell keine Annahmen darüber gemacht werden, wie der Status für den Benutzer angezeigt werden kann!

`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;
        }
    }
}
```

Als nächstes das ViewModel.

Dadurch werden Änderungen am Modell und an den Formatdaten des Modells ausgelöst, um sie

in der Ansicht anzuzeigen. Beachten Sie, dass hier ausgewertet wird, welche GUI-Darstellung für den vom Modell `resolveCounterColor` `resolveLabelText` geeignet ist (`resolveCounterColor` und `resolveLabelText`). So könnten wir beispielsweise problemlos ein `UnderachieverClickerModel` implementieren, das niedrigere Schwellenwerte für den Erregungszustand aufweist, ohne Code im viewModel oder in der Ansicht zu berühren.

Beachten Sie auch, dass das ViewModel keine Referenzen auf Ansichtsobjekte enthält. Alle Eigenschaften werden über die Annotationen `@Bindable` gebunden und aktualisiert, wenn entweder `notifyChange()` (signalisiert, dass alle Eigenschaften aktualisiert werden müssen) oder `notifyPropertyChanged(BR.propertyName)` (bedeutet, dass diese Eigenschaften aktualisiert werden müssen).

`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;
        }
    }

}
```

Alles in der Aktivität zusammen binden!

Hier sehen wir die Ansicht, die das viewModel mit allen Abhängigkeiten initialisiert, die es möglicherweise benötigt, die aus einem Android-Kontext instanziiert werden müssen.

Nachdem das viewModel initialisiert wurde, wird es über das DataBindingUtil an das XML-Layout gebunden. (Benennen Sie die generierten Klassen bitte im Abschnitt "Syntax").

Hinweis Abonnements werden auf dieser Ebene abonniert, da das Abbestellen der Abonnements erforderlich ist, wenn die Aktivität angehalten oder zerstört wird, um Speicherlecks und NPEs zu vermeiden. Hier wird auch das persistierende und erneute Laden des viewState bei OrientationChanges ausgelöst

`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;
    }
}
```

Um alles in Aktion zu sehen, schauen Sie sich dieses Beispielprojekt an .

MVVM (Architektur) online lesen: https://riptutorial.com/de/android/topic/7549/mvvm--architektur-

Kapitel 163: Nachrüstung2

Einführung

Die offizielle Retrofit-Seite beschreibt sich selbst als

Ein typsicherer REST-Client für Android und Java.

Retrofit macht aus Ihrer REST-API eine Java-Schnittstelle. Es verwendet Anmerkungen, um HTTP-Anforderungen zu beschreiben, die Ersetzung von URL-Parametern und die Unterstützung von Abfrageparametern ist standardmäßig integriert. Darüber hinaus bietet es Funktionen für mehrteilige Anforderungshauptteile und Dateiuploads.

Bemerkungen

Abhängigkeiten für die Retrofit-Bibliothek:

Aus der offiziellen Dokumentation :

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

Eine einfache GET-Anfrage

Wir zeigen Ihnen, wie Sie eine `GET` Anfrage an eine API richten, die mit einem `JSON` Objekt oder einem `JSON` Array antwortet. Als Erstes müssen wir die Abhängigkeiten von Retrofit und `GSON` Converter zur Gradle-Datei unseres Moduls hinzufügen.

Fügen Sie die Abhängigkeiten für die Retrofit-Bibliothek hinzu, wie im Abschnitt Anmerkungen beschrieben.

Beispiel eines erwarteten JSON-Objekts:

```
{
    "deviceId": "56V56C14SF5B4SF",
    "name": "Steven",
    "eventCount": 0
}
```

Beispiel eines JSON-Arrays:

```
[
    {
        "deviceId": "56V56C14SF5B4SF",
        "name": "Steven",
        "eventCount": 0
    },
    {
        "deviceId": "35A80SF3QDV7M9F",
        "name": "John",
        "eventCount": 2
    }
]
```

Beispiel für eine entsprechende Modellklasse:

```
public class Device
{
    @SerializedName("deviceId")
    public String id;

    @SerializedName("name")
    public String name;

    @SerializedName("eventCount")
    public int eventCount;
}
```

Die `@SerializedName` Annotationen hier stammen aus der `GSON` Bibliothek und ermöglichen es uns, diese Klasse in `JSON` zu `serialize` und `deserialize` , wobei der serialisierte Name als Schlüssel verwendet wird. Jetzt können wir die Schnittstelle für die API erstellen, die die Daten tatsächlich vom Server abruft.

```
public interface DeviceAPI
{
    @GET("device/{deviceId}")
    Call<Device> getDevice (@Path("deviceId") String deviceID);

    @GET("devices")
    Call<List<Device>> getDevices();
}
```

Hier ist viel los in einer ziemlich kompakten Umgebung, also lassen Sie uns das aufteilen:

- Die `@GET` Annotation stammt aus Retrofit und teilt der Bibliothek mit, dass wir eine GET-

Anforderung definieren.
- Der Pfad in den Klammern ist der Endpunkt, den unsere GET-Anfrage treffen soll (wir setzen die Basis-URL etwas später).
- Die geschweiften Klammern ermöglichen es uns, Teile des Pfads zur Laufzeit zu ersetzen, damit wir Argumente übergeben können.
- Die Funktion, die wir definieren, heißt `getDevice` und nimmt die `getDevice` Geräte-ID als Argument an.
- Die `@PATH` Annotation teilt Retrofit mit, dass dieses Argument den Platzhalter "deviceId" im Pfad ersetzen soll.
- Die Funktion gibt ein `Call` Objekt vom Typ `Device` .

Erstellen einer Wrapper-Klasse:

Jetzt werden wir eine kleine Wrapper-Klasse für unsere API erstellen, um den Retrofit-Initialisierungscode gut verpackt zu halten.

```
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);
    }
}
```

Diese Klasse erstellt eine GSON-Instanz, um die JSON-Antwort analysieren zu können, erstellt eine Retrofit-Instanz mit unserer Basis-URL und einem GSONConverter und erstellt dann eine Instanz unserer API.

Aufruf der API:

```
// Getting a JSON object
Call<Device> callObject = api.getDevice(deviceID);
callObject.enqueue(new Callback<Response<Device>>()
{
    @Override
    public void onResponse (Call<Device> call, Response<Device> response)
    {
        if (response.isSuccessful())
        {
           Device device = response.body();
        }
    }

    @Override
    public void onFailure (Call<Device> call, Throwable t)
    {
```

```
        Log.e(TAG, t.getLocalizedMessage());
    }
});

// Getting a JSON array
Call<List<Device>> callArray = api.getDevices();
callArray.enqueue(new Callback<Response<List<Device>>()
{
    @Override
    public void onResponse (Call<List<Device>> call, Response<List<Device>> response)
    {
        if (response.isSuccessful())
        {
           List<Device> devices = response.body();
        }
    }

    @Override
    public void onFailure (Call<List<Device>> call, Throwable t)
    {
        Log.e(TAG, t.getLocalizedMessage());
    }
});
```

Diese verwendet unsere API-Schnittstelle, um ein `Call<Device>` -Objekt bzw. ein `Call<List<Device>>` `Call<Device>` zu erstellen. Beim Aufruf von `enqueue` wird Retrofit `enqueue` , diesen Aufruf in einem Hintergrundthread `enqueue` und das Ergebnis an den hier erstellten Rückruf zurückzugeben.

Anmerkung: Das Parsing eines JSON-Arrays aus primitiven Objekten (wie *String, Integer, Boolean* und *Double*) ähnelt dem Parsing eines JSON-Arrays. Sie benötigen jedoch keine eigene Modellklasse. Sie können das Array von Strings beispielsweise abrufen, indem `Call<List<String>>` den Rückgabetyp des Anrufs als `Call<List<String>>` .

Protokollierung zu Retrofit2 hinzufügen

Nachrüstaufträge können mit einem Interceptor protokolliert werden. Es stehen verschiedene Detailebenen zur Verfügung: KEINE, BASIC, HEADERS, BODY. Siehe das Github-Projekt hier .

1. Fügen Sie Abhängigkeit zu build.gradle hinzu:

```
compile 'com.squareup.okhttp3:logging-interceptor:3.8.1'
```

2. Fügen Sie beim Erstellen von Retrofit Protokollierungs-Interceptor hinzu:

```
  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();
```

Das Freigeben der Protokolle im Terminal (Android Monitor) sollte in der Release-Version vermieden werden, da dies zu unerwünschtem Freigeben wichtiger Informationen wie Auth-Token usw. führen kann

Überprüfen Sie die folgende Bedingung, um zu vermeiden, dass die Protokolle zur Laufzeit angezeigt werden

```
if(BuildConfig.DEBUG){
     //your interfector code here
    }
```

Zum Beispiel:

```
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();
```

Hochladen einer Datei über Multipart

Deklarieren Sie Ihre Schnittstelle mit Retrofit2-Anmerkungen:

```
public interface BackendApiClient {
    @Multipart
    @POST("/uploadFile")
    Call<RestApiDefaultResponse> uploadPhoto(@Part("file\"; filename=\"photo.jpg\" ")
RequestBody photo);
}
```

Dabei ist `RestApiDefaultResponse` eine benutzerdefinierte Klasse, die die Antwort enthält.

Aufbau der Implementierung Ihrer API und Aufnahme des Aufrufs:

```
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>);
```

Nachrüstung mit OkHttp-Interceptor

Dieses Beispiel zeigt, wie ein Anforderungsinterceptor mit OkHttp verwendet wird. Dies hat zahlreiche Anwendungsfälle wie:

- Universellen `header` zur Anfrage hinzufügen. ZB eine Anfrage authentifizieren
- Debuggen von vernetzten Anwendungen
- Rohe `response` abrufen
- Protokollierung der Netzwerktransaktion usw.
- Legen Sie einen benutzerdefinierten Benutzeragenten fest

```
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);
```

Weitere Informationen finden Sie im OkHttp-Thema .

Header und Body: ein Authentifizierungsbeispiel

Die Annotationen `@Header` und `@Body` können in die Methodensignaturen `@Body` werden, und Retrofit erstellt diese automatisch basierend auf Ihren Modellen.

```
public interface MyService {
     @POST("authentication/user")
     Call<AuthenticationResponse> authenticateUser(@Body AuthenticationRequest request,
@Header("Authorization") String basicToken);
}
```

AuthenticaionRequest ist unser Modell, ein POJO, das die Informationen enthält, die der Server benötigt. Für dieses Beispiel möchte unser Server den Clientschlüssel und das Geheimnis.

```
public class AuthenticationRequest {
```

```
     String clientKey;
     String clientSecret;
}
```

Beachten Sie, dass wir in `@Header("Authorization")` dass der Authorization-Header `@Header("Authorization")` . Die anderen Header werden automatisch aufgefüllt, da Retrofit auf die Art der Objekte schließen kann, die wir als Gegenleistung senden und erwarten.

Wir schaffen irgendwo unseren Retrofit-Service. Wir stellen sicher, dass Sie HTTPS verwenden.

```
Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https:// some example site")
            .client(client)
            .build();
MyService myService = retrofit.create(MyService.class)
```

Dann können wir unseren Service nutzen.

```
AuthenticationRequest request = new AuthenticationRequest();
request.setClientKey(getClientKey());
request.setClientSecret(getClientSecret());
String basicToken = "Basic " + token;
myService.authenticateUser(request, basicToken);
```

Laden Sie mehrere Dateien mit Retrofit als Multipart hoch

Nachdem Sie die Retrofit-Umgebung in Ihrem Projekt eingerichtet haben, können Sie das folgende Beispiel verwenden, das das Hochladen mehrerer Dateien mit Retrofit veranschaulicht:

```
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;
}
```

Folgendes ist die Schnittstelle

```
public interface WebAPIService {
    @Multipart
    @POST("main.php")
    Call<String> postFile(@PartMap Map<String,RequestBody> Files, @Part("json") String
description);
}
```

Laden Sie eine Datei mit Retrofit2 vom Server herunter

Schnittstellendeklaration zum Herunterladen einer Datei

```
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);
}
```

Die Option 1 wird zum Herunterladen einer Datei vom Server mit fester URL verwendet. Option 2 wird verwendet, um einen dynamischen Wert als vollständige URL an den Anforderungsaufruf zu übergeben. Dies kann beim Herunterladen von Dateien hilfreich sein, die von Parametern wie Benutzer oder Zeit abhängen.

Setup-Nachrüstung für das Aufrufen von APIs

```
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);
    }

}
```

Nehmen Sie nun die Implementierung von API zum Herunterladen der Datei vom Server vor

```
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) {

            }
        });
    }
```

Nachdem Sie im Rückruf eine Antwort erhalten haben, codieren Sie einige Standard-E / A, um die Datei auf der Festplatte zu speichern. Hier ist der Code:

```
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;
        }
    }
```

Hinweis: Wir haben **ResponseBody** als Rückgabetyp angegeben. Andernfalls versucht Retrofit, den Code zu analysieren und zu konvertieren.

Wenn Sie mehr über Retrofit erfahren möchten, besuchen Sie diesen Link, da er sehr nützlich ist. [1]: https://futurestud.io/blog/retrofit-getting-started-and-android-client

Debuggen mit Stetho

Fügen Sie Ihrer Anwendung die folgenden Abhängigkeiten hinzu.

```
compile 'com.facebook.stetho:stetho:1.5.0'
```

```
compile 'com.facebook.stetho:stetho-okhttp3:1.5.0'
```

Rufen Sie in der `onCreate` Methode Ihrer Application-Klasse Folgendes auf.

```
Stetho.initializeWithDefaults(this);
```

Erstellen Sie beim Erstellen Ihrer `Retrofit` Instanz eine benutzerdefinierte OkHttp-Instanz.

```
OkHttpClient.Builder clientBuilder = new OkHttpClient.Builder();
clientBuilder.addNetworkInterceptor(new StethoInterceptor());
```

Legen Sie dann diese benutzerdefinierte OkHttp-Instanz in der Retrofit-Instanz fest.

```
Retrofit retrofit = new Retrofit.Builder()
    // ...
    .client(clientBuilder.build())
    .build();
```

Verbinden Sie jetzt Ihr Telefon mit Ihrem Computer, starten Sie die App und geben Sie `chrome://inspect` in Ihren Chrome-Browser ein. Nachgerüstete Netzwerkanrufe sollten jetzt angezeigt werden.

Retrofit 2 Custom Xml Converter

Hinzufügen von Abhängigkeiten zur build.gradle-Datei.

```
dependencies {
    ....
    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile ('com.thoughtworks.xstream:xstream:1.4.7') {
        exclude group: 'xmlpull', module: 'xmlpull'
    }
    ....
}
```

Dann erstellen Sie Converter Factory

```
public class XStreamXmlConverterFactory extends Converter.Factory {

    /** Create an instance using a default {@link com.thoughtworks.xstream.XStream} instance
for conversion. */
    public static XStreamXmlConverterFactory create() {
        return create(new XStream());
    }

    /** Create an instance using {@code xStream} for conversion. */
    public static XStreamXmlConverterFactory create(XStream xStream) {
        return new XStreamXmlConverterFactory(xStream);
    }

    private final XStream xStream;

    private XStreamXmlConverterFactory(XStream xStream) {
```

```
        if (xStream == null) throw new NullPointerException("xStream == null");
        this.xStream = xStream;
    }

    @Override
    public Converter<ResponseBody, ?> responseBodyConverter(Type type, Annotation[]
annotations, Retrofit retrofit) {

        if (!(type instanceof Class)) {
            return null;
        }

        Class<?> cls = (Class<?>) type;

        return new XStreamXmlResponseBodyConverter<>(cls, xStream);
    }

    @Override
    public Converter<?, RequestBody> requestBodyConverter(Type type,
          Annotation[] parameterAnnotations, Annotation[] methodAnnotations, Retrofit
retrofit) {

        if (!(type instanceof Class)) {
            return null;
        }

        return new XStreamXmlRequestBodyConverter<>(xStream);
    }
}
```

Erstellen Sie eine Klasse, um die Anfrage zu bearbeiten.

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

Erstellen Sie eine Klasse, um mit der Antwort des Körpers umzugehen.

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

An diesem Punkt können wir jedes XML senden und empfangen. Wir müssen lediglich XStream-Anmerkungen für die Entitäten erstellen.

Dann erstellen Sie eine Retrofit-Instanz:

```
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();
```

Eine einfache POST-Anfrage mit GSON

Beispiel JSON:

```
{
    "id": "12345",
    "type": "android"
}
```

Definieren Sie Ihre Anfrage:

```
public class GetDeviceRequest {
```

```
    @SerializedName("deviceId")
    private String mDeviceId;

    public GetDeviceRequest(String deviceId) {
        this.mDeviceId = deviceId;
    }

    public String getDeviceId() {
        return mDeviceId;
    }

}
```

Definieren Sie Ihren Service (Endpunkte für Treffer):

```
public interface Service {

    @POST("device")
    Call<Device> getDevice(@Body GetDeviceRequest getDeviceRequest);

}
```

Definieren Sie Ihre Singleton-Instanz des Netzwerkclients:

```
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;
    }

}
```

Definieren Sie ein einfaches Modellobjekt für das Gerät:

```
public class Device {

    @SerializedName("id")
```

```
    private String mId;

    @SerializedName("type")
    private String mType;

    public String getId() {
        return mId;
    }

    public String getType() {
        return mType;
    }

}
```

Definieren Sie einen Controller, um die Anforderungen für das Gerät zu bearbeiten

```
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
            }

        });
```

XML-Formular-URL mit Retrofit 2 lesen

Wir werden retrofit 2 und SimpleXmlConverter verwenden, um XML-Daten von der URL abzurufen und in die Java-Klasse zu analysieren.

Fügen Sie dem Gradle-Skript eine Abhängigkeit hinzu:

```
compile 'com.squareup.retrofit2:retrofit:2.1.0'
compile 'com.squareup.retrofit2:converter-simplexml:2.1.0'
```

Schnittstelle erstellen

Erstellen Sie in unserer Fall-Rss-Klasse auch einen XML-Klassen-Wrapper

```
    public interface ApiDataInterface{

        // path to xml link on web site

        @GET (data/read.xml)

        Call<Rss> getData();

}
```

XML-Lesefunktion

```
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());
        }


    }
```

Dies ist ein Beispiel für eine Java-Klasse mit SimpleXML-Anmerkungen

Weitere Informationen zu Anmerkungen 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;
```

Nachrüstung2 online lesen: https://riptutorial.com/de/android/topic/1132/nachrustung2

Kapitel 164: Navigationsansicht

Bemerkungen

Die **Navigationsansicht** stellt ein Standardnavigationsmenü für die Anwendung dar. Der Menüinhalt kann mit einer Menüressourcendatei gefüllt werden.

Bevor Sie die `NavigationView` , müssen Sie die Abhängigkeit der Entwurfsunterstützungsbibliothek in der Datei build.gradle hinzufügen:

```
 dependencies {
    compile 'com.android.support:design:24.2.0'
}
```

Offizielle Dokumentation:

https://developer.android.com/reference/android/support/design/widget/NavigationView.html

Material Design Spezifikationen:

https://material.google.com/patterns/navigation-drawer.html#navigation-drawer-content

Examples

So fügen Sie die Navigationsansicht hinzu

Um eine Navigationsansicht zu verwenden, fügen Sie die Abhängigkeit in der Datei `build.gradle` , wie im Abschnitt "Anmerkungen" beschrieben

Fügen Sie dann die `NavigationView` in das Layout ein

```
<?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 : Die Ansicht, die oben in der Schublade angezeigt wird

```
<?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 Eine Abstraktionsebene für die Symbolleiste, um sie vom Inhalt zu trennen:

```
<?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 Der eigentliche Inhalt der Aktivität nur zur Demo, hier würden Sie Ihre normale Layout-XML einfügen:

```
<?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>
```

Definieren Sie Ihre Menüdatei als *res/menu/activity_main_drawer.xml* :

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

    <group android:checkableBehavior="single">
        <item
```

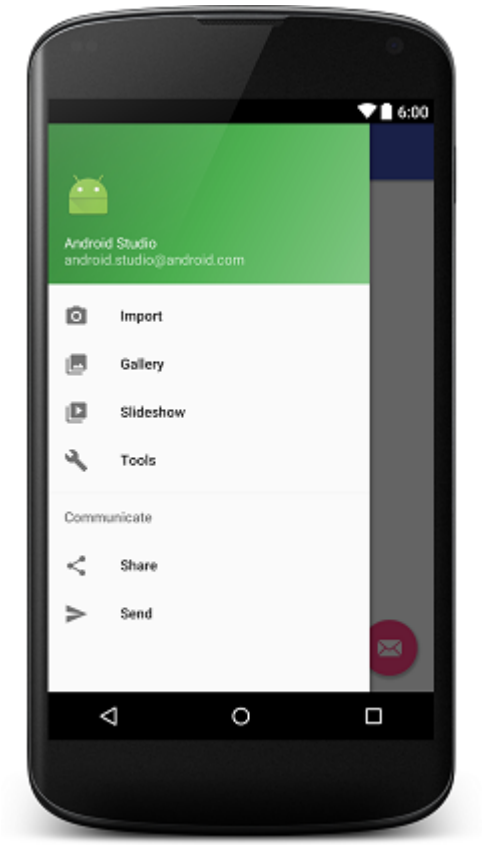
```
            android:id="@+id/nav_camera"
            android:icon="@drawable/ic_menu_camera"
            android:title="Import" />
        <item
            android:id="@+id/nav_gallery"
            android:icon="@drawable/ic_menu_gallery"
            android:title="Gallery" />
        <item
            android:id="@+id/nav_slideshow"
            android:icon="@drawable/ic_menu_slideshow"
            android:title="Slideshow" />
        <item
            android:id="@+id/nav_manage"
            android:icon="@drawable/ic_menu_manage"
            android:title="Tools" />
    </group>

    <item android:title="Communicate">
        <menu>
            <item
                android:id="@+id/nav_share"
                android:icon="@drawable/ic_menu_share"
                android:title="Share" />
            <item
                android:id="@+id/nav_send"
                android:icon="@drawable/ic_menu_send"
                android:title="Send" />
        </menu>
    </item>

</menu>
```

Und zum Schluss noch das `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;
    }
}
```

Es wird so aussehen:

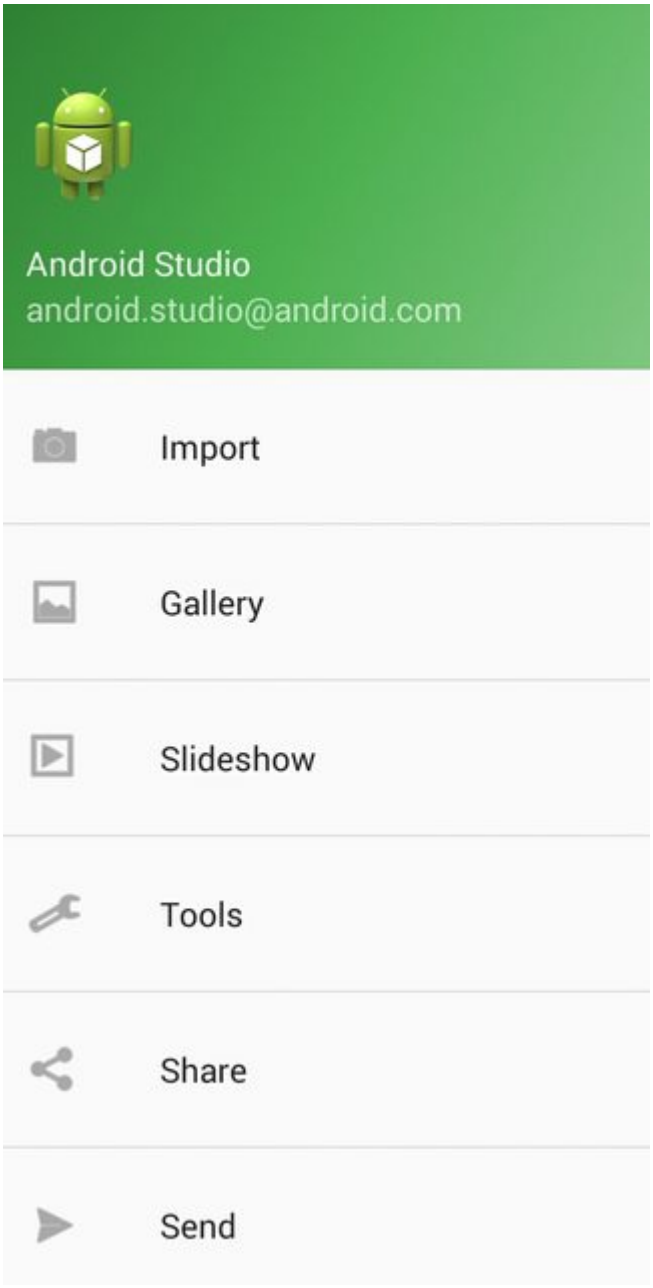


Unterstriche in Menüelementen hinzufügen

Jede Gruppe endet mit einem Zeilentrennzeichen. Wenn für jedes Element in Ihrem Menü eine eigene Gruppe vorhanden ist, erhalten Sie die gewünschte grafische Ausgabe. Es funktioniert nur, wenn Ihre verschiedenen Gruppen unterschiedliche `android:id` . `menu.xml` auch in `menu.xml` daran, `android:checkable="true"` zu erwähnen `android:checkable="true"` für einzelnes Element und `android:checkableBehavior="single"` für eine Elementgruppe.

```
<?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>
```



Fügen Sie dem Menü Separatoren hinzu

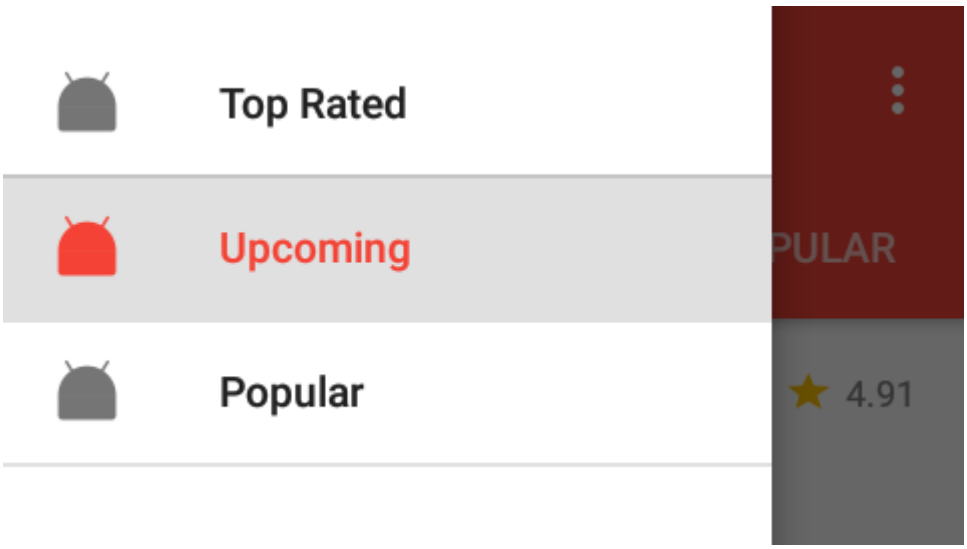
Rufen Sie die RecyclerView in der Navigationsansicht auf und fügen Sie ItemDecoration hinzu .

```
NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new DividerItemDecoration(this));
```

Code für DividerItemDecoration

```
public class DividerItemDecoration extends RecyclerView.ItemDecoration {

    private static final int[] ATTRS = new int[]{android.R.attr.listDivider};

    private Drawable mDivider;

    public DividerItemDecoration(Context context) {
        final TypedArray styledAttributes = context.obtainStyledAttributes(ATTRS);
        mDivider = styledAttributes.getDrawable(0);
        styledAttributes.recycle();
    }


    @Override
    public void onDraw(Canvas c, RecyclerView parent, RecyclerView.State state) {
        int left = parent.getPaddingLeft();
        int right = parent.getWidth() - parent.getPaddingRight();

        int childCount = parent.getChildCount();
        for (int i = 1; i < childCount; i++) {
            View child = parent.getChildAt(i);

            RecyclerView.LayoutParams params = (RecyclerView.LayoutParams)
child.getLayoutParams();

            int top = child.getBottom() + params.bottomMargin;
            int bottom = top + mDivider.getIntrinsicHeight();

            mDivider.setBounds(left, top, right, bottom);
            mDivider.draw(c);
        }
    }
}
```

Vorschau:



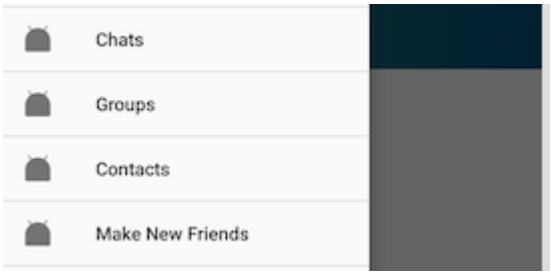
Menü hinzufügen Divider mit Standard DividerItemDecoration.

Verwenden Sie einfach die standardmäßige DividerItemDecoration-Klasse:

```
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation);
```

```
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new
DividerItemDecoration(context,DividerItemDecoration.VERTICAL));
```

Vorschau:



Navigationsansicht online lesen: https://riptutorial.com/de/android/topic/97/navigationsansicht

Kapitel 165: OkHttp

Examples

Abfangjäger protokollieren

`Interceptors` werden verwendet, um `OkHttp` Aufrufe abzufangen. Der Abfanggrund könnte darin bestehen, Anrufe zu überwachen, neu zu schreiben und erneut zu versuchen. Es kann sowohl für ausgehende Anforderungen als auch für eingehende Antworten verwendet werden.

```
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;
  }
}
```

Antworten umschreiben

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

Grundlegendes Anwendungsbeispiel

Ich `OkHttp` mein `OkHttp` in eine Klasse namens `HttpClient` , und in dieser Klasse habe ich Methoden für jedes der wichtigsten HTTP-Verben, `post` , `get` , `put` und `delete` , am häufigsten. (In der Regel füge ich eine Schnittstelle hinzu, um die Implementierung zu ermöglichen, um bei Bedarf problemlos zu einer anderen Implementierung wechseln zu können.)

```
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();
}
```

Die Syntax ist für `put` , `get` und `delete` identisch, mit Ausnahme von 1 Wort (`.put(body)`). `.put(body)` kann es `.put(body)` sein, auch diesen Code zu posten. Die Verwendung ist ziemlich einfach. Rufen Sie einfach die entsprechende Methode für einige `url` mit etwas `json` Payload auf. Die Methode gibt als Ergebnis eine Zeichenfolge zurück, die Sie später verwenden und analysieren können. Nehmen wir an , dass die Antwort sein , wird `json` , wir schaffen können `JSONObject` leicht davon:

```
String response = httpClient.post(MY_URL, JSON_PAYLOAD);
JSONObject json = new JSONObject(response);
// continue to parse the response according to it's structure
```

Synchroner Anruf abrufen

```
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());
  }
```

Asynchroner Anruf abrufen

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

Buchungsformularparameter

```
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());
  }
```

Buchung einer mehrteiligen Anfrage

```
 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 einrichten

Über Maven greifen:

```
<dependency>
  <groupId>com.squareup.okhttp3</groupId>
  <artifactId>okhttp</artifactId>
  <version>3.6.0</version>
</dependency>
```

oder Gradle:

```
compile 'com.squareup.okhttp3:okhttp:3.6.0'
```

OkHttp online lesen: https://riptutorial.com/de/android/topic/3625/okhttp

Kapitel 166: Okio

Examples

Herunterladen / Implementieren

Lade das neueste JAR herunter oder greife über Maven:

```
<dependency>
    <groupId>com.squareup.okio</groupId>
    <artifactId>okio</artifactId>
    <version>1.12.0</version>
</dependency>
```

oder Gradle:

```
compile 'com.squareup.okio:okio:1.12.0'
```

PNG-Decoder

Die Dekodierung der Chunks einer PNG-Datei zeigt Okio in der Praxis.

```
private static final ByteString PNG_HEADER = ByteString.decodeHex("89504e470d0a1a0a");

public void decodePng(InputStream in) throws IOException {
  try (BufferedSource pngSource = Okio.buffer(Okio.source(in))) {
    ByteString header = pngSource.readByteString(PNG_HEADER.size());
    if (!header.equals(PNG_HEADER)) {
      throw new IOException("Not a PNG.");
    }

    while (true) {
      Buffer chunk = new Buffer();

      // Each chunk is a length, type, data, and CRC offset.
      int length = pngSource.readInt();
      String type = pngSource.readUtf8(4);
      pngSource.readFully(chunk, length);
      int crc = pngSource.readInt();

      decodeChunk(type, chunk);
      if (type.equals("IEND")) break;
    }
  }
}

private void decodeChunk(String type, Buffer chunk) {
  if (type.equals("IHDR")) {
    int width = chunk.readInt();
    int height = chunk.readInt();
    System.out.printf("%08x: %s %d x %d%n", chunk.size(), type, width, height);
  } else {
    System.out.printf("%08x: %s%n", chunk.size(), type);
```

```
    }
}
```

ByteStrings und Puffer

ByteStrings und Puffer

Okio basiert auf zwei Typen, die viele Funktionen in eine unkomplizierte API integrieren:

ByteString ist eine unveränderliche Folge von Bytes. Für Zeichendaten ist String grundlegend. ByteString ist der lange verlorene Bruder von String, wodurch es einfach ist, binäre Daten als Wert zu behandeln. Diese Klasse ist ergonomisch: Sie kann sich selbst als Hex, Base64 und UTF-8 kodieren und dekodieren.

Puffer ist eine veränderliche Folge von Bytes. Wie bei ArrayList müssen Sie den Puffer nicht im Voraus anpassen. Sie lesen und schreiben Puffer als Warteschlange: Schreiben Sie die Daten an das Ende und lesen Sie sie von vorne. Es besteht keine Verpflichtung, Positionen, Limits oder Kapazitäten zu verwalten.

Intern tun `ByteString` und `Buffer` einige clevere Dinge, um CPU und Speicher zu sparen. Wenn Sie einen UTF-8-String als ByteString codieren, speichert er einen Verweis auf diesen String, sodass Sie beim späteren Dekodieren keine weiteren Schritte ausführen müssen.

`Buffer` wird als verknüpfte Liste von Segmenten implementiert. Wenn Sie Daten von einem Puffer in einen anderen verschieben, wird der Besitz der Segmente neu zugewiesen, anstatt die Daten zu kopieren. Dieser Ansatz ist besonders für Multithread-Programme hilfreich: Ein Thread, der mit dem Netzwerk kommuniziert, kann Daten mit einem Worker-Thread austauschen, ohne dass er kopiert oder zeremoniert werden muss.

Okio online lesen: https://riptutorial.com/de/android/topic/9952/okio

Kapitel 167: Optimiertes VideoView

Einführung

Die `VideoView` eines Videos mit einer `VideoView` die `SurfaceView` innerhalb einer Zeile einer `ListView` scheint zunächst zu funktionieren, bis der Benutzer versucht, die Liste zu scrollen. Sobald die Liste zu scrollen beginnt, wird das Video schwarz (manchmal weiß). Es wird weiterhin im Hintergrund abgespielt, kann jedoch nicht mehr angezeigt werden, da der Rest des Videos als Black Box dargestellt wird. Mit dem benutzerdefinierten optimierten VideoView werden die Videos wie in Instagram, Facebook und Twitter im `ListView` .

Examples

Optimierte VideoView in ListView

Dies ist das benutzerdefinierte `VideoView` , das Sie in Ihrem Paket benötigen.

Benutzerdefiniertes VideoView-Layout:

```
<your.packagename.VideoView
    android:id="@+id/video_view"
    android:layout_width="300dp"
    android:layout_height="300dp" />
```

Code für benutzerdefiniertes optimiertes `VideoView` :

```
package your.package.com.whateveritis;

import android.content.Context;
import android.content.Intent;
import android.graphics.SurfaceTexture;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.media.MediaPlayer.OnErrorListener;
import android.media.MediaPlayer.OnInfoListener;
import android.net.Uri;
import android.util.AttributeSet;
import android.util.Log;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.Surface;
import android.view.TextureView;
import android.view.View;
import android.widget.MediaController;
import android.widget.MediaController.MediaPlayerControl;

import java.io.IOException;


/**
```

```
 * VideoView is used to play video, just like
 * {@link android.widget.VideoView VideoView}. We define a custom view, because
 * we could not use {@link android.widget.VideoView VideoView} in ListView. <br/>
 * VideoViews inside ScrollViews do not scroll properly. Even if you use the
 * workaround to set the background color, the MediaController does not scroll
 * along with the VideoView. Also, the scrolling video looks horrendous with the
 * workaround, lots of flickering.
 *
 * @author leo
 */
public class VideoView extends TextureView implements MediaPlayerControl {

    private static final String TAG = "tag";

    // all possible internal states
    private static final int STATE_ERROR = -1;
    private static final int STATE_IDLE = 0;
    private static final int STATE_PREPARING = 1;
    private static final int STATE_PREPARED = 2;
    private static final int STATE_PLAYING = 3;
    private static final int STATE_PAUSED = 4;
    private static final int STATE_PLAYBACK_COMPLETED = 5;

    // currentState is a VideoView object's current state.
    // targetState is the state that a method caller intends to reach.
    // For instance, regardless the VideoView object's current state,
    // calling pause() intends to bring the object to a target state
    // of STATE_PAUSED.
    private int mCurrentState = STATE_IDLE;
    private int mTargetState = STATE_IDLE;

    // Stuff we need for playing and showing a video
    private MediaPlayer mMediaPlayer;
    private int mVideoWidth;
    private int mVideoHeight;
    private int mSurfaceWidth;
    private int mSurfaceHeight;
    private SurfaceTexture mSurfaceTexture;
    private Surface mSurface;
    private MediaController mMediaController;
    private MediaPlayer.OnCompletionListener mOnCompletionListener;
    private MediaPlayer.OnPreparedListener mOnPreparedListener;

    private MediaPlayer.OnErrorListener mOnErrorListener;
    private MediaPlayer.OnInfoListener mOnInfoListener;

    private int mSeekWhenPrepared; // recording the seek position while
    // preparing
    private int mCurrentBufferPercentage;
    private int mAudioSession;
    private Uri mUri;

    private Context mContext;

    public VideoView(final Context context) {
        super(context);
        mContext = context;
        initVideoView();
    }

    public VideoView(final Context context, final AttributeSet attrs) {
```

```
        super(context, attrs);
        mContext = context;
        initVideoView();
    }

    public VideoView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        mContext = context;
        initVideoView();
    }

    public void initVideoView() {
        mVideoHeight = 0;
        mVideoWidth = 0;
        setFocusable(false);
        setSurfaceTextureListener(mSurfaceTextureListener);
    }

    public int resolveAdjustedSize(int desiredSize, int measureSpec) {
        int result = desiredSize;
        int specMode = MeasureSpec.getMode(measureSpec);
        int specSize = MeasureSpec.getSize(measureSpec);

        switch (specMode) {
            case MeasureSpec.UNSPECIFIED:
            /*
             * Parent says we can be as big as we want. Just don't be larger
             * than max size imposed on ourselves.
             */
                result = desiredSize;
                break;

            case MeasureSpec.AT_MOST:
            /*
             * Parent says we can be as big as we want, up to specSize. Don't be
             * larger than specSize, and don't be larger than the max size
             * imposed on ourselves.
             */
                result = Math.min(desiredSize, specSize);
                break;

            case MeasureSpec.EXACTLY:
                // No choice. Do what we are told.
                result = specSize;
                break;
        }
        return result;
    }

    public void setVideoPath(String path) {
        Log.d(TAG, "Setting video path to: " + path);
        setVideoURI(Uri.parse(path));
    }

    public void setVideoURI(Uri _videoURI) {
        mUri = _videoURI;
        mSeekWhenPrepared = 0;
        requestLayout();
        invalidate();
        openVideo();
    }
```

```
public Uri getUri() {
    return mUri;
}

public void setSurfaceTexture(SurfaceTexture _surfaceTexture) {
    mSurfaceTexture = _surfaceTexture;
}

public void openVideo() {
    if ((mUri == null) || (mSurfaceTexture == null)) {
        Log.d(TAG, "Cannot open video, uri or surface texture is null.");
        return;
    }
    // Tell the music playback service to pause
    // TODO: these constants need to be published somewhere in the
    // framework.
    Intent i = new Intent("com.android.music.musicservicecommand");
    i.putExtra("command", "pause");
    mContext.sendBroadcast(i);
    release(false);
    try {
        mSurface = new Surface(mSurfaceTexture);
        mMediaPlayer = new MediaPlayer();
        if (mAudioSession != 0) {
            mMediaPlayer.setAudioSessionId(mAudioSession);
        } else {
            mAudioSession = mMediaPlayer.getAudioSessionId();
        }

        mMediaPlayer.setOnBufferingUpdateListener(mBufferingUpdateListener);
        mMediaPlayer.setOnCompletionListener(mCompleteListener);
        mMediaPlayer.setOnPreparedListener(mPreparedListener);
        mMediaPlayer.setOnErrorListener(mErrorListener);
        mMediaPlayer.setOnInfoListener(mOnInfoListener);
        mMediaPlayer.setOnVideoSizeChangedListener(mVideoSizeChangedListener);

        mMediaPlayer.setSurface(mSurface);
        mCurrentBufferPercentage = 0;
        mMediaPlayer.setDataSource(mContext, mUri);

        mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
        mMediaPlayer.setScreenOnWhilePlaying(true);

        mMediaPlayer.prepareAsync();
        mCurrentState = STATE_PREPARING;
    } catch (IllegalStateException e) {
        mCurrentState = STATE_ERROR;
        mTargetState = STATE_ERROR;
        String msg = (e.getMessage()==null)?"":e.getMessage();
        Log.i("",msg);  // TODO auto-generated catch block
    } catch (IOException e) {
        mCurrentState = STATE_ERROR;
        mTargetState = STATE_ERROR;
        String msg = (e.getMessage()==null)?"":e.getMessage();
        Log.i("",msg);  // TODO auto-generated catch block
    }
}

public void stopPlayback() {
    if (mMediaPlayer != null) {
```

```
            mMediaPlayer.stop();
            mMediaPlayer.release();
            mMediaPlayer = null;
            if (null != mMediaControllListener) {
                mMediaControllListener.onStop();
            }
        }
    }

    public void setMediaController(MediaController controller) {
        if (mMediaController != null) {
            mMediaController.hide();
        }
        mMediaController = controller;
        attachMediaController();
    }

    private void attachMediaController() {
        if (mMediaPlayer != null && mMediaController != null) {
            mMediaController.setMediaPlayer(this);
            View anchorView = this.getParent() instanceof View ? (View) this.getParent() :
this;
            mMediaController.setAnchorView(anchorView);
            mMediaController.setEnabled(isInPlaybackState());
        }
    }

    private void release(boolean cleartargetstate) {
        Log.d(TAG, "Releasing media player.");
        if (mMediaPlayer != null) {
            mMediaPlayer.reset();
            mMediaPlayer.release();
            mMediaPlayer = null;
            mCurrentState = STATE_IDLE;
            if (cleartargetstate) {
                mTargetState = STATE_IDLE;
            }
        } else {
            Log.d(TAG, "Media player was null, did not release.");
        }
    }

    @Override
    protected void onMeasure(final int widthMeasureSpec, final int heightMeasureSpec) {
        // Will resize the view if the video dimensions have been found.
        // video dimensions are found after onPrepared has been called by
        // MediaPlayer
        int width = getDefaultSize(mVideoWidth, widthMeasureSpec);
        int height = getDefaultSize(mVideoHeight, heightMeasureSpec);
        if ((mVideoWidth > 0) && (mVideoHeight > 0)) {
            if ((mVideoWidth * height) > (width * mVideoHeight)) {
                Log.d(TAG, "Video too tall, change size.");
                height = (width * mVideoHeight) / mVideoWidth;
            } else if ((mVideoWidth * height) < (width * mVideoHeight)) {
                Log.d(TAG, "Video too wide, change size.");
                width = (height * mVideoWidth) / mVideoHeight;
            } else {
                Log.d(TAG, "Aspect ratio is correct.");
            }
        }
        setMeasuredDimension(width, height);
```

```
    }

    @Override
    public boolean onTouchEvent(MotionEvent ev) {
        if (isInPlaybackState() && mMediaController != null) {
            toggleMediaControlsVisiblity();
        }
        return false;
    }

    @Override
    public boolean onTrackballEvent(MotionEvent ev) {
        if (isInPlaybackState() && mMediaController != null) {
            toggleMediaControlsVisiblity();
        }
        return false;
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        boolean isKeyCodeSupported = keyCode != KeyEvent.KEYCODE_BACK && keyCode !=
KeyEvent.KEYCODE_VOLUME_UP && keyCode != KeyEvent.KEYCODE_VOLUME_DOWN
                && keyCode != KeyEvent.KEYCODE_VOLUME_MUTE && keyCode != KeyEvent.KEYCODE_MENU
&& keyCode != KeyEvent.KEYCODE_CALL
                && keyCode != KeyEvent.KEYCODE_ENDCALL;
        if (isInPlaybackState() && isKeyCodeSupported && mMediaController != null) {
            if (keyCode == KeyEvent.KEYCODE_HEADSETHOOK || keyCode ==
KeyEvent.KEYCODE_MEDIA_PLAY_PAUSE) {
                if (mMediaPlayer.isPlaying()) {
                    pause();
                    mMediaController.show();
                } else {
                    start();
                    mMediaController.hide();
                }
                return true;
            } else if (keyCode == KeyEvent.KEYCODE_MEDIA_PLAY) {
                if (!mMediaPlayer.isPlaying()) {
                    start();
                    mMediaController.hide();
                }
                return true;
            } else if (keyCode == KeyEvent.KEYCODE_MEDIA_STOP || keyCode ==
KeyEvent.KEYCODE_MEDIA_PAUSE) {
                if (mMediaPlayer.isPlaying()) {
                    pause();
                    mMediaController.show();
                }
                return true;
            } else {
                toggleMediaControlsVisiblity();
            }
        }

        return super.onKeyDown(keyCode, event);
    }

    private void toggleMediaControlsVisiblity() {
        if (mMediaController.isShowing()) {
            mMediaController.hide();
        } else {
```

```
            mMediaController.show();
        }
    }

    public void start() {
        // This can potentially be called at several points, it will go through
        // when all conditions are ready
        // 1. When setting the video URI
        // 2. When the surface becomes available
        // 3. From the activity
        if (isInPlaybackState()) {
            mMediaPlayer.start();
            mCurrentState = STATE_PLAYING;
            if (null != mMediaControllListener) {
                mMediaControllListener.onStart();
            }
        } else {
            Log.d(TAG, "Could not start. Current state " + mCurrentState);
        }
        mTargetState = STATE_PLAYING;
    }

    public void pause() {
        if (isInPlaybackState()) {
            if (mMediaPlayer.isPlaying()) {
                mMediaPlayer.pause();
                mCurrentState = STATE_PAUSED;
                if (null != mMediaControllListener) {
                    mMediaControllListener.onPause();
                }
            }
        }
        mTargetState = STATE_PAUSED;
    }

    public void suspend() {
        release(false);
    }

    public void resume() {
        openVideo();
    }

    @Override
    public int getDuration() {
        if (isInPlaybackState()) {
            return mMediaPlayer.getDuration();
        }

        return -1;
    }

    @Override
    public int getCurrentPosition() {
        if (isInPlaybackState()) {
            return mMediaPlayer.getCurrentPosition();
        }
        return 0;
    }

    @Override
```

```
    public void seekTo(int msec) {
        if (isInPlaybackState()) {
            mMediaPlayer.seekTo(msec);
            mSeekWhenPrepared = 0;
        } else {
            mSeekWhenPrepared = msec;
        }
    }

    @Override
    public boolean isPlaying() {
        return isInPlaybackState() && mMediaPlayer.isPlaying();
    }

    @Override
    public int getBufferPercentage() {
        if (mMediaPlayer != null) {
            return mCurrentBufferPercentage;
        }
        return 0;
    }

    private boolean isInPlaybackState() {
        return ((mMediaPlayer != null) && (mCurrentState != STATE_ERROR) && (mCurrentState !=
STATE_IDLE) && (mCurrentState != STATE_PREPARING));
    }

    @Override
    public boolean canPause() {
        return false;
    }

    @Override
    public boolean canSeekBackward() {
        return false;
    }

    @Override
    public boolean canSeekForward() {
        return false;
    }

    @Override
    public int getAudioSessionId() {
        if (mAudioSession == 0) {
            MediaPlayer foo = new MediaPlayer();
            mAudioSession = foo.getAudioSessionId();
            foo.release();
        }
        return mAudioSession;
    }

    // Listeners
    private MediaPlayer.OnBufferingUpdateListener mBufferingUpdateListener = new
MediaPlayer.OnBufferingUpdateListener() {
        @Override
        public void onBufferingUpdate(final MediaPlayer mp, final int percent) {
            mCurrentBufferPercentage = percent;
        }
    };
```

```
    private MediaPlayer.OnCompletionListener mCompleteListener = new
MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(final MediaPlayer mp) {
            mCurrentState = STATE_PLAYBACK_COMPLETED;
            mTargetState = STATE_PLAYBACK_COMPLETED;
            mSurface.release();

            if (mMediaController != null) {
                mMediaController.hide();
            }

            if (mOnCompletionListener != null) {
                mOnCompletionListener.onCompletion(mp);
            }

            if (mMediaControllListener != null) {
                mMediaControllListener.onComplete();
            }
        }
    };

    private MediaPlayer.OnPreparedListener mPreparedListener = new
MediaPlayer.OnPreparedListener() {
        @Override
        public void onPrepared(final MediaPlayer mp) {
            mCurrentState = STATE_PREPARED;

            mMediaController = new MediaController(getContext());

            if (mOnPreparedListener != null) {
                mOnPreparedListener.onPrepared(mMediaPlayer);
            }
            if (mMediaController != null) {
                mMediaController.setEnabled(true);
                //mMediaController.setAnchorView(getRootView());
            }

            mVideoWidth = mp.getVideoWidth();
            mVideoHeight = mp.getVideoHeight();

            int seekToPosition = mSeekWhenPrepared; // mSeekWhenPrepared may be
            // changed after seekTo()
            // call
            if (seekToPosition != 0) {
                seekTo(seekToPosition);
            }

            requestLayout();
            invalidate();
            if ((mVideoWidth != 0) && (mVideoHeight != 0)) {
                if (mTargetState == STATE_PLAYING) {
                    mMediaPlayer.start();
                    if (null != mMediaControllListener) {
                        mMediaControllListener.onStart();
                    }
                }
            } else {
                if (mTargetState == STATE_PLAYING) {
                    mMediaPlayer.start();
                    if (null != mMediaControllListener) {
```

```
                        mMediaControllListener.onStart();
                    }
                }
            }
        }
    };

    private MediaPlayer.OnVideoSizeChangedListener mVideoSizeChangedListener = new
MediaPlayer.OnVideoSizeChangedListener() {
        @Override
        public void onVideoSizeChanged(final MediaPlayer mp, final int width, final int
height) {
            mVideoWidth = mp.getVideoWidth();
            mVideoHeight = mp.getVideoHeight();
            if (mVideoWidth != 0 && mVideoHeight != 0) {
                requestLayout();
            }
        }
    };

    private MediaPlayer.OnErrorListener mErrorListener = new MediaPlayer.OnErrorListener() {
        @Override
        public boolean onError(final MediaPlayer mp, final int what, final int extra) {
            Log.d(TAG, "Error: " + what + "," + extra);
            mCurrentState = STATE_ERROR;
            mTargetState = STATE_ERROR;

            if (mMediaController != null) {
                mMediaController.hide();
            }

            /* If an error handler has been supplied, use it and finish. */
            if (mOnErrorListener != null) {
                if (mOnErrorListener.onError(mMediaPlayer, what, extra)) {
                    return true;
                }
            }

            /*
             * Otherwise, pop up an error dialog so the user knows that
             * something bad has happened. Only try and pop up the dialog if
             * we're attached to a window. When we're going away and no longer
             * have a window, don't bother showing the user an error.
             */
            if (getWindowToken() != null) {

//                new AlertDialog.Builder(mContext).setMessage("Error: " + what + "," +
extra).setPositiveButton("OK", new DialogInterface.OnClickListener() {
//                    public void onClick(DialogInterface dialog, int whichButton) {
//                        /*
//                         * If we get here, there is no onError listener, so at
//                         * least inform them that the video is over.
//                         */
//                        if (mOnCompletionListener != null) {
//                            mOnCompletionListener.onCompletion(mMediaPlayer);
//                        }
//                    }
//                }).setCancelable(false).show();
            }
            return true;
        }
```

```
    };

    SurfaceTextureListener mSurfaceTextureListener = new SurfaceTextureListener() {
        @Override
        public void onSurfaceTextureAvailable(final SurfaceTexture surface, final int width,
final int height) {
            Log.d(TAG, "onSurfaceTextureAvailable.");
            mSurfaceTexture = surface;
            openVideo();
        }

        @Override
        public void onSurfaceTextureSizeChanged(final SurfaceTexture surface, final int width,
final int height) {
            Log.d(TAG, "onSurfaceTextureSizeChanged: " + width + '/' + height);
            mSurfaceWidth = width;
            mSurfaceHeight = height;
            boolean isValidState = (mTargetState == STATE_PLAYING);
            boolean hasValidSize = (mVideoWidth == width && mVideoHeight == height);
            if (mMediaPlayer != null && isValidState && hasValidSize) {
                if (mSeekWhenPrepared != 0) {
                    seekTo(mSeekWhenPrepared);
                }
                start();
            }
        }

        @Override
        public boolean onSurfaceTextureDestroyed(final SurfaceTexture surface) {

            mSurface = null;
            if (mMediaController != null)
                mMediaController.hide();
            release(true);
            return true;
        }

        @Override
        public void onSurfaceTextureUpdated(final SurfaceTexture surface) {

        }
    };

    /**
     * Register a callback to be invoked when the media file is loaded and ready
     * to go.
     *
     * @param l The callback that will be run
     */
    public void setOnPreparedListener(MediaPlayer.OnPreparedListener l) {
        mOnPreparedListener = l;
    }

    /**
     * Register a callback to be invoked when the end of a media file has been
     * reached during playback.
     *
     * @param l The callback that will be run
     */
    public void setOnCompletionListener(OnCompletionListener l) {
        mOnCompletionListener = l;
```

```
    }

    /**
     * Register a callback to be invoked when an error occurs during playback or
     * setup. If no listener is specified, or if the listener returned false,
     * VideoView will inform the user of any errors.
     *
     * @param l The callback that will be run
     */
    public void setOnErrorListener(OnErrorListener l) {
        mOnErrorListener = l;
    }

    /**
     * Register a callback to be invoked when an informational event occurs
     * during playback or setup.
     *
     * @param l The callback that will be run
     */
    public void setOnInfoListener(OnInfoListener l) {
        mOnInfoListener = l;
    }

    public static interface MediaControllListener {
        public void onStart();

        public void onPause();

        public void onStop();

        public void onComplete();
    }

    MediaControllListener mMediaControllListener;

    public void setMediaControllListener(MediaControllListener mediaControllListener) {
        mMediaControllListener = mediaControllListener;
    }


    @Override
    public void setVisibility(int visibility) {
        System.out.println("setVisibility: " + visibility);
        super.setVisibility(visibility);
    }
}
```

Hilfe aus diesem Gitub-Repository . Obwohl es einige Probleme hatte, wie es vor 3 Jahren geschrieben wurde, habe ich es geschafft, sie selbst zu beheben, wie oben beschrieben.

Optimiertes VideoView online lesen: https://riptutorial.com/de/android/topic/10638/optimiertes-videoview

Kapitel 168: Orientierungsänderungen

Bemerkungen

Referenz: https://guides.codepath.com/android/Handling-Configuration-Changes#references

Examples

Aktivitätsstatus speichern und wiederherstellen

Wenn Ihre Aktivität zu stoppen beginnt, ruft das System `onSaveInstanceState()` damit Ihre Aktivität Zustandsinformationen mit einer Sammlung von Schlüssel-Wert-Paaren speichern kann. Die Standardimplementierung dieser Methode speichert automatisch Informationen zum Status der Ansichtshierarchie der Aktivität, z. B. den Text in einem `EditText` Widget oder die `ListView` einer `ListView` .

Um zusätzliche `onSaveInstanceState()` für Ihre Aktivität zu speichern, müssen Sie `onSaveInstanceState()` implementieren und dem Bundle-Objekt Schlüsselwertpaare hinzufügen. Zum Beispiel:

```
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);
    }
}
```

Das System ruft diese Methode auf, bevor eine Aktivität zerstört wird. Später ruft das System `onRestoreInstanceState` wo wir den Status aus dem Bundle wiederherstellen können:

```
@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);
}
```

Der Instanzstatus kann auch in der standardmäßigen Activity # onCreate-Methode wiederhergestellt werden. In onRestoreInstanceState ist dies jedoch `onRestoreInstanceState` wodurch sichergestellt wird, dass die gesamte Initialisierung abgeschlossen ist und die

Unterklassen entscheiden, ob sie die Standardimplementierung verwenden. Lesen Sie diesen Stackoverflow-Beitrag für Details.

Beachten Sie, dass es nicht garantiert ist, dass `onSaveInstanceState` und `onRestoreInstanceState` gemeinsam aufgerufen werden. Android ruft `onSaveInstanceState()` wenn die Möglichkeit besteht, dass die Aktivität zerstört wird. Es gibt jedoch Fälle, in denen `onSaveInstanceState` aufgerufen wird, die Aktivität jedoch nicht zerstört wird und `onRestoreInstanceState` daher nicht aufgerufen wird.

Fragmentstatus speichern und wiederherstellen

Fragmente haben auch eine `onSaveInstanceState()` Methode, die aufgerufen wird, wenn ihr Status gespeichert werden muss:

```
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);
    }
}
```

Dann können wir Daten aus diesem gespeicherten Status in `onCreateView` :

```
public class MySimpleFragment extends Fragment {
   // ...

   // Inflate the view for the fragment based on layout XML
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.my_simple_fragment, container, false);
        if (savedInstanceState != null) {
            someStateValue = savedInstanceState.getInt(SOME_VALUE_KEY);
            // Do something with value if needed
        }
        return view;
   }
}
```

Damit der Fragmentstatus ordnungsgemäß gespeichert werden kann, müssen wir sicherstellen, dass das Fragment bei Konfigurationsänderungen nicht unnötig neu erstellt wird. Dies bedeutet, dass Sie vorhandene Fragmente nicht erneut initialisieren, wenn sie bereits vorhanden sind. Alle Fragmente, die in einer Aktivität initialisiert werden, müssen nach einer Konfigurationsänderung nach einem Tag gesucht werden:

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

Dies erfordert, dass wir ein Tag für die Suche einschließen, wenn Sie ein Fragment in die Aktivität einer Transaktion einfügen:

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

Mit diesem einfachen Muster können wir Fragmente ordnungsgemäß wiederverwenden und ihren Status bei Konfigurationsänderungen wiederherstellen.

Fragmente erhalten

In vielen Fällen können Probleme vermieden werden, wenn eine Aktivität neu erstellt wird, indem einfach Fragmente verwendet werden. Wenn sich Ihre Ansichten und Ihr Status in einem Fragment befinden, kann das Fragment leicht beibehalten werden, wenn die Aktivität neu erstellt wird:

```
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;
    }
}
```

Dieser Ansatz verhindert, dass das Fragment während des Aktivitätslebenszyklus zerstört wird. Sie werden stattdessen im Fragment-Manager beibehalten. Weitere Informationen finden Sie in den offiziellen Dokumenten von Android.

Jetzt können Sie anhand des Tags prüfen, ob das Fragment bereits vorhanden ist, bevor Sie ein Fragment erstellen, und das Fragment behält seinen Status bei Konfigurationsänderungen. Weitere Informationen finden Sie im Handbuch Umgang mit Laufzeitänderungen.

Bildschirmausrichtung sperren

Wenn Sie die Änderung der Bildschirmausrichtung eines Bildschirms (Aktivität) Ihrer Android-Anwendung sperren möchten, müssen Sie lediglich die `android:screenOrientation` einer `<activity>` im **AndroidManifest.xml festlegen** :

```
<activity
    android:name="com.techblogon.screenorientationexample.MainActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
    <!-- ... -->
</activity>
```

Nun muss diese Aktivität immer im **Hochformat** angezeigt werden.

Konfigurationsänderungen manuell verwalten

Wenn Ihre Anwendung während einer bestimmten Konfigurationsänderung keine Ressourcen aktualisieren muss und Sie eine Leistungsbeschränkung haben, die den Neustart der Aktivität verhindert, können Sie erklären, dass Ihre Aktivität die Konfigurationsänderung selbst übernimmt, wodurch das System Ihren Neustart nicht durchführen kann Aktivität.

Diese Technik sollte jedoch als letzter Ausweg angesehen werden, wenn Sie Neustarts aufgrund einer Konfigurationsänderung vermeiden müssen. Dies wird für die meisten Anwendungen nicht empfohlen. Um diesen Ansatz zu `android:configChanges` , müssen wir den Knoten `android:configChanges` zur Aktivität in der **AndroidManifest.xml hinzufügen** :

```
<activity android:name=".MyActivity"
          android:configChanges="orientation|screenSize|keyboardHidden"
          android:label="@string/app_name">
```

Wenn sich nun eine dieser Konfigurationen ändert, wird die Aktivität nicht neu `onConfigurationChanged()` sondern empfängt stattdessen einen Aufruf an `onConfigurationChanged()` :

```
// Within the activity which receives these changes
// Checks the current device orientation, and toasts accordingly
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);

    // Checks the orientation of the screen
    if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
        Toast.makeText(this, "landscape", Toast.LENGTH_SHORT).show();
    } else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT){
        Toast.makeText(this, "portrait", Toast.LENGTH_SHORT).show();
    }
}
```

Siehe den Abschnitt Umgang mit den Änderungsdokumenten . Weitere Informationen darüber, welche Konfigurationsänderungen Sie in Ihrer Aktivität ausführen können, finden Sie in der Dokumentation zu android: configChanges und in der Konfigurationsklasse .

Umgang mit AsyncTask

Problem:

- Wenn nach dem `AsyncTask` eine Bildschirmrotation erfolgt, wird die Besitzeraktivität zerstört und neu erstellt.
- Wenn die `AsyncTask` , möchte sie die möglicherweise nicht mehr gültige Benutzeroberfläche aktualisieren.

Lösung:

Mit Loaders kann man die Zerstörung / Wiederherstellung von Aktivitäten leicht überwinden.

Beispiel:

Hauptaktivität:

```
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;
    }
}
```

Hinweis:

Es ist wichtig, entweder die v4-Kompatibilitätsbibliothek zu verwenden oder nicht, aber nicht einen Teil des einen und einen Teil des anderen, da dies zu Kompilierungsfehlern führen kann. Um zu überprüfen, können Sie die Importe für `android.support.v4.content` und `android.content` (Sie sollten nicht beides haben).

Bildschirmsperre programmgesteuert sperren

Es ist sehr üblich, dass es während der Entwicklung **sehr nützlich sein** kann, **den Bildschirm**

des Geräts während bestimmter Teile des Codes zu sperren / zu entsperren .

*Wenn Sie beispielsweise ein Dialogfeld mit Informationen anzeigen, möchte der Entwickler möglicherweise die Bildschirmrotation **sperren** , um zu verhindern, dass das Dialogfeld geschlossen wird und die aktuelle Aktivität neu erstellt wird, **um** sie wieder zu **entsperren** , wenn das Dialogfeld geschlossen wird.*

Obwohl wir aus dem Manifest eine Rotationssperre erreichen können, indem wir Folgendes tun:

```
<activity
    android:name=".TheActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
</activity>
```

Man kann das auch programmgesteuert machen, indem man Folgendes tut:

```
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);
        }
    }
}
```

Und dann den folgenden Aufruf, um die Gerätedrehung zu sperren bzw. zu entsperren

```
lockDeviceRotation(true)
```

und

```
lockDeviceRotation(false)
```

Orientierungsänderungen online lesen:
https://riptutorial.com/de/android/topic/4621/orientierungsanderungen

Kapitel 169: ORMLite in Android

Examples

Beispiel für Android OrmLite über SQLite

ORMLite ist ein Object Relational Mapping-Paket, das einfache und **kompakte** Funktionen für das Beibehalten von Java-Objekten in SQL-Datenbanken bietet und dabei die Komplexität und den Mehraufwand von Standard-ORM-Paketen vermeidet.

OrmLite wird für Android über die sofort unterstützte Datenbank SQLite implementiert. Sie ruft die API direkt auf, um auf SQLite zuzugreifen.

Gradle-Setup

Um zu beginnen, sollten Sie das Paket dem Build-Gradle hinzufügen.

```
 // https://mvnrepository.com/artifact/com.j256.ormlite/ormlite-android
compile group: 'com.j256.ormlite', name: 'ormlite-android', version: '5.0'
POJO configuration
```

Dann sollten Sie einen POJO so konfigurieren, dass er in der Datenbank gespeichert bleibt. Hier muss auf die Anmerkungen geachtet werden:

- Fügen Sie die @DatabaseTable-Anmerkung oben jeder Klasse hinzu. Sie können auch @Entity verwenden.
- Fügen Sie die @DatabaseField-Anmerkung direkt vor jedem zu persistierenden Feld hinzu. Sie können auch @Column und andere verwenden.
- Fügen Sie jeder Klasse einen Konstruktor ohne Argumente hinzu, der mindestens die Paketsichtbarkeit hat.

```
@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;
    }
}
```

Im obigen Beispiel gibt es eine Tabelle (form_model) mit 4 Feldern.

ID-Feld ist automatisch erstellter Index.

Benutzername ist ein Index für die Datenbank.

Weitere Informationen zur Annotation finden Sie in der offiziellen Dokumentation .

Datenbank-Helfer

Um fortzufahren, müssen Sie eine Datenbank-Helper-Klasse erstellen, die die OrmLiteSqliteOpenHelper-Klasse erweitern soll.

Diese Klasse erstellt und aktualisiert die Datenbank, wenn Ihre Anwendung installiert wird, und kann auch die DAO-Klassen bereitstellen, die von Ihren anderen Klassen verwendet werden.

DAO steht für Data Access Object und bietet alle Scrum-Funktionen und ist auf die Behandlung einer einzelnen persistenten Klasse spezialisiert.

Die Helfer-Klasse muss die folgenden zwei Methoden implementieren:

- onCreate (SQLiteDatabase sqliteDatabase, ConnectionSource connectionSource);

  onCreate erstellt die Datenbank, wenn Ihre App zum ersten Mal installiert wird

- onUpgrade (SQLiteDatabase-Datenbank, ConnectionSource connectionSource, int oldVersion, int newVersion);

  OnUpgrade übernimmt die Aktualisierung der Datenbanktabellen, wenn Sie Ihre App auf eine neue Version aktualisieren

Beispiel für die Datenbankhilfeklasse:

```
  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;
        }
```

```
    }
```

Objekt bleibt in SQLite bestehen

Schließlich die Klasse, die das Objekt in der Datenbank speichert.

```
     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;
    }
}
```

Der Accessor des DOA am Konstruktor der obigen Klasse ist definiert als:

```
 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;
}
```

ORMLite in Android online lesen: https://riptutorial.com/de/android/topic/7571/ormlite-in-android

Kapitel 170: Ort

Einführung

Android-Standort-APIs werden in einer Vielzahl von Apps für verschiedene Zwecke verwendet, z. B. zum Ermitteln des Benutzerorts, zum Benachrichtigen, wenn ein Benutzer einen allgemeinen Bereich (Geofencing) verlassen hat, und zur Interpretation der Benutzeraktivitäten (Gehen, Laufen, Fahren usw.).

Android-Standort-APIs sind jedoch nicht die einzige Möglichkeit, den Standort des Benutzers zu ermitteln. Im Folgenden finden Sie Beispiele zur Verwendung von `LocationManager` und anderen gängigen Standortbibliotheken von Android.

Bemerkungen

Für das Erstellen von standortbezogenen Apps in Android gibt es zwei Pfade:

- Der native Open Source- `LocationManager` Android
- `FusedLocationProviderApi` von Google, ein Teil der Google Play Services

Location-Manager

Pros

- Feinere Kontrolle
- In allen Geräten verfügbar
- Teil des Android-Frameworks

Cons

- Die Batterieentladung ist ein Problem, wenn sie nicht ordnungsgemäß verwaltet wird
- Erfordert eine Logik zum Wechseln der Standortanbieter, wenn das Gerät keinen Standort finden kann (z. B. schlechtes GPS in einem Gebäude)

Eigenschaften

- NMEA-Listener
- GPS Status Listener
- Hören Sie sich den Status des Anbieters ab (z. B. GPS ist vom Benutzer deaktiviert)
- Liste der Anbieter, aus denen die Standortquelle ausgewählt werden kann

Anbieter

Geographisches Positionierungs System

- Erforderliche Berechtigungen:

    - `ACCESS_FINE_LOCATION`
- Genauigkeit: 10m - 100m
- Leistungsbedarf: HIGH
- Verfügbarkeit: Weltweit (mit freier Sicht zum Himmel)
- **HINWEISE** :
    - Standortaktualisierungen werden in der Regel einmal pro Sekunde gesendet. In Situationen, in denen GPS jedoch längere Zeit nicht verwendet wurde und A-GPS nicht verfügbar ist, dauert es einige Minuten, bis ein Standort empfangen wird.
    - In Fällen, in denen die freie Sicht zum Himmel versperrt ist, werden GPS-Punkte nicht sehr gut gebündelt (Positionspunkte "springen"), und die Genauigkeit kann aufgrund des " Urban Canyon " -Effekts in bestimmten Bereichen irreführend sein.

Netzwerk

- Erforderliche Berechtigungen:
    - `ACCESS_COARSE_LOCATION` oder `ACCESS_FINE_LOCATION`
- Genauigkeit: 100m - 1000m +
- Leistungsbedarf: LOW - MEDIUM
- Verfügbarkeit: In Reichweite des Zellenturm- oder WLAN-Signals
- **ANMERKUNGEN:**
    - Standortaktualisierungen werden seltener als GPS ausgeführt
    - Standortaktualisierungen bilden sich normalerweise nicht gut zusammen (Standortpunkte "springen") und die Genauigkeit kann von der Anzahl verschiedener Faktoren abhängen (Anzahl der WLAN-Signale, Signalstärke, Typ des Zellturms usw.).

Passiv

- Erforderliche Berechtigungen:
    - `ACCESS_FINE_LOCATION`
- Genauigkeit: 10m - 1000m +
- Leistungsbedarf: KEINE
- Verfügbarkeit: Nur wenn eine andere App einen Standort von GPS oder Netzwerk empfängt
- **ANMERKUNGEN:**
    - Verlassen Sie sich nicht darauf, um Sie ständig zu aktualisieren. Dieser hört passiv auf andere Apps, die Standortanfragen stellen, und gibt diese Orte zurück.
    - Gibt keine von FusedLocationProviderApi generierten Punkte zurück, sondern nur die zugrunde liegenden Standortpunkte, die zu ihrer Erzeugung verwendet werden.

FusedLocationProviderApi

Pros

- Bietet weniger Batterieentlastung "out of the box"
- Behandelt schlechtes GPS gut
- Ruft regelmäßig Updates auf

Cons

- Weniger genaue Kontrolle über GPS
- Ist möglicherweise nicht auf allen Geräten oder in bestimmten Ländern verfügbar
- Erfordert die Bibliotheksabhängigkeit von Drittanbietern

Eigenschaften

- Gut gemanagte Nutzung von Standortanbietern für optimale Einsparungen beim Batteriebetrieb
- Erzeugt normalerweise genauere Punkte als Network Location Provider
- Häufigere Aktualisierungen der Bibliothek ermöglichen weitere Verbesserungen
- Sie müssen nicht angeben, welcher Providertyp verwendet werden soll

LocationRequest Prioritätsstufen

PRIORITY_HIGH_ACCURACY

- Erforderliche Berechtigungen:
  - `ACCESS_FINE_LOCATION` für einen genaueren Standort oder `ACCESS_COARSE_LOCATION` für einen weniger genauen Standort
- Genauigkeit: 10m - 100m
- Leistungsbedarf: HIGH
- Verfügbarkeit: Wo immer Google Play-Dienste verfügbar sind.
- **ANMERKUNGEN:**
  - Wenn `ACCESS_FINE_LOCATION` nicht verwendet wird, verwendet dies kein GPS zum Generieren von Standortaktualisierungen, findet jedoch unter den richtigen Bedingungen immer noch einen ziemlich genauen Punkt.
  - Wenn `ACCESS_FINE_LOCATION` verwendet wird, kann GPS möglicherweise verwendet werden, um Standortpunkte zu generieren, je nachdem, wie genau das Gerät unter den Umgebungsbedingungen aktuell verfolgt werden kann.
  - Obwohl dies möglicherweise genauere Standortaktualisierungen enthält als die anderen Einstellungen, ist es dennoch anfällig für den " Urban Canyon " -Effekt.

PRIORITY_BALANCED_POWER_ACCURACY

- Erforderliche Berechtigungen:
  - `ACCESS_FINE_LOCATION` für einen genaueren Standort oder `ACCESS_COARSE_LOCATION` für einen weniger genauen Standort
- Genauigkeit: 100m - 1000m +
- Leistungsbedarf: MEDIUM
- Verfügbarkeit: Wo immer Google Play-Dienste verfügbar sind.
- **ANMERKUNGEN:**
  - Gleiche Notizen wie `PRIORITY_HIGH_ACCURACY`
  - Obwohl es unwahrscheinlich ist, kann diese Einstellung immer noch GPS verwenden, um einen Standort zu generieren.

PRIORITY_LOW_POWER

- Erforderliche Berechtigungen:
  - `ACCESS_FINE_LOCATION` oder `ACCESS_COARSE_LOCATION`
- Genauigkeit: 100m - 1000m +
- Leistungsbedarf: LOW
- Verfügbarkeit: Wo immer Google Play-Dienste verfügbar sind.
- **ANMERKUNGEN:**
  - Verwendet wahrscheinlich kein GPS, ist aber bisher noch nicht getestet.
  - Updates sind normalerweise nicht sehr genau
  - Wird im Allgemeinen zum Erkennen signifikanter Standortänderungen verwendet

PRIORITY_NO_POWER

- Erforderliche Berechtigungen:
  - `ACCESS_FINE_LOCATION` oder `ACCESS_COARSE_LOCATION`
- Genauigkeit: 10m - 1000m +
- Leistungsbedarf: KEINE
- Verfügbarkeit: Wo immer Google Play-Dienste verfügbar sind.
- **ANMERKUNGEN:**
  - Funktioniert fast identisch mit dem `LocationManager` `PASSIVE_PROVIDER`
  - `PASSIVE_PROVIDER` bei `PASSIVE_PROVIDER` Aktualisierungen der Google Play-Dienste zurück, wobei `PASSIVE_PROVIDER` die zugrunde liegenden Standortaktualisierungen `PASSIVE_PROVIDER`

Fehlerbehebung

OnLocationChanged () wurde nie aufgerufen

Da dies anscheinend ein häufiges Problem beim Beziehen von Android-Standorten ist, werde ich eine kurze Checkliste der gebräuchlichen Korrekturen auflisten:

---

1. Überprüfen Sie Ihr Manifest!

   Eines der häufigsten Probleme ist, dass die richtigen Berechtigungen niemals erteilt wurden. Wenn Sie GPS (mit oder ohne Netzwerk) verwenden, verwenden Sie `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>` , ansonsten verwenden Sie `<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>` . Die FusedLocationApi von Google erfordert `ACCESS_FINE_LOCATION` .

---

2. (Für Android 6+) Überprüfen Sie die Laufzeitberechtigungen !

   Nach Berechtigungen suchen und Berechtigungen anfordern! Wenn Sie niemals Berechtigungen erhalten, werden Sie zum Absturz führen oder, schlimmer (wenn Sie alle Ausnahmen wahrnehmen), erhalten Sie keine Hinweise auf irgendetwas! Es ist egal, ob der Benutzer Ihnen beim Start der App die Berechtigung erteilt. Überprüfen Sie immer, ob Sie für alle Anrufe Berechtigungen haben. Der Benutzer kann einfach zu seinen Einstellungen wechseln und sie widerrufen.

---

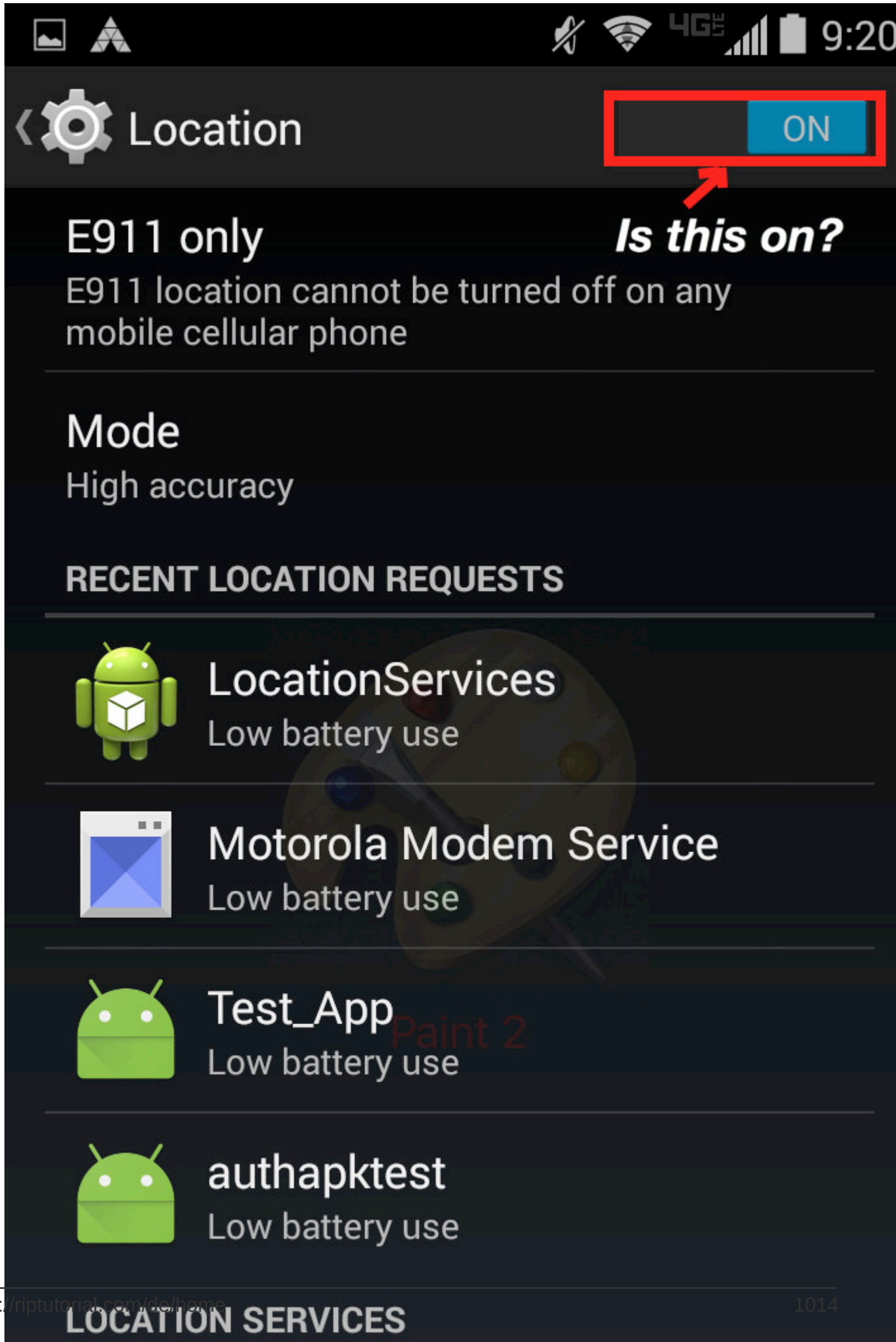
3. Überprüfen Sie Ihren Code!

   Sind Sie sicher, dass Sie den richtigen Hörer übergeben? Haben Sie diesen `BroadcastReceiver` oder `IntentService` zu Ihrem Manifest `IntentService` ? Verwenden Sie `PendingIntent.getService()` für eine `BroadcastReceiver` Klasse oder `getBroadcast()` für eine `IntentService` Klasse? Sind Sie sicher, dass Sie Ihren Listener nicht sofort nach der Abfrage an einer anderen Stelle in Ihrem Code abmelden?

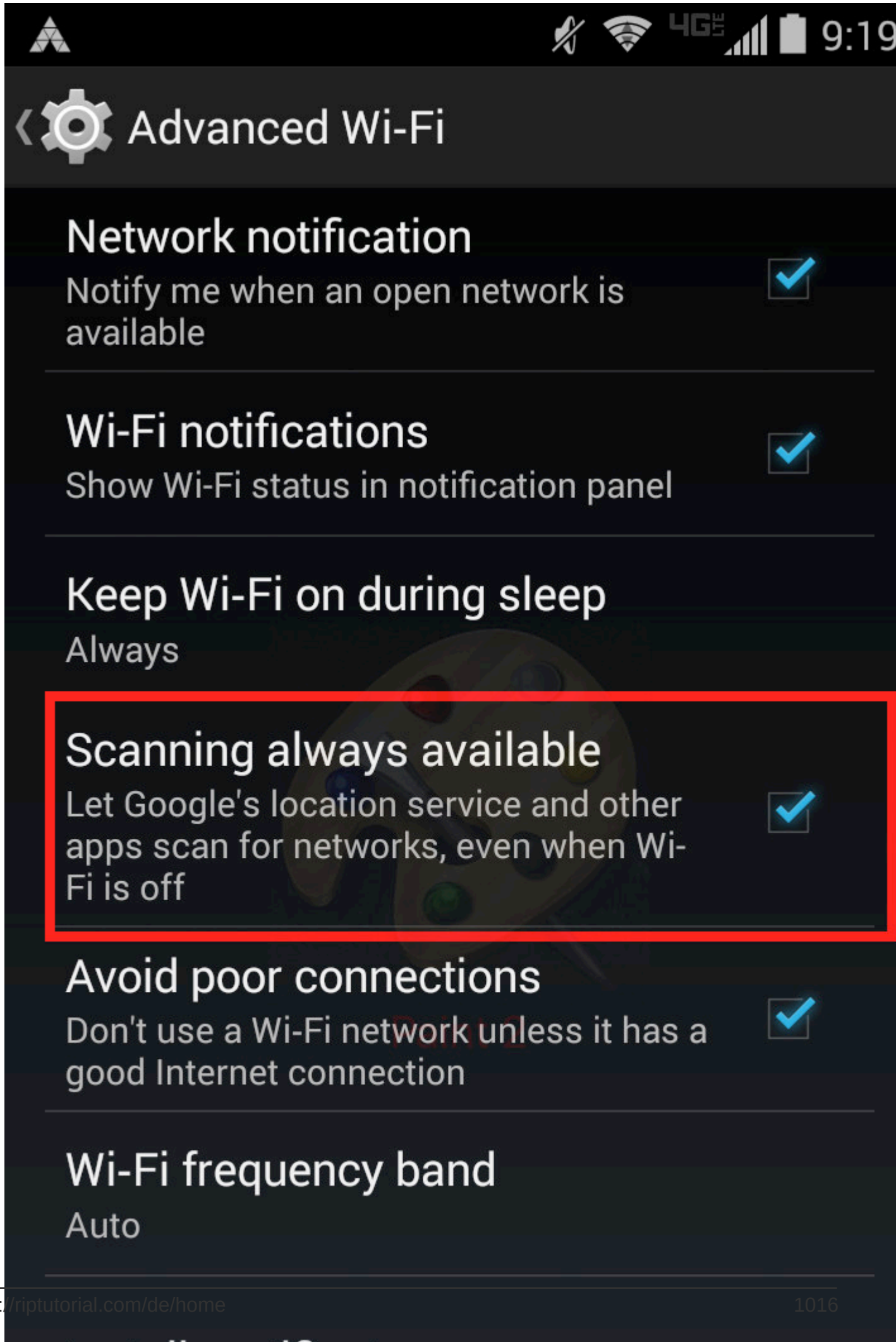
---

4. Geräteeinstellungen überprüfen!

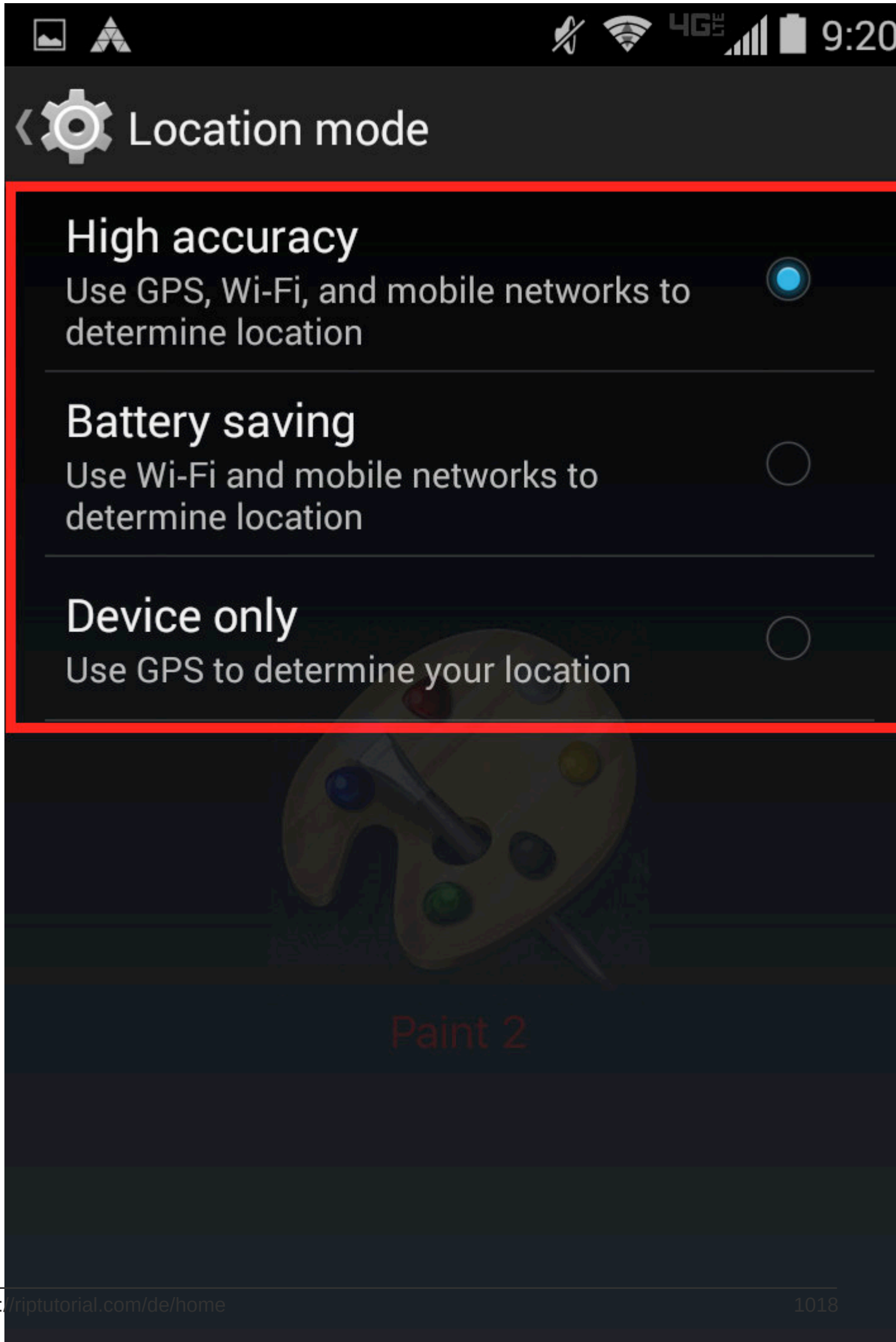
   Stellen Sie natürlich sicher, dass Sie Standortdienste aktiviert haben.


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

Wenn Sie Netzwerkdienste verwenden, haben Sie "Scannen immer verfügbar" aktiviert? Haben Sie Ihren Standortmodus auf "Beste" ("Hohe Genauigkeit") oder "Batteriesparen" ("Nur Netzwerk") eingestellt?


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

Wenn Sie GPS verwenden, haben Sie im Standortmodus "Beste" ("Hohe Genauigkeit") oder "Nur Gerät" aktiviert?


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

Ja, das ist zweimal hier. Haben Sie versucht, einen `LocationListener` anstelle eines `PendingIntent` oder umgekehrt, um sicherzustellen, dass Sie `LocationManager` tatsächlich ordnungsgemäß implementiert haben? Sind Sie sicher, dass die Standortanfrage in einem Teil des Aktivitäten- oder Service-Lebenszyklus, von dem Sie nicht erwartet hätten, nicht entfernt wird?

`PendingIntent` oder umgekehrt, um sicherzustellen, dass Sie `LocationManager` tatsächlich ordnungsgemäß implementiert haben? Sind Sie sicher, dass die Standortanfrage in einem Teil des Aktivitäten- oder Service-Lebenszyklus, von dem Sie nicht erwartet hätten, nicht entfernt wird?

---

6. Überprüfen Sie Ihre Umgebung!

   Testen Sie GPS im ersten Stock eines Gebäudes mitten in San Francisco? Testen Sie Netzwerkstandorte mitten im Nirgendwo? Arbeiten Sie in einem geheimen unterirdischen Bunker ohne Funksignale und fragen Sie sich, warum Ihr Gerät keinen Standort findet? Überprüfen Sie immer Ihre Umgebung, wenn Sie versuchen, Standortprobleme zu beheben!

Es kann viele andere, weniger offensichtliche Gründe dafür geben, dass der Standort nicht funktioniert. Bevor Sie jedoch die esoterischen Korrekturen durchsuchen, müssen Sie diese kurze Checkliste durchgehen.

Examples

Fixierte Standort-API

Beispiel mit Activity mit LocationRequest

```
/*
 * This example is useful if you only want to receive updates in this
 * activity only, and have no use for location anywhere else.
 */
public class LocationActivity extends AppCompatActivity implements
        GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
LocationListener {

    private GoogleApiClient mGoogleApiClient;
    private LocationRequest mLocationRequest;

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

        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
```

```
        mLocationRequest = new LocationRequest()
                .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY) //GPS quality location
points
                .setInterval(2000) //At least once every 2 seconds
                .setFastestInterval(1000); //At most once a second
    }

    @Override
    protected void onStart(){
        super.onStart();
        mGoogleApiClient.connect();
    }

    @Override
    protected void onResume(){
        super.onResume();
        //Permission check for Android 6.0+
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
            if(mGoogleApiClient.isConnected()) {
                LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
            }
        }
    }

    @Override
    protected void onPause(){
        super.onPause();
        //Permission check for Android 6.0+
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
            if(mGoogleApiClient.isConnected()) {
                LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient,
this);
            }
        }
    }

    @Override
    protected void onStop(){
        super.onStop();
        mGoogleApiClient.disconnect();
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
        }
    }

    @Override
    public void onConnectionSuspended(int i) {
        mGoogleApiClient.connect();
    }

    @Override
```

```
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    }

    @Override
    public void onLocationChanged(Location location) {
        //Handle your location update code here
    }
}
```

Beispiel mit Service mit PendingIntent und BroadcastReceiver

BeispielAktivität

Empfohlene Lektüre: 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
            }
        }
    }
}
```

LocationService

HINWEIS: Vergessen Sie nicht, diesen Dienst im Manifest zu registrieren!

```
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

HINWEIS: Vergessen Sie nicht, diesen Empfänger im Manifest zu registrieren!

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

Anfordern von Standortaktualisierungen mit LocationManager

Wie immer müssen Sie sicherstellen, dass Sie über die erforderlichen Berechtigungen verfügen.

```
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) {

    }
}
```

Anfordern von Standortaktualisierungen in einem separaten Thread mithilfe von LocationManager

Wie immer müssen Sie sicherstellen, dass Sie über die erforderlichen Berechtigungen verfügen.

```
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) {

    }
}
```

Geofence registrieren

Ich habe die `GeoFenceObserversationService` **Singleton-** Klasse erstellt.

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);
        }
    }
```

Wo habe ich Service angefangen? Aus der Anwendungsklasse

- `startService(new Intent(getApplicationContext(),GeoFenceObserversationService.class));`

Wie habe ich Geofences registriert?

- `GeoFenceObserversationService.getInstant().addGeofences();`

Adresse von Standort mit Geocoder abrufen

Nachdem Sie das bekam `Location` - Objekt aus `FusedAPI` , können Sie ganz einfach erwerben `Address` von diesem Objekt.

```
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;
}
```

Standortaktualisierungen in einem BroadcastReceiver abrufen

Erstellen Sie zuerst eine BroadcastReceiver-Klasse, um die eingehenden Standortaktualisierungen zu verarbeiten:

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

```
        }
    }
}
```

Wenn Sie dann beim onConnected-Rückruf eine Verbindung zum GoogleApiClient herstellen:

```
@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);
}
```

Vergessen Sie nicht, die Standortaktualisierungsabsicht im entsprechenden Lebenszyklus-Rückruf zu entfernen:

```
@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();
}
```

Ort online lesen: https://riptutorial.com/de/android/topic/1837/ort

Kapitel 171: Otto Event Bus

Bemerkungen

Otto ist veraltet zugunsten von `RxJava` und `RxAndroid` . Diese Projekte erlauben dasselbe ereignisgesteuerte Programmiermodell wie Otto, sind jedoch leistungsfähiger und bieten eine bessere Steuerung des Threadings.

Examples

Eine Veranstaltung übergeben

Dieses Beispiel beschreibt das Übergeben eines Ereignisses mit dem Otto Event Bus .

Um den Otto Event Bus in **Android Studio verwenden zu können** , müssen Sie die folgende Anweisung in die Modul-Datei einfügen:

```
dependencies {
    compile 'com.squareup:otto:1.3.8'
}
```

Das Ereignis, das wir übergeben möchten, ist ein einfaches Java-Objekt:

```
public class DatabaseContentChangedEvent {
    public String message;

    public DatabaseContentChangedEvent(String message) {
        this.message = message;
    }
}
```

Wir brauchen einen Bus, um Ereignisse zu senden. Dies ist normalerweise ein Singleton:

```
import com.squareup.otto.Bus;

public final class BusProvider {
    private static final Bus mBus = new Bus();

    public static Bus getInstance() {
        return mBus;
    }

    private BusProvider() {
    }
}
```

Um eine Veranstaltung zu versenden, benötigen wir nur unseren BusProvider und dessen `post` Methode. Hier senden wir ein Ereignis, wenn die Aktion einer AsyncTask abgeschlossen ist:

```
public abstract class ContentChangingTask extends AsyncTask<Object, Void, Void> {

    ...

    @Override
    protected void onPostExecute(Void param) {
        BusProvider.getInstance().post(
            new DatabaseContentChangedEvent("Content changed")
        );
    }
}
```

Ereignis empfangen

Um ein Ereignis zu erhalten, muss eine Methode mit dem Ereignistyp als Parameter implementiert und mit `@Subscribe` kommentiert werden. Außerdem müssen Sie die Instanz Ihres Objekts beim `BusProvider` registrieren / die `BusProvider` (siehe Beispiel *Ereignis senden*):

```
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);
    }
}
```

Wichtig: Um dieses Ereignis zu erhalten, muss eine Instanz der Klasse vorhanden sein. Dies ist normalerweise nicht der Fall, wenn Sie ein Ergebnis von einer Aktivität an eine andere Aktivität senden möchten. Überprüfen Sie also Ihren Use Case für den Event-Bus.

Otto Event Bus online lesen: https://riptutorial.com/de/android/topic/6068/otto-event-bus

Kapitel 172: Paginierung in RecyclerView

Einführung

Paginierung ist ein häufiges Problem bei vielen mobilen Apps, die mit Datenlisten umgehen müssen. Die meisten mobilen Apps nutzen nun das "Endlos-Seiten" -Modell, bei dem das Scrollen automatisch in neuen Inhalt geladen wird. CWAC Endless Adapter macht es sehr einfach, dieses Muster in Android-Anwendungen zu verwenden

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

öffentliche Schnittstelle 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);
        }
    }

}
```

Paginierung in RecyclerView online lesen:
https://riptutorial.com/de/android/topic/9243/paginierung-in-recyclerview

Kapitel 173: Paketierbar

Einführung

Parcelable ist eine Android-spezifische Schnittstelle, in der Sie die Serialisierung selbst implementieren. Es wurde erstellt, um effizienter zu sein als Serializable, und um einige Probleme mit dem Standard-Java-Serialisierungsschema zu umgehen.

Bemerkungen

Denken Sie daran, dass die Reihenfolge, in der Sie Felder in ein Paket schreiben, MÜSSEN, DASS SIE DIESE BESTELLUNG MÜSSEN, dass Sie sie beim Erstellen Ihres benutzerdefinierten Objekts aus dem Paket lesen.

Die Paketschnittstelle hat eine strikte Beschränkung von 1 MB. Das bedeutet, dass jedes Objekt oder eine Kombination von Objekten, die Sie in ein Flurstück legen, das mehr als 1 MB Speicherplatz belegt, auf der anderen Seite beschädigt wird. Dies kann schwer zu entdecken sein. Denken Sie also daran, welche Art von Objekten Sie planen, um Parcelable zu erstellen. Wenn sie große Abhängigkeitsbäume haben, sollten Sie eine andere Methode für die Weitergabe von Daten in Betracht ziehen.

Examples

Ein benutzerdefiniertes Objekt erstellen

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

Parcelable-Objekt, das ein anderes Parcelable-Objekt enthält

Ein Beispiel für eine Klasse, die eine Parcelable-Klasse enthält:

```
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;
    }
}
```

Besitzer ist nur eine normale Klasse für Parcelable.

Verwenden von Enums mit Parcelable

```
/**
 * Created by Nick Cardoso on 03/08/16.
```

```
 * This is not a complete parcelable implementation, it only highlights the easiest
 * way to read and write your Enum values to your parcel
 */
public class Foo implements Parcelable {

    private final MyEnum myEnumVariable;
    private final MyEnum mySaferEnumVariableExample;

    public Foo(Parcel in) {

        //the simplest way
        myEnumVariable = MyEnum.valueOf( in.readString() );

        //with some error checking
        try {
            mySaferEnumVariableExample= MyEnum.valueOf( in.readString() );
        } catch (IllegalArgumentException e) { //bad string or null value
            mySaferEnumVariableExample= MyEnum.DEFAULT;
        }

    }

    ...

    @Override
    public void writeToParcel(Parcel dest, int flags) {

        //the simple way
        dest.writeString(myEnumVariable.name());

        //avoiding NPEs with some error checking
        dest.writeString(mySaferEnumVariableExample == null? null :
mySaferEnumVariableExample.name());

    }

}

public enum MyEnum {
    VALUE_1,
    VALUE_2,
    DEFAULT
}
```

Dies ist vorzuziehen, um (beispielsweise) eine Ordinalzahl zu verwenden, da das Einfügen neuer Werte in Ihre Enummen zuvor gespeicherte Werte nicht beeinflusst

Paketierbar online lesen: https://riptutorial.com/de/android/topic/1849/paketierbar

Kapitel 174: Paket-Manager

Examples

Anwendungsversion abrufen

```
public String getAppVersion() throws PackageManager.NameNotFoundException {
    PackageManager manager = getApplicationContext().getPackageManager();
    PackageInfo info = manager.getPackageInfo(
                            getApplicationContext().getPackageName(),
                            0);

    return info.versionName;
}
```

Versionsname und Versionscode

Um `versionName` und `versionCode` der aktuellen Build Ihrer Anwendung sollten Sie Android Paketmanager abfragen.

```
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
}
```

Installationszeit und Aktualisierungszeit

Um den Zeitpunkt zu ermitteln, zu dem Ihre App installiert oder aktualisiert wurde, sollten Sie den Paketmanager von Android abfragen.

```
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
}
```

Dienstprogrammmethode mit PackageManager

Hier finden Sie eine nützliche Methode mit PackageManager.

Die Methode unten hilft, den App-Namen anhand des Paketnamens zu ermitteln

```
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;
}
```

Die Methode unten hilft, das App-Symbol anhand des Paketnamens zu erhalten.

```
private Drawable getAppIcon(String packageName, Context context) {
    Drawable appIcon = null;
    try {
        appIcon = context.getPackageManager().getApplicationIcon(packageName);
    } catch (PackageManager.NameNotFoundException e) {
    }

    return appIcon;
}
```

Die Methode unten hilft, die Liste der installierten Anwendungen abzurufen.

```
public static List<ApplicationInfo> getLaunchIntent(PackageManager packageManager) {

    List<ApplicationInfo> list =
packageManager.getInstalledApplications(PackageManager.GET_META_DATA);

    return list;
}
```

Hinweis: Die obige Methode gibt auch die Launcher-Anwendung an.

Die nachstehende Methode hilft dabei, das App-Symbol vor dem Startprogramm auszublenden.

```
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);
    }
}
```

Hinweis: Nach dem Ausschalten des Geräts und dem Einschalten wird dieses Symbol wieder im Launcher angezeigt.

Paket-Manager online lesen: https://riptutorial.com/de/android/topic/4670/paket-manager

Kapitel 175: Picasso

Einführung

Picasso ist eine Bildbibliothek für Android. Es wurde von Square erstellt und verwaltet. Es vereinfacht die Anzeige von Bildern von externen Standorten aus. Die Bibliothek verarbeitet jede Phase des Prozesses, von der ersten HTTP-Anforderung bis zum Zwischenspeichern des Images. In vielen Fällen sind nur wenige Codezeilen erforderlich, um diese übersichtliche Bibliothek zu implementieren.

Bemerkungen

Picasso ist eine leistungsstarke Bibliothek zum Herunterladen und Zwischenspeichern von Bildern für Android.
Folgen Sie diesem Beispiel , um die Bibliothek zu Ihrem Projekt hinzuzufügen.

Websites:

- Quelle
- Doc
- Änderungsprotokoll

Examples

Picasso-Bibliothek zu Ihrem Android-Projekt hinzufügen

Aus der offiziellen Dokumentation :

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>
```

Platzhalter und Fehlerbehandlung

Picasso unterstützt sowohl Platzhalter für das Herunterladen als auch für Fehler als optionale Funktionen. Es bietet auch Rückrufe zur Verarbeitung des Download-Ergebnisses.

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

Eine Anforderung wird dreimal wiederholt, bevor der Fehlerplatzhalter angezeigt wird.

Größe ändern und drehen

```
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);
```

Kreisavatare mit Picasso

Hier ist ein Beispiel für eine Picasso-Circle-Transformationsklasse, die auf dem Original basiert, einen dünnen Rahmen hinzugefügt hat und außerdem Funktionen für ein optionales Trennzeichen zum Stapeln enthält:

```
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){
```

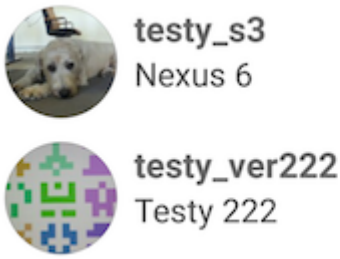
```
        mCircleSeparator = circleSeparator;
    }

    @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";
    }
}
```

Gehen Sie wie folgt vor, wenn Sie ein Bild laden (vorausgesetzt, `this` ist ein Aktivitätskontext und die `url` ist eine Zeichenfolge mit der URL des zu ladenden Bildes):

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
```

```
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform())
    .into(ivAvatar);
```

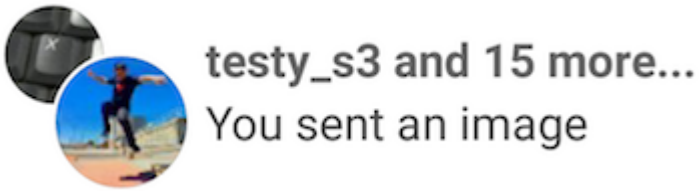
Ergebnis:



Geben Sie für die Verwendung des Trennzeichens dem Konstruktor für das obere Bild den Wert `true` an:

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform(true))
    .into(ivAvatar);
```

Ergebnis (zwei ImageViews in einem FrameLayout):



Deaktivieren Sie den Cache in Picasso

```
Picasso.with(context)
.load(uri)
.networkPolicy(NetworkPolicy.NO_CACHE)
.memoryPolicy(MemoryPolicy.NO_CACHE)
.placeholder(R.drawable.placeholder)
.into(imageView);
```

Bild von externem Speicher laden

```
String filename = "image.png";
String imagePath = getExternalFilesDir() + "/" + filename;

Picasso.with(context)
    .load(new File(imagePath))
    .into(imageView);
```

Bild als Bitmap mit Picasso herunterladen

Wenn Sie ein Bild als `Bitmap` mit `Picasso` herunterladen möchten, hilft der folgende Code:

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

Abbrechen von Bildanfragen mit Picasso

In bestimmten Fällen müssen wir eine Bilddownloadanfrage in Picasso abbrechen, bevor der Download abgeschlossen ist.

Dies kann aus verschiedenen Gründen geschehen, zum Beispiel, wenn die übergeordnete Ansicht zu einer anderen Ansicht übergegangen ist, bevor der Download der Bilder abgeschlossen werden kann.

In diesem Fall können Sie die `cancelRequest()` mit der Methode `cancelRequest()` abbrechen:

```
ImageView imageView;

//......

Picasso.with(imageView.getContext()).cancelRequest(imageView);
```

Verwenden von Picasso als ImageGetter für Html.fromHtml

Verwenden von Picasso als ImageGetter für Html.fromHtml

```
public class PicassoImageGetter implements Html.ImageGetter {

private TextView textView;

private Picasso picasso;

public PicassoImageGetter(@NonNull Picasso picasso, @NonNull TextView textView) {
    this.picasso = picasso;
    this.textView = textView;
}

@Override
```

```
public Drawable getDrawable(String source) {
    Log.d(PicassoImageGetter.class.getName(), "Start loading url " + source);

    BitmapDrawablePlaceHolder drawable = new BitmapDrawablePlaceHolder();

    picasso
            .load(source)
            .error(R.drawable.connection_error)
            .into(drawable);

    return drawable;
}

private class BitmapDrawablePlaceHolder extends BitmapDrawable implements Target {

    protected Drawable drawable;

    @Override
    public void draw(final Canvas canvas) {
        if (drawable != null) {
            checkBounds();
            drawable.draw(canvas);
        }
    }

    public void setDrawable(@Nullable Drawable drawable) {
        if (drawable != null) {
            this.drawable = drawable;
            checkBounds();
        }
    }

    private void checkBounds() {
        float defaultProportion = (float) drawable.getIntrinsicWidth() / (float)
drawable.getIntrinsicHeight();
        int width = Math.min(textView.getWidth(), drawable.getIntrinsicWidth());
        int height = (int) ((float) width / defaultProportion);

        if (getBounds().right != textView.getWidth() || getBounds().bottom != height) {

            setBounds(0, 0, textView.getWidth(), height); //set to full width

            int halfOfPlaceHolderWidth = (int) ((float) getBounds().right / 2f);
            int halfOfImageWidth = (int) ((float) width / 2f);

            drawable.setBounds(
                    halfOfPlaceHolderWidth - halfOfImageWidth, //centering an image
                    0,
                    halfOfPlaceHolderWidth + halfOfImageWidth,
                    height);

            textView.setText(textView.getText()); //refresh text
        }
    }

    //------------------------------------------------------------------//

    @Override
    public void onBitmapLoaded(Bitmap bitmap, Picasso.LoadedFrom from) {
        setDrawable(new BitmapDrawable(Application.getContext().getResources(), bitmap));
    }
```

```
    @Override
    public void onBitmapFailed(Drawable errorDrawable) {
        setDrawable(errorDrawable);
    }

    @Override
    public void onPrepareLoad(Drawable placeHolderDrawable) {
        setDrawable(placeHolderDrawable);
    }

    //------------------------------------------------------------------//

}
}
```

Die Verwendung ist einfach:

```
Html.fromHtml(textToParse, new PicassoImageGetter(picasso, textViewTarget), null);
```

Versuchen Sie es zuerst mit dem Offline-Festplattencache, gehen Sie dann online und rufen Sie das Image ab

Fügen Sie zunächst OkHttp zur Gradle-Build-Datei des App-Moduls hinzu

```
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'
```

Dann machen Sie eine Klasse, die Application erweitert

```
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);

    }
}
```

Fügen Sie es wie folgt der Manifest-Datei hinzu:

```
<application
```

```
        android:name=".Global"
        .. >

</application>
```

Normaler Gebrauch

```
Picasso.with(getActivity())
.load(imageUrl)
.networkPolicy(NetworkPolicy.OFFLINE)
.into(imageView, new Callback() {
    @Override
    public void onSuccess() {
        //Offline Cache hit
    }

    @Override
    public void onError() {
        //Try again online if cache failed
        Picasso.with(getActivity())
                .load(imageUrl)
                .error(R.drawable.header)
                .into(imageView, new Callback() {
            @Override
            public void onSuccess() {
                 //Online download
            }

            @Override
            public void onError() {
                Log.v("Picasso","Could not fetch image");
            }
        });
    }
});
```

Link zur ursprünglichen Antwort

Picasso online lesen: https://riptutorial.com/de/android/topic/2172/picasso

Kapitel 176: Ping ICMP

Einführung

Die ICMP-Ping-Anforderung kann in Android ausgeführt werden, indem ein neuer Prozess zum Ausführen der Ping-Anforderung erstellt wird. Das Ergebnis der Anforderung kann nach Abschluss der Ping-Anforderung in seinem Prozess ausgewertet werden.

Examples

Führt einen einzelnen Ping aus

In diesem Beispiel wird eine einzelne Ping-Anforderung versucht. Der ping-Befehl im `runtime.exec` Methodenaufruf kann in einen beliebigen gültigen ping-Befehl geändert werden, den Sie möglicherweise selbst in der Befehlszeile ausführen.

```
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();
}
```

Ping ICMP online lesen: https://riptutorial.com/de/android/topic/9434/ping-icmp

Kapitel 177: Port Mapping mithilfe der Cling-Bibliothek in Android

Examples

Hinzufügen von Cling-Unterstützung zu Ihrem Android-Projekt

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'

}
```

Zuordnung eines NAT-Ports

```
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;
}
```

Port Mapping mithilfe der Cling-Bibliothek in Android online lesen:
https://riptutorial.com/de/android/topic/6208/port-mapping-mithilfe-der-cling-bibliothek-in-android

Kapitel 178: PorterDuff-Modus

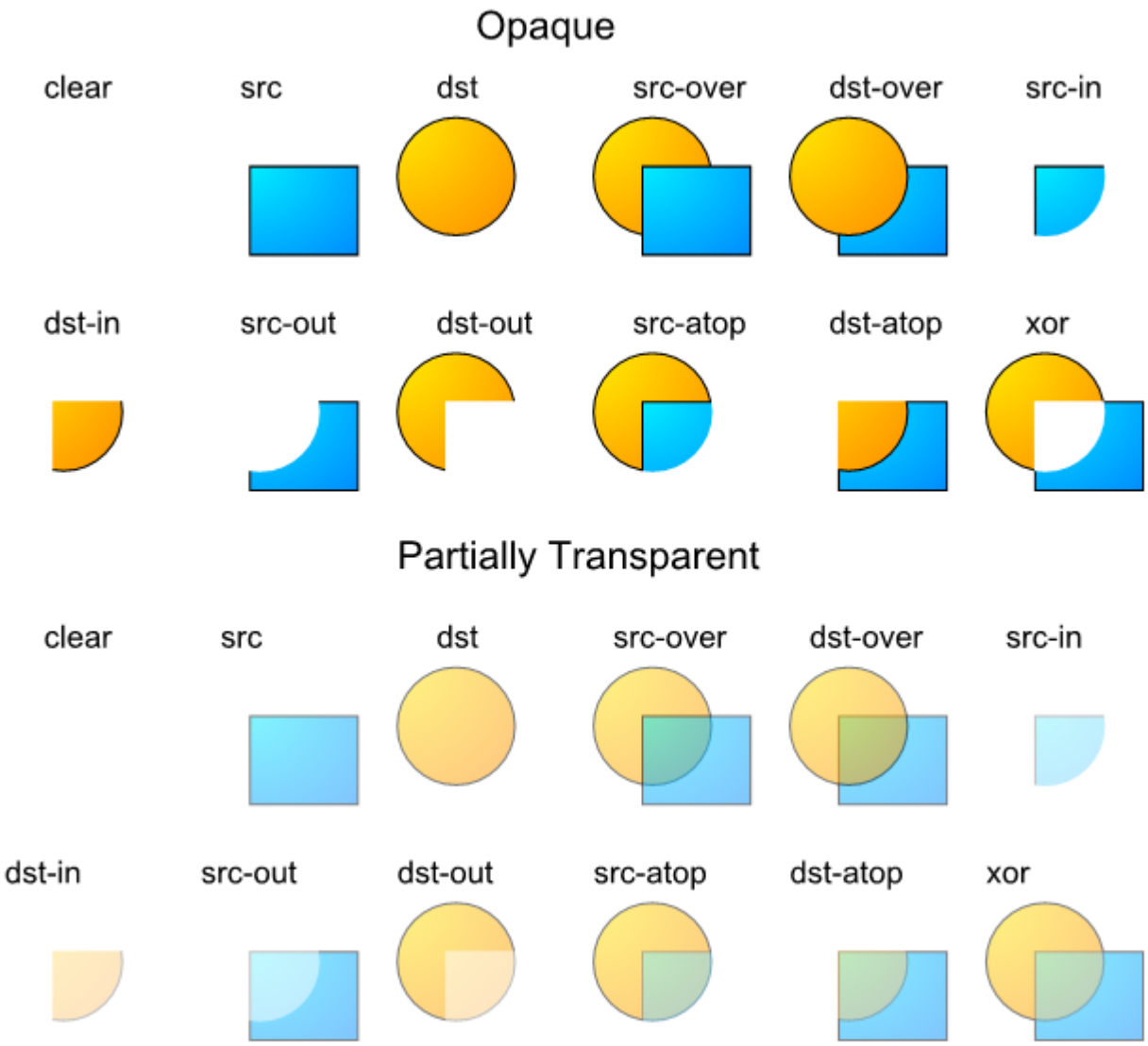
Einführung

PorterDuff wird als eine Möglichkeit beschrieben, Bilder so zu kombinieren, als wären sie "unregelmäßig geformte Kartonstücke", die übereinander angeordnet sind, sowie ein Schema zum Mischen der überlappenden Teile

Bemerkungen

"Porter Duff" an sich ist eine Alpha-Compositing-Technik, benannt nach einem Artikel von Thomas Porter und Tom Duff.

Zusammenfassend nimmt die Technik zwei Bilder mit Alphakanal und erzeugt das Ausgabebild durch Kombinieren der Pixelwerte von zwei Bildern. Die verschiedenen Kombinationsmodi führen zu unterschiedlichen Ausgabebildern. Im folgenden Bild wird beispielsweise die blaue Form (Quelle, vorhandene Pixel) mit der gelben Form (Ziel, neue Pixel) in verschiedenen Modi kombiniert:



Examples

Erstellen eines PorterDuff ColorFilter

`PorterDuff.Mode` wird zum Erstellen eines `PorterDuffColorFilter` . Ein Farbfilter ändert die Farbe jedes Pixels einer visuellen Ressource.

```
ColorFilter filter = new PorterDuffColorFilter(Color.BLUE, PorterDuff.Mode.SRC_IN);
```

Der obige Filter färbt die nicht transparenten Pixel blau ein.

Der Farbfilter kann auf ein `Drawable` :

```
drawable.setColorFilter(filter);
```

Es kann auf eine `ImageView` :

```
imageView.setColorFilter(filter);
```

Es kann auch auf eine `Paint` angewendet werden, sodass die Farbe, die mit dieser Farbe gezeichnet wird, vom Filter geändert wird:

```
paint.setColorFilter(filter);
```

Erstellen eines PorterDuff-XferMode

Ein `Xfermode` Modus (think "transfer" -Modus) dient als Übertragungsschritt beim Zeichnen der Pipeline. Wenn ein `Xfermode` auf einen `Paint` angewendet wird, werden die mit dem Paint gezeichneten Pixel mit darunter liegenden Pixeln (bereits gezeichnet) gemäß dem Modus kombiniert:

```
paint.setColor(Color.BLUE);
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
```

Jetzt haben wir eine blaue Tönungsfarbe. Jede gezeichnete Form färbt die bereits vorhandenen, nicht transparenten Pixel im Bereich der Form blau.

Wenden Sie eine radiale Maske (Vignette) mit PorterDuffXfermode auf eine Bitmap an

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

PorterDuff-Modus online lesen: https://riptutorial.com/de/android/topic/377/porterduff-modus

Kapitel 179: ProGuard - Verschleiern und Verkleinern Ihres Codes

Examples

Regeln für einige der weit verbreiteten Bibliotheken

Derzeit enthält es Regeln für folgende Bibliotheken: -

1. Buttermesser
2. RxJava
3. Android-Support-Bibliothek
4. Android Design-Unterstützungsbibliothek
5. Nachrüstung
6. Gson und Jackson
7. Otto
8. Crashlitycs
9. Picasso
10. Volley
11. OkHttp3
12. Paketierbar

```
#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;
}
```

Aktivieren Sie ProGuard für Ihren Build

Um `ProGuard` Konfigurationen für Ihre Anwendung zu aktivieren, müssen Sie sie in Ihrer Modullevel-Datei aktivieren. Sie müssen den Wert von `minifyEnabled true` .

Sie können auch `shrinkResources true` , um Ressourcen zu entfernen, die `ProGuard` als nicht verwendet markiert.

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

Der obige Code wird Ihre ProGuard-Konfigurationen, die in `proguard-rules.pro` ("proguard-project.txt" in Eclipse) enthalten sind, auf Ihre freigegebene apk- `proguard-rules.pro` anwenden.

Damit Sie später feststellen können, in welcher Zeile eine Ausnahme in einem Stack-Trace aufgetreten ist, sollte "proguard-rules.pro" folgende Zeilen enthalten:

```
-renamesourcefileattribute SourceFile
-keepattributes SourceFile,LineNumberTable
```

Um Proguard in Eclipse zu aktivieren, fügen Sie `proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt` zu "project.properties" hinzu.

Entfernen Sie die Traceprotokollierungsanweisungen (und andere Anweisungen) während der Erstellung

Wenn Sie Aufrufe bestimmter Methoden entfernen möchten, vorausgesetzt, dass sie ungültig sind und keine Nebeneffekte haben (z. B. werden durch das Aufrufen keine Systemwerte, Referenzargumente, Statik usw. geändert), können Sie sie von ProGuard aus dem System entfernen Ausgabe nachdem der Build abgeschlossen ist.

Ich halte dies beispielsweise für das Entfernen von Anweisungen zum Debuggen / Verbose-Protokollieren hilfreich, die beim Debuggen hilfreich sind, aber das Generieren der Zeichenfolgen für sie ist in der Produktion nicht erforderlich.

```
# 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(...);
}
```

Hinweis: Wenn `-dontoptimize` in einer ProGuard-Konfiguration verwendet wird, um nicht verwendeten Code nicht zu minimieren bzw. zu entfernen, werden die Anweisungen dadurch nicht entfernt. (Aber wer möchte nicht ungenutzten Code entfernen, oder?)

Hinweis 2: Durch diesen Aufruf wird der Anruf zum Protokollieren entfernt, der Code wird jedoch nicht geschützt. Die Strings bleiben tatsächlich im generierten apk. Lesen Sie mehr in diesem Beitrag .

Schützen Sie Ihren Code vor Hackern

Verschleierung wird oft als eine magische Lösung für den Codeschutz angesehen, indem Ihr Code schwieriger zu verstehen ist, wenn er jemals von Hackern dekompiliert wird.

Wenn Sie jedoch der `Log.x(..)` sind, dass das Entfernen von `Log.x(..)` tatsächlich die Informationen entfernt, die die Hacker benötigen, werden Sie eine böse Überraschung `Log.x(..)` .

Entfernen Sie alle Ihre Loganrufe mit:

```
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    ...etc
}
```

entfernt zwar den Log-Aufruf selbst, aber normalerweise *nicht* die Strings, die Sie in sie einfügen.

Wenn Sie beispielsweise innerhalb Ihres Protokollaufrufs eine allgemeine Protokollnachricht `Log.d(MyTag,"Score="+score);` z. `Log.d(MyTag,"Score="+score);` ∴ `Log.d(MyTag,"Score="+score);` Der Compiler konvertiert das `+` außerhalb des Log-Aufrufs in einen neuen StringBuilder (). ProGuard ändert dieses neue Objekt nicht.

Ihr dekompilierter Code enthält weiterhin einen hängenden `StringBuilder` für `"Score="` , an den die verschleierte Version für die `score` Variable angehängt ist (angenommen, er wurde in `b` konvertiert).

Jetzt weiß der Hacker, was `b` , und macht seinen Code verständlich.

Um diese Residuen tatsächlich aus Ihrem Code zu entfernen, sollten Sie sie entweder gar nicht erst dort ablegen (Verwenden Sie stattdessen String-Formatierungsprogramm mit Proguard-Regeln, um sie zu entfernen) oder um Ihre `Log` Aufrufe mit folgenden Elementen zu versehen:

```
    if (BuildConfig.DEBUG) {
        Log.d(TAG,".."+var);
    }
```

Spitze:

Testen Sie, wie gut Ihr verschleierter Code geschützt ist, indem Sie ihn selbst dekompilieren!

1. dex2jar - wandelt die apk in jar um
2. jd - dekompiliert das Glas und öffnet es in einem GUI-Editor

Aktivieren von ProGuard mit einer benutzerdefinierten Verschleierungskonfigurationsdatei

Mit ProGuard kann der Entwickler seinen Code verschleiern, verkleinern und optimieren.

 # 1 Der erste Schritt des Verfahrens besteht darin, Proguard beim Build zu aktivieren .

Dies kann durch **Setzen des Befehls 'minifyEnabled'** für den gewünschten Build **auf true gesetzt** werden

 # 2 Als zweiten Schritt müssen Sie angeben, welche Proguard-Dateien für den angegebenen Build verwendet werden

Dies kann durch **Setzen der 'ProguardFiles'-Zeile mit den richtigen Dateinamen erfolgen**

```
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 Der Entwickler kann dann seine Proguard-Datei mit den von ihm gewünschten Regeln bearbeiten.

Dies kann durch Bearbeiten der Datei (z. B. 'proguard-rules-tests.pro') und Hinzufügen der

gewünschten Nebenbedingungen erfolgen. *Die folgende Datei dient als Beispiel für eine Proguard-Datei*

```
// 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
```

Immer wenn der Entwickler seine neue .APK-Datei ausführt und / oder generiert, werden die benutzerdefinierten Proguard-Konfigurationen angewendet, um die Anforderungen zu erfüllen.

ProGuard - Verschleiern und Verkleinern Ihres Codes online lesen:
https://riptutorial.com/de/android/topic/4500/proguard---verschleiern-und-verkleinern-ihres-codes

Kapitel 180: Projekt-SDK-Versionen

Einführung

Eine Android-Anwendung muss auf allen Arten von Geräten ausgeführt werden. Auf jedem Gerät kann eine andere Version von Android ausgeführt werden.

Nun unterstützt jede Android-Version möglicherweise nicht alle Funktionen, die für Ihre App erforderlich sind. Wenn Sie also eine App erstellen, müssen Sie die Mindest- und Maximalversion der Android-Version berücksichtigen.

Parameter

Parameter	Einzelheiten
SDK-Version	Die SDK-Version für jedes Feld ist die Ganzzahl der SDK-API-Ebene der Android-Version. Beispielsweise entspricht Froyo (Android 2.2) der API-Ebene 8. Diese Ganzzahlen werden auch in `Build.VERSION_CODES` definiert.

Bemerkungen

In jedem Projekt gibt es vier relevante SDK-Versionen:

- `targetSdkVersion` ist die neueste Version von Android, mit der Sie getestet haben.

  Das Framework verwendet `targetSdkVersion` , um zu bestimmen, wann bestimmte Kompatibilitätsverhalten aktiviert werden sollen. Durch das Targeting der API-Ebene 23 oder höher können Sie beispielsweise das Laufzeitberechtigungsmodell aktivieren .

- `minSdkVersion` ist die Mindestversion von Android, die Ihre Anwendung unterstützt. Benutzer, die eine ältere Android-Version als diese Version ausführen, können Ihre Anwendung weder installieren noch im Play Store anzeigen.

- `maxSdkVersion` ist die maximale Version von Android, die von Ihrer Anwendung unterstützt wird. Benutzer, die eine neuere Android-Version als diese Version ausführen, können Ihre Anwendung weder installieren noch im Play Store anzeigen. Dies sollte im Allgemeinen nicht verwendet werden, da die meisten Anwendungen mit neueren Android-Versionen ohne zusätzlichen Aufwand funktionieren.

- `compileSdkVersion` ist die Version des Android SDK, mit der Ihre Anwendung kompiliert wird. Es sollte im Allgemeinen die neueste Version von Android sein, die öffentlich veröffentlicht wurde. Dadurch wird festgelegt, auf welche APIs Sie beim Schreiben Ihres Codes zugreifen können. Sie können keine Methoden aufrufen, die in API-Ebene 23 eingeführt wurden, wenn Ihre `compileSdkVersion` auf 22 oder niedriger eingestellt ist.

Examples

Definieren von Projekt-SDK-Versionen

`build.gradle` in Ihrer `build.gradle` Datei des Hauptmoduls (**app**) Ihre Mindest- und Zielversionsnummer.

```
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
    }
}
```

Projekt-SDK-Versionen online lesen: https://riptutorial.com/de/android/topic/162/projekt-sdk-versionen

Kapitel 181: Protokollierung und Verwendung von Logcat

Syntax

- `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)`

Parameter

Möglichkeit	Beschreibung
-b (Puffer)	Lädt einen alternativen Protokollpuffer zur Anzeige, z. B. Ereignisse oder Radio. Der Hauptpuffer wird standardmäßig verwendet. Siehe Alternative Protokollpuffer anzeigen.
-c	Löscht das gesamte Protokoll und löscht es.
-d	Gibt das Protokoll auf dem Bildschirm aus und wird beendet.
-f (Dateiname)	Schreibt die Protokollnachricht in (Dateiname). Der Standardwert ist stdout.
-G	Gibt die Größe des angegebenen Protokollpuffers aus und wird beendet.
-n (zählen)	Legt die maximale Anzahl gedrehter Protokolle auf (Anzahl) fest. Der Standardwert ist 4. Erfordert die Option -r.
-r (kBytes)	Dreht die Protokolldatei alle (kBytes) der Ausgabe. Der Standardwert ist 16. Erfordert die Option -f.
-s	Setzt die Standardfilterspezifikation auf "stumm".
-v (Format)	Legt das Ausgabeformat für Protokollnachrichten fest. Der Standard ist das kurze Format.

Bemerkungen

Definition

Logcat ist ein Befehlszeilentool, das ein Protokoll der Systemnachrichten abbildet, einschließlich Stapelablaufverfolgungen, wenn das Gerät einen Fehler ausgibt, und Meldungen, die Sie mit der Protokollklasse aus Ihrer App geschrieben haben.

Wann verwenden?

Wenn Sie erwägen, die System.out-Methoden von Java zum Drucken an die Konsole zu verwenden, anstatt eine der Protokollmethoden von Android zu verwenden, sollten Sie wissen, dass sie grundsätzlich gleich funktionieren. Es ist jedoch besser, Java-Methoden nicht zu verwenden, da die zusätzlichen Informationen und Formatierungen, die von den Android-Protokollmethoden bereitgestellt werden, vorteilhafter sind. Außerdem werden die `Log.i()` System.out zur Methode `Log.i()` umgeleitet .

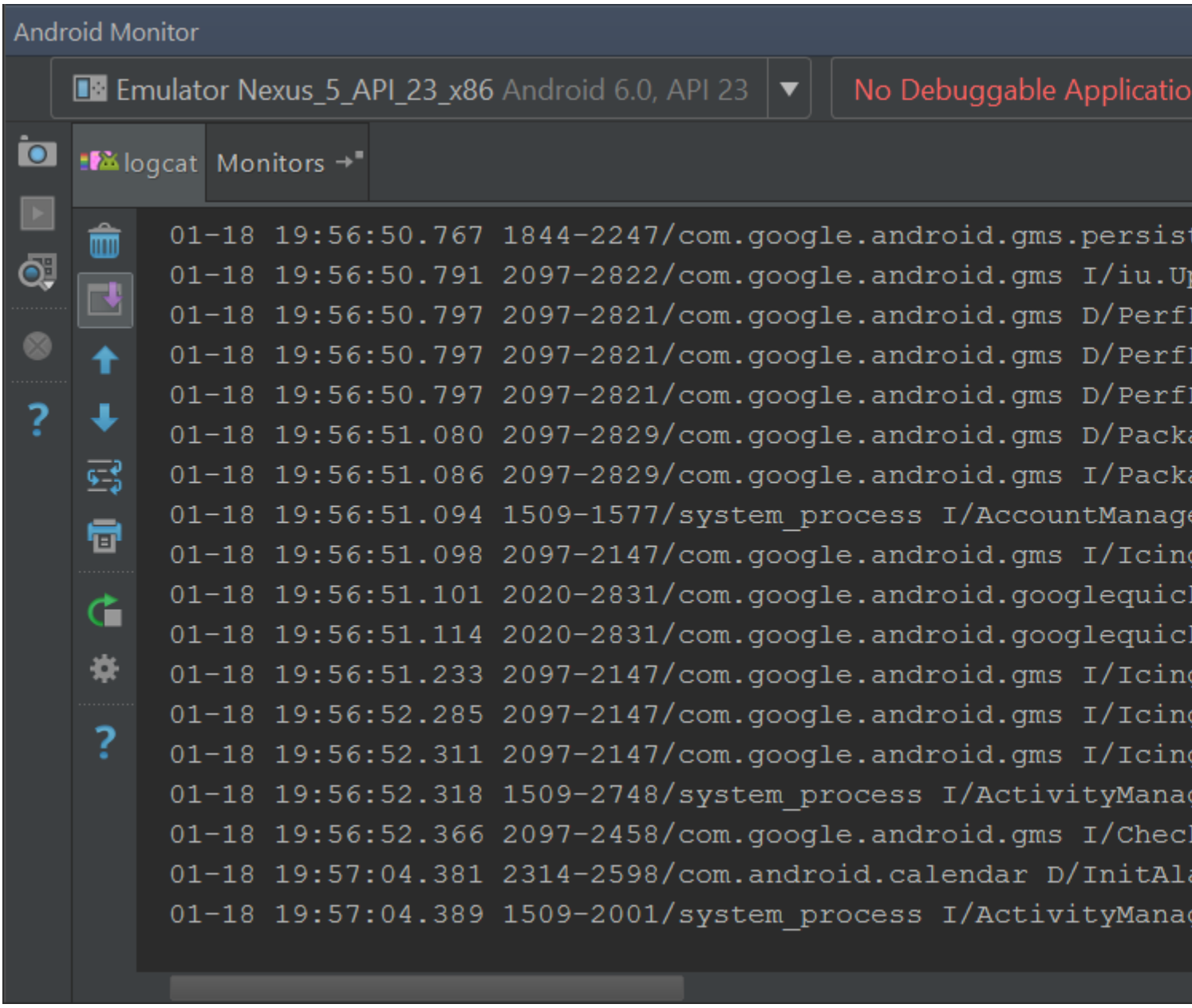
Nützliche Links

- Offizielle Dokumentation des Android-Entwicklers für Log und Logcat .
- Stackoveflow-Frage: Android Log.v (), Log.d (), Log.i (), Log.w (), Log.e () - Wann werden sie jeweils verwendet?

Examples

Filtern der Logcat-Ausgabe

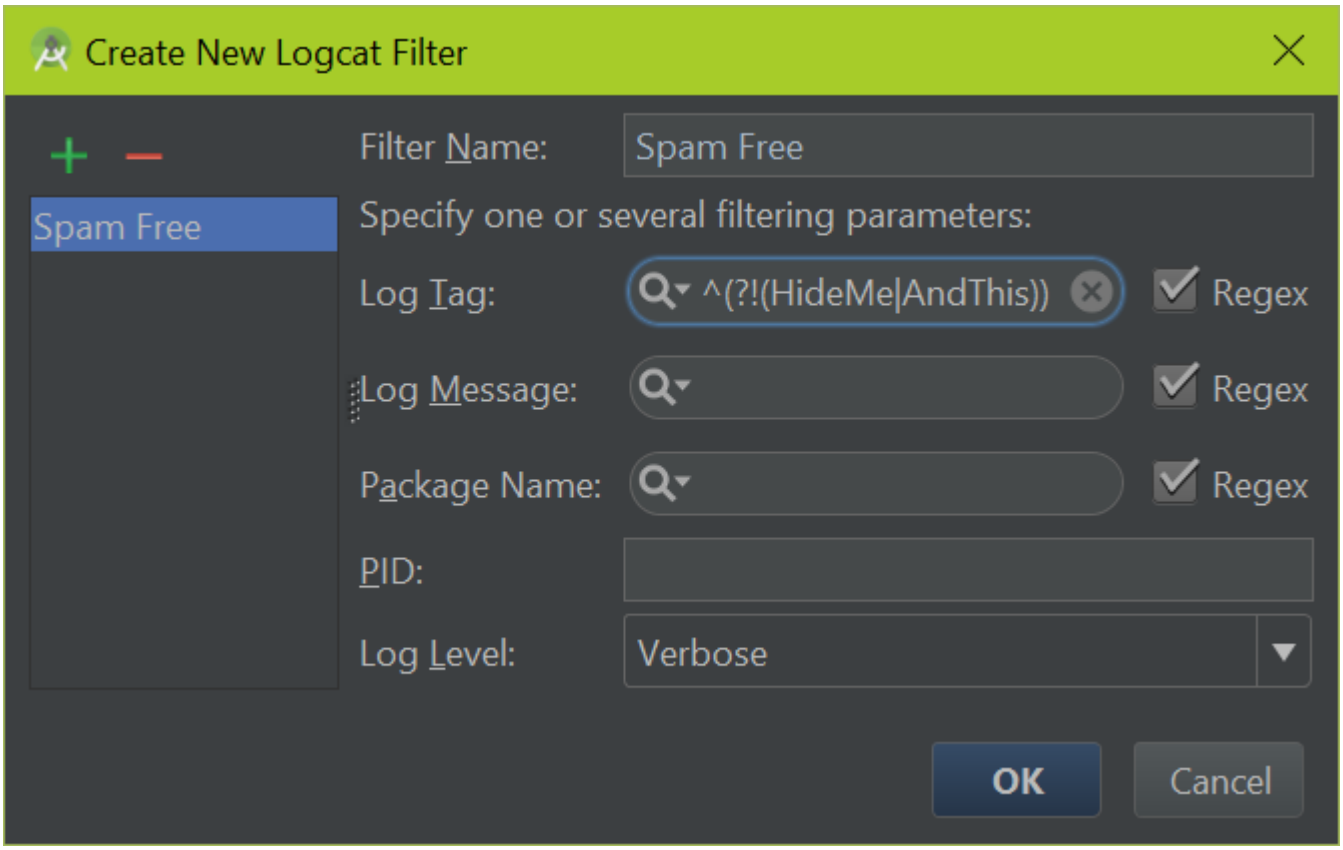
Es ist hilfreich, die Logcat-Ausgabe zu filtern, da es viele Meldungen gibt, die nicht von Interesse sind. Um die Ausgabe zu filtern, öffnen Sie den "Android Monitor", klicken Sie auf das Dropdown-Feld oben rechts und wählen Sie *Filterkonfiguration bearbeiten*



Jetzt können Sie benutzerdefinierte Filter hinzufügen, um interessierende Nachrichten anzuzeigen, und bekannte Protokollzeilen herausfiltern, die ignoriert werden können. Um einen Teil der Ausgabe zu ignorieren, können Sie einen regulären Ausdruck definieren. Hier ist ein Beispiel für das Ausschließen passender Tags:

```
^(?!(HideMe|AndThis))
```

Dies kann durch folgendes Beispiel eingegeben werden:



Das obige ist ein regulärer Ausdruck, der Eingaben ausschließt. Wenn Sie ein weiteres Tag zur *Blacklist* hinzufügen *möchten* , fügen Sie es nach einer Pipe `|` Charakter. Wenn Sie beispielsweise "GC" auf die schwarze Liste setzen möchten, verwenden Sie einen Filter wie diesen:

```
^(?!(HideMe|AndThis|GC))
```

Weitere Dokumentation und Beispiele finden Sie unter Protokollieren und Verwenden von Logcat

Protokollierung

Jede hochwertige Android-Anwendung wird anhand von Anwendungsprotokollen auf dem Laufenden halten. Diese Protokolle ermöglichen dem Entwickler eine einfache Debugging-Hilfe, um zu diagnostizieren, was mit der Anwendung passiert. Die vollständige Android-Dokumentation finden Sie hier , es folgt jedoch eine Zusammenfassung:

Grundlegende Protokollierung

Die `Log` ist die Hauptquelle für das Schreiben von Entwicklerprotokollen, indem ein `tag` und eine `message` . Mit diesem Tag können Sie Protokollnachrichten filtern, um zu ermitteln, welche Zeilen von Ihrer jeweiligen Aktivität stammen. Einfach anrufen

```
Log.v(String tag, String msg);
```

Das Android-System schreibt dem logcat eine Nachricht:

```
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
```

SPITZE:
Beachten Sie die Prozess-ID und die Thread-ID. Wenn sie gleich sind - das Protokoll kommt vom Haupt- / UI-Thread!

Jedes Tag kann verwendet werden, es ist jedoch üblich, den Klassennamen als Tag zu verwenden:

```
public static final String tag = MyAnimator.class.getSimpleName();
```

Log Levels

Der Android-Logger verfügt über 6 verschiedene Ebenen, die jeweils einen bestimmten Zweck erfüllen:

- `ERROR` : `Log.e()`
  - Wird verwendet, um einen kritischen Fehler anzuzeigen. Hierbei handelt es sich um die Ebene, in der beim Auslösen einer `Exception` gedruckt wird.
- `WARN` : `Log.w()`
  - Wird verwendet, um eine Warnung anzuzeigen, hauptsächlich für behebbare Fehler
- `INFO` : `Log.i()`
  - Wird verwendet, um Informationen über den Status der Anwendung auf höherer Ebene anzuzeigen
- `DEBUG` : `Log.d()`
  - Wird verwendet, um Informationen zu protokollieren, die beim Debuggen der Anwendung hilfreich sein könnten, beim Ausführen der Anwendung jedoch stören würden
- `VERBOSE` : `Log.v()`
  - Dient zum Protokollieren von Informationen, die die kleinen Details zum Status der Anwendung widerspiegeln
- `ASSERT` : `Log.wtf()`
  - Wird verwendet, um Informationen zu einer Bedingung zu protokollieren, die niemals auftreten sollte.
  - *wtf* steht für "What a Terrible Failure".

Motivation für die Protokollierung

Die Motivation für die Protokollierung besteht darin, Fehler, Warnungen und andere Informationen durch einen Blick auf die Ereigniskette der Anwendung leicht zu finden. Stellen Sie sich beispielsweise eine Anwendung vor, die Zeilen aus einer Textdatei liest, aber fälschlicherweise davon ausgeht, dass die Datei niemals leer ist. Die Protokollablaufverfolgung (einer App, die nicht protokolliert) würde in etwa wie folgt aussehen:

```
E/MyApplication: Process: com.example.myapplication, PID: 25788
                         com.example.SomeRandomException: Expected string, got 'null' instead
```

Es folgte eine Reihe von Stapelverfolgungsspuren, die schließlich zu der fehlerhaften Linie führen würden, wo das Durchlaufen eines Debuggers schließlich zu dem Problem führen würde

Der Protokollablauf einer Anwendung mit aktivierter Protokollierung könnte jedoch wie folgt aussehen:

```
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
```

Ein kurzer Blick auf die Protokolle und es ist offensichtlich, dass die Datei leer war.

Was Sie beim Logging beachten sollten:

Obwohl die Protokollierung ein leistungsfähiges Werkzeug ist, mit dem Android-Entwickler einen tieferen Einblick in die Funktionsweise ihrer Anwendung erhalten, hat die Protokollierung einige Nachteile.

Log-Lesbarkeit:

In Android-Anwendungen werden normalerweise mehrere Protokolle synchron ausgeführt. Daher ist es sehr wichtig, dass jedes Protokoll leicht lesbar ist und nur relevante, notwendige Informationen enthält.

Performance:

Für die Protokollierung sind nur wenige Systemressourcen erforderlich. Im Allgemeinen gibt dies keinen Anlass zur Sorge. Bei Überbeanspruchung kann jedoch die Protokollierung die Anwendungsleistung negativ beeinflussen.

Sicherheit:

In letzter Zeit wurden dem Google Play-Marktplatz mehrere Android-Anwendungen hinzugefügt, mit denen der Nutzer Protokolle aller laufenden Anwendungen anzeigen kann. Durch diese unbeabsichtigte Anzeige von Daten können Benutzer möglicherweise vertrauliche Informationen anzeigen. Entfernen Sie als Faustregel grundsätzlich immer Protokolle, die nicht öffentliche Daten enthalten, *bevor Sie* Ihre Anwendung auf dem Marktplatz veröffentlichen.

Fazit:

Die Protokollierung ist ein wesentlicher Bestandteil einer Android-Anwendung, da sie den Entwicklern eine hohe Leistung bietet. Die Möglichkeit, eine nützliche Protokollablaufverfolgung zu erstellen, ist einer der schwierigsten Aspekte der Softwareentwicklung. Die Protokollklasse von Android hilft jedoch dabei, die Protokollierung zu erleichtern.

Weitere Dokumentation und Beispiele finden Sie unter Protokollieren und Verwenden von Logcat

Melden Sie sich mit einem Link zur Quelle direkt aus Logcat

Dies ist ein schöner Trick, um einen Link zum Code hinzuzufügen, so dass es einfach ist, zu dem Code zu springen, der das Protokoll ausgegeben hat.

Mit folgendem Code diesen Aufruf:

```
MyLogger.logWithLink("MyTag","param="+param);
```

Wird darin enden, dass:

```
07-26...012/com.myapp D/MyTag: MyFrag:onStart(param=3)  (MyFrag.java:2366) // << logcat
converts this to a link to source!
```

Dies ist der Code (innerhalb einer Klasse namens MyLogger):

```
static StringBuilder sb0 = new StringBuilder(); // reusable string object

public static void logWithLink(String TAG, Object param) {
    StackTraceElement stack = Thread.currentThread().getStackTrace()[3];
    sb0.setLength(0);
    String c = stack.getFileName().substring(0, stack.getFileName().length() - 5); // removes
the ".java"
    sb0.append(c).append(":");
    sb0.append(stack.getMethodName()).append('(');
    if (param != null) {
        sb0.append(param);
    }
    sb0.append(") ");
    sb0.append("
(").append(stack.getFileName()).append(':').append(stack.getLineNumber()).append(')');
    Log.d(TAG, sb0.toString());
}
```

Dies ist ein einfaches Beispiel. Es kann leicht erweitert werden, um dem Anrufer eine Verknüpfung zu geben (Hinweis: Der Stapel wird [4] statt [3] sein), und Sie können auch andere relevante Informationen hinzufügen.

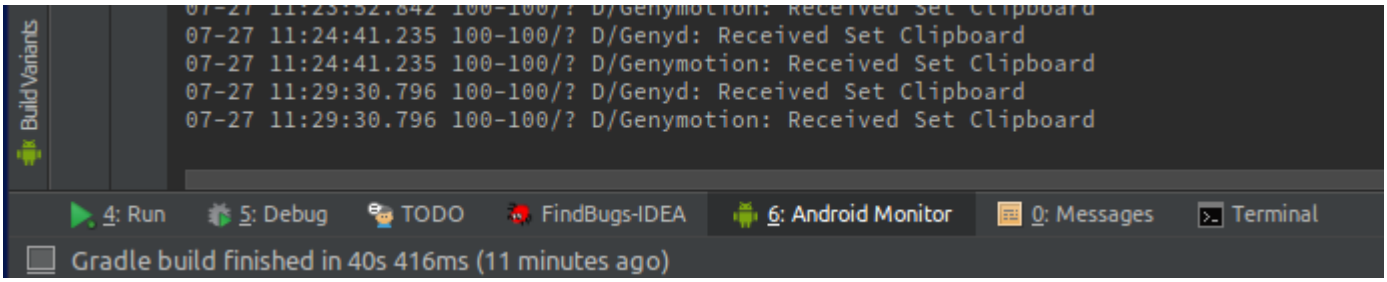
Logcat verwenden

Logcat ist ein Befehlszeilentool, das ein Protokoll der Systemnachrichten abbildet, einschließlich Stapelablaufverfolgungen, wenn das Gerät einen Fehler ausgibt, und Meldungen, die Sie mit der Protokollklasse aus Ihrer App geschrieben haben.

Die Logcat-Ausgabe kann im Android-Monitor von Android Studio oder mit der Adb-Befehlszeile angezeigt werden.

In Android Studio

Zeigen Sie durch Klicken auf das Symbol "Android Monitor":



Oder drücken Sie `Alt` + `6` unter Windows / Linux oder `CMD` + `6` unter Mac.

über die Kommandozeile:

Einfache Verwendung:

```
$ adb logcat
```

Mit Zeitstempeln:

```
$ adb logcat -v time
```

Nach bestimmten Texten filtern:

```
$ adb logcat -v time | grep 'searchtext'
```

Es gibt viele Optionen und Filter für das *Befehlszeilenprotokoll* , die hier dokumentiert sind .
Ein einfaches, aber nützliches Beispiel ist der folgende Filterausdruck, der alle Protokollnachrichten mit der Prioritätsstufe "error" für alle Tags anzeigt:

```
$ adb logcat *:E
```

Protokollierungscode generieren

`Live templates Android Studio` bieten eine Reihe von Verknüpfungen für die schnelle Protokollierung.
Um Live-Vorlagen zu verwenden, müssen Sie nur den Namen der Vorlage eingeben und die `TAB` oder die Eingabetaste `TAB` , um die Anweisung einzufügen.

Beispiele:

- `logi` → wird zu → `android.util.Log.i(TAG, "$METHOD_NAME$: $content$");`
    - `$METHOD_NAME$` wird automatisch durch Ihren Methodennamen ersetzt und der Cursor wartet, bis der Inhalt gefüllt ist.
- `loge` → gleich, für Fehler
- usw. für den Rest der Protokollierungsstufen.

Eine vollständige Liste der Vorlagen finden Sie in den Einstellungen von `Android Studio` (`ALT` + `s` und Typ "live"). Außerdem können Sie Ihre benutzerdefinierten Vorlagen hinzufügen.

Wenn die `Live templates Android Studio` für Ihre Anforderungen nicht ausreichen, können Sie das Android Postfix-Plugin in Betracht ziehen

Dies ist eine sehr nützliche Bibliothek, mit der Sie die Protokollzeile nicht manuell schreiben müssen.

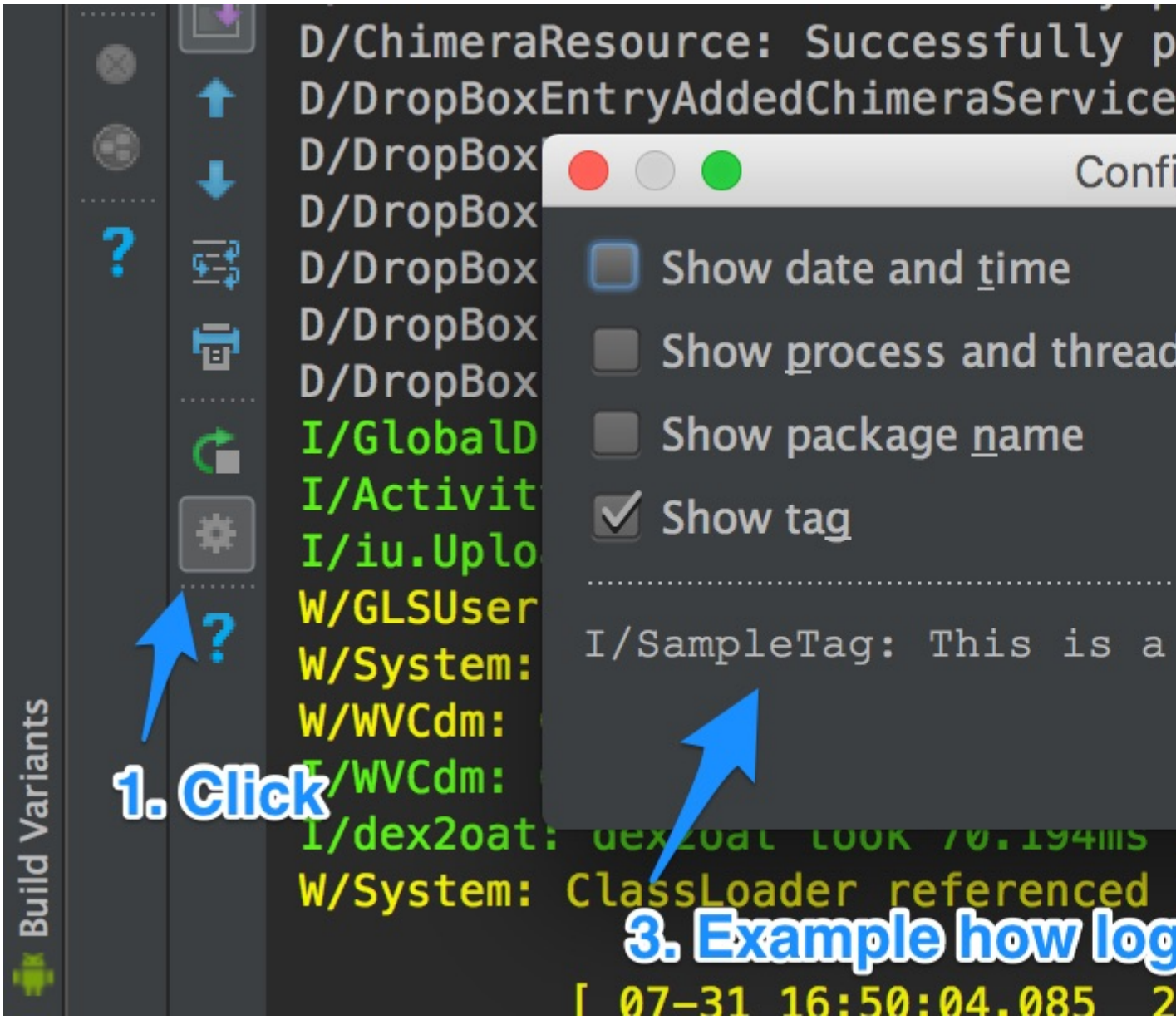
Die Syntax ist absolut einfach:

.log - Protokollierung. Wenn die Variable "TAG" konstant ist, wird "TAG" verwendet. Sonst verwenden Sie den Klassennamen.

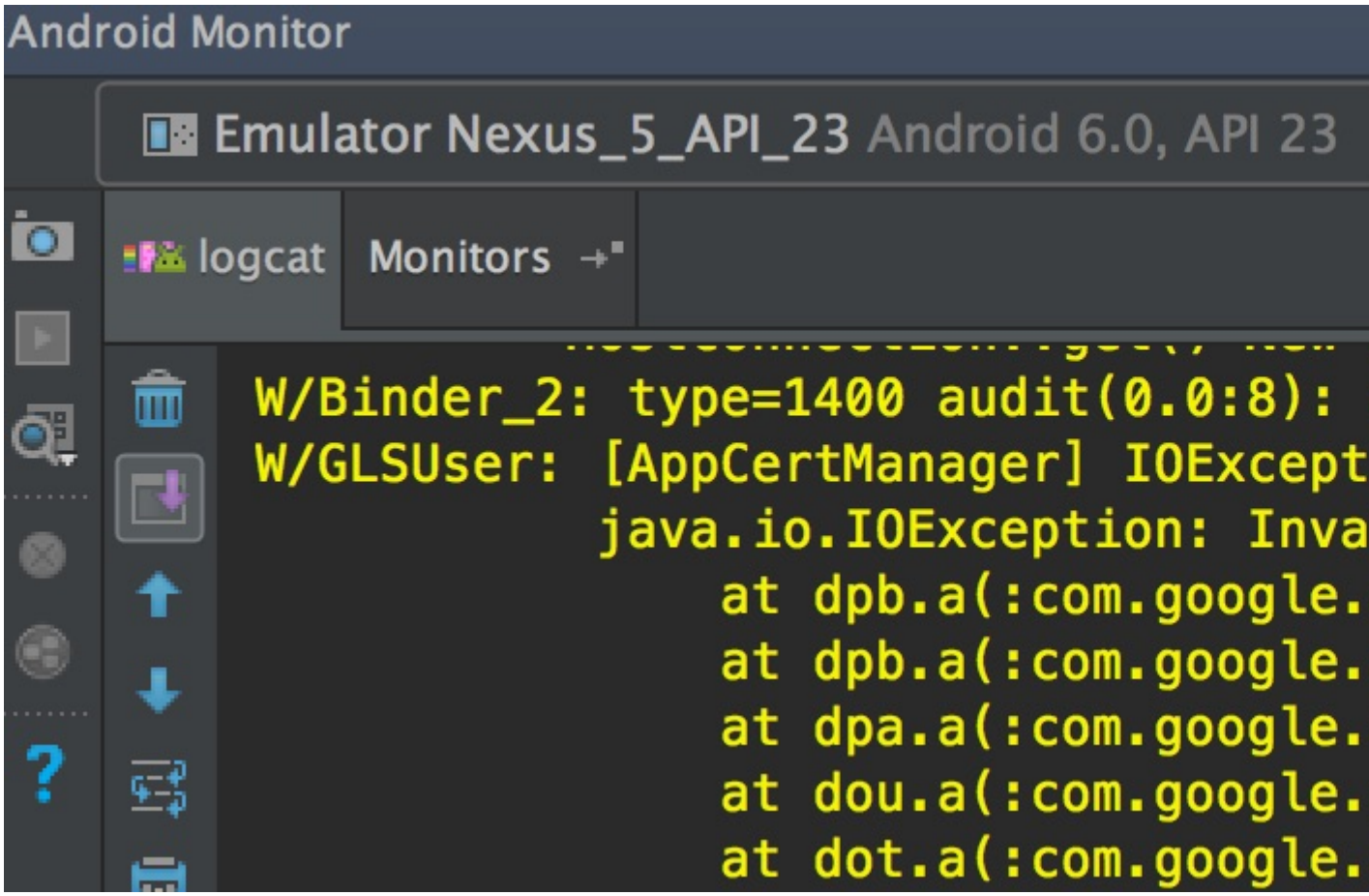
```
ublic class MyActivity extends Activity {
  static final String T= "test";
  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      new View.OnClickListener() {
          @Override
          public void onClick(View view) {

          }
      };
  }
```

Android Studio-Nutzung

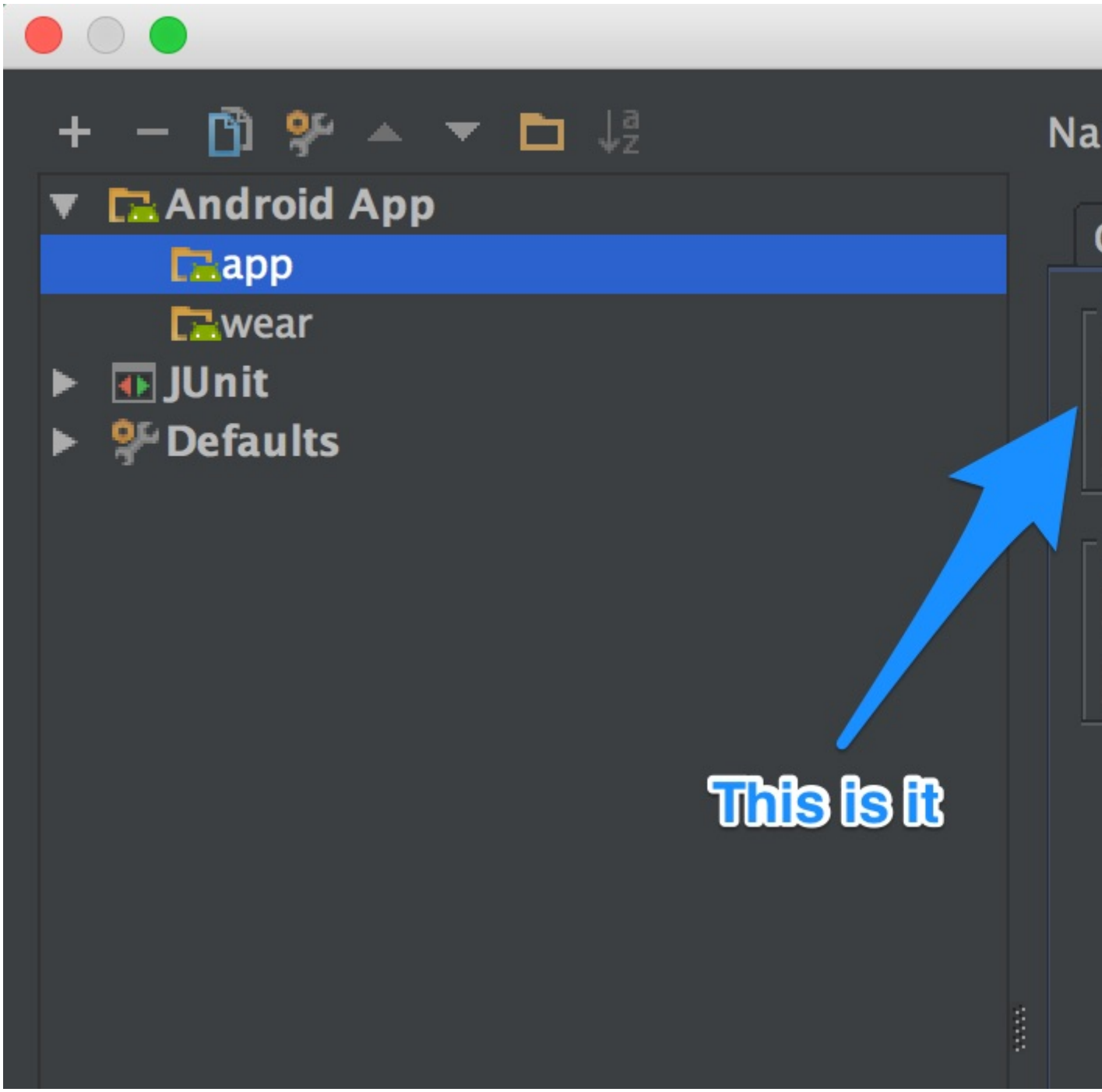
1. Gedruckte Informationen ausblenden / anzeigen:



2. Kontrollieren Sie die Ausführlichkeit der Protokollierung:



3. Das Öffnen des Protokollfensters beim Starten der Run / Debug-Anwendung deaktivieren / aktivieren



Protokolle löschen

Um das gesamte Protokoll zu löschen (leeren):

```
adb logcat -c
```

Protokollierung und Verwendung von Logcat online lesen:
https://riptutorial.com/de/android/topic/1552/protokollierung-und-verwendung-von-logcat

Kapitel 182: RecyclerView

Einführung

RecyclerView ist eine erweiterte Version der Listenansicht mit verbesserter Leistung und zusätzlichen Funktionen.

Parameter

Parameter	Detail
Adapter	Eine Unterklasse von RecyclerView.Adapter, die für das Bereitstellen von Ansichten verantwortlich ist, die Elemente in einem Datensatz darstellen
Position	Die Position eines Datenelements in einem Adapter
Index	Der Index einer angefügten untergeordneten Ansicht, wie er in einem Aufruf von getChildAt (int) verwendet wird. Kontrast zur Position
Bindung	Der Vorgang des Vorbereitens einer untergeordneten Ansicht zum Anzeigen von Daten, die einer Position im Adapter entsprechen
Recycling (Ansicht)	Eine zuvor zum Anzeigen von Daten für eine bestimmte Adapterposition verwendete Ansicht kann zur späteren Wiederverwendung in einen Cache gestellt werden, um später denselben Datentyp wieder anzuzeigen. Dies kann die Leistung drastisch verbessern, indem die anfängliche Layoutinflation oder -konstruktion übersprungen wird
Schrott (Ansicht)	Eine untergeordnete Ansicht, die während des Layouts vorübergehend getrennt wurde. Ausschnittsansichten können wiederverwendet werden, ohne dass sie vollständig von der übergeordneten RecyclerView getrennt werden, entweder unverändert, wenn keine erneute Bindung erforderlich ist, oder vom Adapter geändert, wenn die Ansicht als verschmutzt gilt
Dirty (Ansicht)	Eine untergeordnete Ansicht, die vom Adapter vor der Anzeige neu gebunden werden muss

Bemerkungen

Die `RecyclerView` ist eine flexible Ansicht, um ein begrenztes Fenster für einen großen Datensatz bereitzustellen.

Bevor Sie `RecyclerView` , müssen Sie die Abhängigkeit der Unterstützungsbibliothek in der Datei `build.gradle` :

```
dependencies {
    // Match the version of your support library dependency
    compile 'com.android.support:recyclerview-v7:25.3.1'
}
```

Die neueste Versionsnummer des Recyclingberichts finden Sie auf der offiziellen Website .

Andere verwandte Themen:

Es gibt andere Themen, die die `RecyclerView` Komponenten beschreiben:

- RecyclerView LayoutManagers
- RecyclerView ItemDekorations
- RecyclerView onClickListeners

Offizielle Dokumentation

http://developer.android.com/reference/android/support/v7/widget/RecyclerView.html

Ältere Versionen:

```
  //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 hinzufügen

Fügen Sie die Abhängigkeit wie im Abschnitt Anmerkung beschrieben hinzu, und fügen Sie Ihrem Layout dann eine `RecyclerView` :

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

Nachdem Sie Ihrem Layout ein `RecyclerView` Widget hinzugefügt haben, erhalten Sie einen Handle für das Objekt, verbinden Sie es mit einem Layout-Manager und hängen Sie einen Adapter für die anzuzeigenden Daten an:

```
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);
```

Oder richten Sie den Layout-Manager einfach aus XML ein, indem Sie folgende Zeilen hinzufügen:

```
xmlns:app="http://schemas.android.com/apk/res-auto"
app:layoutManager="android.support.v7.widget.LinearLayoutManager"
```

Wenn Sie wissen , dass Änderungen in Inhalt der `RecyclerView` nicht die Layoutgröße des ändern `RecyclerView` , verwenden Sie den folgenden Code , um die Leistung des Bauteils zu verbessern. Wenn RecyclerView eine feste Größe hat, weiß es, dass RecyclerView aufgrund seiner untergeordneten Elemente nicht die Größe selbst ändert, sodass das Anforderungslayout überhaupt nicht aufgerufen wird. Es kümmert sich einfach um die Änderung selbst. Wenn ungültig ist, was auch immer das übergeordnete Element ist, der Koordinator, das Layout oder was auch immer. (Sie können diese Methode auch verwenden, bevor Sie `LayoutManager` und `Adapter` einstellen.)

```
mRecyclerView.setHasFixedSize(true);
```

`RecyclerView` stellt diese integrierten Layout-Manager zur Verfügung. So können Sie mit `RecyclerView` eine Liste, ein Raster und ein gestaffeltes Raster erstellen:

1. LinearLayoutManager zeigt Elemente in einer vertikalen oder horizontalen Bildlaufliste an.
2. GridLayoutManager zeigt Elemente in einem Raster an.

3. StaggeredGridLayoutManager zeigt Elemente in einem versetzten Raster an.

Glatteres Laden von Artikeln

Wenn die Elemente in Ihrem `RecyclerView` Daten aus dem Netzwerk laden (in der Regel Bilder) oder andere Verarbeitungsschritte ausführen, kann dies sehr viel Zeit in `RecyclerView` nehmen und Sie können Elemente auf dem Bildschirm erhalten, die jedoch nicht vollständig geladen sind. Um dies zu vermeiden, können Sie den vorhandenen `LinearLayoutManager` , um eine Reihe von Elementen vorab zu `LinearLayoutManager` , bevor sie auf dem Bildschirm angezeigt werden:

```
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;
  }
}
```

Drag & Drop und Streichen mit RecyclerView

Mit dem RecyclerView können Sie die Funktionen zum Abwischen und Ziehen und Ablegen implementieren, ohne Fremdanbieter-Bibliotheken zu verwenden.
Verwenden `ItemTouchHelper` einfach die `ItemTouchHelper` Klasse, die in der RecyclerView-Support-Bibliothek enthalten ist.

Instantiieren Sie den `ItemTouchHelper` mit dem `SimpleCallback` Callback. Abhängig von der unterstützten Funktionalität sollten Sie `onMove(RecyclerView, ViewHolder, ViewHolder)` und / oder `onSwiped(ViewHolder, int)` und schließlich mit Ihrem `RecyclerView` `onSwiped(ViewHolder, int)` .

```
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);
```

Es ist erwähnenswert, dass der `SimpleCallback` Konstruktor auf alle Elemente in der `RecyclerView` dieselbe Wischstrategie anwendet. In jedem Fall ist es möglich, die Standard-Wischrichtung für bestimmte Elemente zu aktualisieren, indem Sie einfach die Methode `getSwipeDirs(RecyclerView, ViewHolder)` .

Nehmen wir zum Beispiel an, dass unsere `RecyclerView` einen `HeaderViewHolder` und wir möchten natürlich keinen Swip-Effekt darauf anwenden. Es reicht aus, `getSwipeDirs` wie folgt zu

überschreiben:

```
@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);
}
```

Kopf- / Fußzeile zu einer RecyclerView hinzufügen

Dies ist ein Beispieladaptercode.

```
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
    }
}
}
```

Hier ist eine gute Lektüre über die Implementierung von `RecyclerView` mit Kopf- und Fußzeile.

Alternative Methode:

Während die obige Antwort arbeiten Sie diesen Ansatz verwenden können , als auch eine Recycler Ansicht unter Verwendung eines mit `NestedScrollView` .Sie ein Layout für Header hinzufügen können Sie den folgenden Ansatz:

```
<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>
```

Sie können auch ein `LinearLayout` mit vertikaler Ausrichtung in Ihrer `NestedScrollView` .

Hinweis: Dies funktioniert nur mit `RecyclerView` über **23.2.0**

```
compile 'com.android.support:recyclerview-v7:23.2.0'
```

Mehrere ViewHolders mit ItemViewType verwenden

In einigen Fällen müssen für RecyclerView verschiedene Arten von Ansichten verwendet werden, um in der Liste auf der Benutzeroberfläche angezeigt zu werden. Für jede Ansicht muss ein anderes Layout-XML-Format zum Aufblasen verwendet werden.

Für dieses Problem können Sie verschiedene ViewHolders in einem einzelnen Adapter verwenden, indem Sie in RecyclerView eine spezielle Methode verwenden - `getItemViewType(int position)` .

Unten sehen Sie ein Beispiel für die Verwendung von zwei ViewHolders:

1. Ein ViewHolder zum Anzeigen von Listeneinträgen
2. Ein ViewHolder zum Anzeigen mehrerer Kopfansichten

```
@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.
    }
}
```

Elemente in RecyclerView mit einer SearchView filtern

`filter` in `RecyclerView.Adapter` hinzufügen:

```
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` wird im Konstruktor des Adapters wie `itemsCopy.addAll(items)` initialisiert.

Wenn Sie dies tun, rufen Sie einfach den `filter` aus `OnQueryTextListener` aus `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;
    }
});
```

Popup-Menü mit RecyclerView

Fügen Sie diesen Code in Ihren ViewHolder ein

Anmerkung: In diesem Code verwende ich `btnExpand` click-event. Für das gesamte `recyclerview` click-Ereignis können Sie den Listener auf itemView-Objekt setzen.

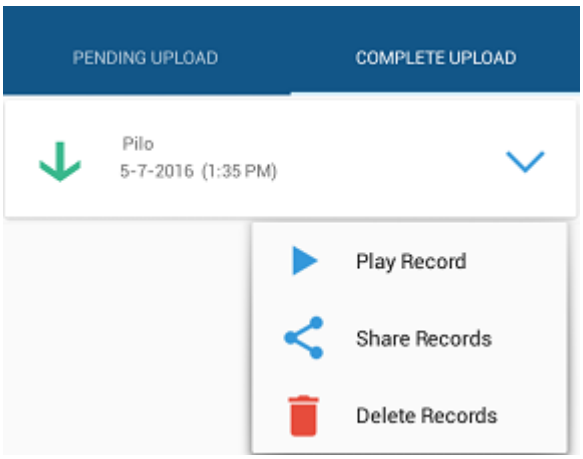
```
public class MyViewHolder extends RecyclerView.ViewHolder{
        CardView cv;
        TextView recordName, visibleFile, date, time;
        Button btnIn, btnExpand;

        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();
                }
            });

        }
    }
```

alternative Möglichkeit, Symbole im Menü anzuzeigen

```
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) {

}
```

Hier ist die Ausgabe:



Datenänderung animieren

`RecyclerView`

führt eine relevante Animation aus, wenn eine der Benachrichtigungsmethoden verwendet wird, mit Ausnahme von `notifyDataSetChanged` . Dazu gehören `notifyItemChanged` , `notifyItemInserted` , `notifyItemMoved` , `notifyItemRemoved` usw.

Der Adapter sollte diese Klasse anstelle von `RecyclerView.Adapter` .

```
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;

import java.util.List;

public abstract class AnimatedRecyclerAdapter<T, VH extends RecyclerView.ViewHolder>
        extends RecyclerView.Adapter<VH> {
    protected List<T> models;

    protected AnimatedRecyclerAdapter(@NonNull List<T> models) {
        this.models = models;
    }

    //Set new models.
    public void setModels(@NonNull final List<T> models) {
        applyAndAnimateRemovals(models);
        applyAndAnimateAdditions(models);
        applyAndAnimateMovedItems(models);
    }

    //Remove an item at position and notify changes.
    private T removeItem(int position) {
        final T model = models.remove(position);
        notifyItemRemoved(position);
        return model;
    }

    //Add an item at position and notify changes.
    private void addItem(int position, T model) {
        models.add(position, model);
        notifyItemInserted(position);
    }

    //Move an item at fromPosition to toPosition and notify changes.
    private void moveItem(int fromPosition, int toPosition) {
        final T model = models.remove(fromPosition);
        models.add(toPosition, model);
        notifyItemMoved(fromPosition, toPosition);
    }

    //Remove items that no longer exist in the new models.
    private void applyAndAnimateRemovals(@NonNull final List<T> newTs) {
        for (int i = models.size() - 1; i >= 0; i--) {
            final T model = models.get(i);
            if (!newTs.contains(model)) {
                removeItem(i);
            }
        }
    }

    //Add items that do not exist in the old models.
    private void applyAndAnimateAdditions(@NonNull final List<T> newTs) {
        for (int i = 0, count = newTs.size(); i < count; i++) {
            final T model = newTs.get(i);
```

```
            if (!models.contains(model)) {
                addItem(i, model);
            }
        }
    }

    //Move items that have changed their position.
    private void applyAndAnimateMovedItems(@NonNull final List<T> newTs) {
        for (int toPosition = newTs.size() - 1; toPosition >= 0; toPosition--) {
            final T model = newTs.get(toPosition);
            final int fromPosition = models.indexOf(model);
            if (fromPosition >= 0 && fromPosition != toPosition) {
                moveItem(fromPosition, toPosition);
            }
        }
    }
}
```

> Sie sollten **NICHT** dieselbe `List` für `setModels` und `List` im Adapter verwenden.

Sie deklarieren `models` als globale Variablen. `DataModel` ist nur eine Dummy-Klasse.

```
private List<DataModel> models;
private YourAdapter adapter;
```

`models` initialisieren `models` bevor sie an den Adapter übergeben werden. `YourAdapter` ist die Implementierung von `AnimatedRecyclerAdapter` .

```
models = new ArrayList<>();
//Add models
models.add(new DataModel());
//Do NOT pass the models directly. Otherwise, when you modify global models,
//you will also modify models in adapter.
//adapter = new YourAdapter(models); <- This is wrong.
adapter = new YourAdapter(new ArrayList(models));
```

Rufen Sie dies auf, nachdem Sie Ihre globalen `models` aktualisiert haben.

```
adapter.setModels(new ArrayList(models));
```

> Wenn Sie nicht `equals` überschreiben, wird der gesamte Vergleich anhand der Referenz verglichen.

Beispiel mit SortedList

`SortedList` nach der Einführung von `RecyclerView` führte Android die `SortedList` Klasse ein. Diese Klasse behandelt alle Aufrufe der Benachrichtigungsmethode an `RecyclerView.Adapter` , um eine ordnungsgemäße Animation zu gewährleisten, und ermöglicht sogar das Bündeln mehrerer Änderungen, damit die Animationen nicht verwackeln.

```
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 mit DataBinding

Hier ist eine generische ViewHolder-Klasse, die Sie mit jedem DataBinding-Layout verwenden

können. Hier wird eine Instanz einer bestimmten `ViewDataBinding` Klasse mithilfe des aufgeblasenen `View` Objekts und der `DataBindingUtil` Dienstprogrammklasse erstellt.

```
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;
    }
}
```

Nachdem Sie diese Klasse erstellt haben, können Sie das `<layout>` in Ihrer Layoutdatei verwenden, um die Datenbindung für dieses Layout wie folgt zu aktivieren:

`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>
```

und hier ist dein Beispiel dataModel:

```
public class ItemModel {
    public String itemLabel;
}
```

Standardmäßig generiert die Bibliothek für Android Data Binding eine `ViewDataBinding` Klasse basierend auf dem Namen der Layoutdatei, konvertiert sie in den Pascal-Fall und fügt "Binding" hinzu. In diesem Beispiel wäre dies `MyItemBinding` für die Layoutdatei `my_item.xml` . Diese Binding-Klasse verfügt auch über eine Setter-Methode, um das als Daten definierte Objekt in der Layout-

Datei `ItemModel` (`ItemModel` diesem Beispiel `ItemModel`).

Nun, da wir alle Teile haben, können wir unseren Adapter folgendermaßen implementieren:

```
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();
    }
}
```

Endloses Scrollen in Recycleview.

Hier habe ich ein Code-Snippet zur Implementierung des endlosen Bildlaufs in der Recycling-Ansicht bereitgestellt.

Schritt 1: Erstellen Sie zunächst eine einzige abstrakte Methode in Recycleview-Adapter (siehe unten).

```
public abstract class ViewAllCategoryAdapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    public abstract void load();
}
```

Schritt 2: Jetzt können Sie die onBindViewHolder- und `getItemCount()` Methode der ViewAllCategoryAdapter-Klasse überschreiben und die `Load()` -Methode wie folgt aufrufen.

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, final int position) {
    if ((position >= getItemCount() - 1)) {
        load();
    }
}

@Override
public int getItemCount() {
    return YOURLIST.size();
```

```
}
```

Schritt 3: Jetzt ist jede Backend-Logik abgeschlossen. Jetzt ist es an der Zeit, diese Logik auszuführen. Es ist ganz einfach, die Lademethode zu überschreiben, in der Sie ein Objekt Ihres Adapters erstellen.

```
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);
```

Jetzt wird die `load()` -Methode automatisch aufgerufen, während der Benutzer am Ende der Liste blättert.

Bestes Glück

Standardansicht anzeigen, bis Elemente geladen werden oder wenn Daten nicht verfügbar sind

Bildschirmfoto

Adapterklasse

```
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>
```

Ich verwende FontAwesome mit Iconics Library für die Bilder. Fügen Sie dies der Build.gradle-Datei auf App-Ebene hinzu.

```
compile 'com.mikepenz:fontawesome-typeface:4.6.0.3@aar'
compile 'com.mikepenz:iconics-core:2.8.1@aar'
```

Hinzufügen von Trennlinien zu RecyclerView-Elementen

Fügen Sie diese Zeilen einfach zur Initialisierung hinzu

```
RecyclerView mRecyclerView = (RecyclerView) view.findViewById(recyclerView);
mRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
mRecyclerView.addItemDecoration(new DividerItemDecoration(getActivity(),
DividerItemDecoration.VERTICAL));
```

Fügen Sie einen `adapter` und rufen Sie `.notifyDataSetChanged();` wie gewöhnlich !
Dies ist keine integrierte Funktion von Recyclerview, sondern wurde in den Support-Bibliotheken hinzugefügt. Vergessen Sie nicht, dies in Ihre Build.gradle-Datei auf App-Ebene aufzunehmen

```
compile "com.android.support:appcompat-v7:25.3.1"
```

```
compile "com.android.support:recyclerview-v7:25.3.1"
```

Zu einer einzelnen RecyclerView können mehrere ItemDecorations hinzugefügt werden.

Ändern der Teilerfarbe :

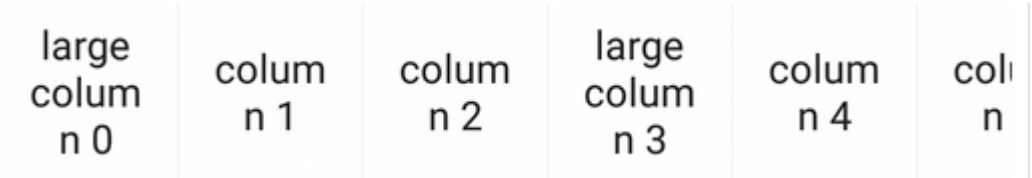
Es ist ziemlich einfach, eine Farbe für eine itemDecoration festzulegen.

1. Schritt ist: Erstellen einer `divider.xml` Datei, die sich im `drawable` Ordner befindet

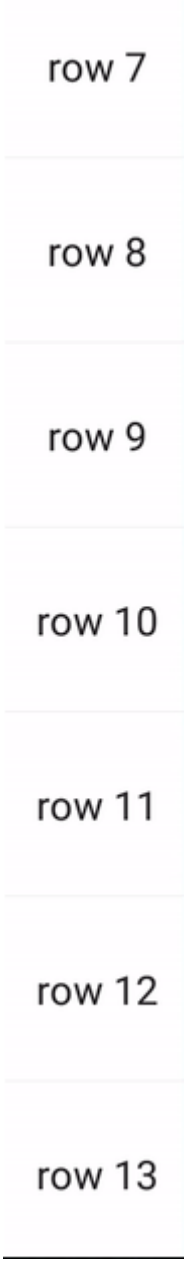
```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="line">
    <size
        android:width="1px"
        android:height="1px"/>
    <solid android:color="@color/divider_color"/>
</shape>
```

2. Schritt ist: Einstellung zeichnen

```
    // 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);
```



RecyclerView online lesen: https://riptutorial.com/de/android/topic/169/recyclerview

Kapitel 183: RecyclerView onClickListeners

Examples

Neues Beispiel

```
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 und RxJava Beispiel

Erstes Beispiel in Kotlin neu implementiert und RxJava für sauberere Interaktion verwendet.

```
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
        }
    }
}
```

Die Verwendung ist dann ganz einfach. Es ist möglich, einen separaten Thread mit RxJava-Funktionen zu abonnieren.

```
val adapter = SampleAdapter(arrayOf("Hello", "World"))
adapter.itemClickStream.subscribe { v ->
    if (v.id == R.id.textView) {
        // do something
    }
}
```

Easy OnLongClick und OnClick-Beispiel

Implementieren Sie zunächst Ihren View-Inhaber:

```
implements View.OnClickListener, View.OnLongClickListener
```

Dann registrieren Sie die Hörer wie folgt:

```
itemView.setOnClickListener(this);
itemView.setOnLongClickListener(this);
```

Als nächstes überschreiben Sie die Zuhörer wie folgt:

```
@Override
public void onClick(View v) {
    onclicklistner.onItemClick(getAdapterPosition(), v);
}

@Override
public boolean onLongClick(View v) {
    onclicklistner.onItemLongClick(getAdapterPosition(), v);
    return true;
}
```

Fügen Sie schließlich folgenden Code hinzu:

```
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);
}
```

Adapter-Demo

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

Der obige Beispielcode kann mit dem folgenden Code aufgerufen werden:

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

Element Klicken Sie auf Listener

Um einen Elementklicklistener und / oder einen Elementklicklistener zu implementieren, können Sie eine Schnittstelle in Ihrem Adapter erstellen:

```
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;
    }

}
```

Dann können Sie Ihre Klicklistener festlegen, nachdem Sie eine Instanz des Adapters erstellt haben:

```
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;
    }
});
```

Eine andere Möglichkeit zum Implementieren von Item Click Listener

Eine andere Möglichkeit zum Implementieren von Elementklicklistenern ist die Verwendung der Schnittstelle mit mehreren Methoden, deren Anzahl der Anzahl anklickbarer Ansichten entspricht, und die Verwendung von überschriebenen Klicklisten, wie unten gezeigt. Diese Methode ist flexibler, da Sie die Click-Listener auf verschiedene Ansichten einstellen und die Klick-Logik ganz einfach für jede steuern können.

```
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);
        }
    }
}
```

Wenn Sie über eine Adapterinstanz verfügen, können Sie Ihren Klicklistener festlegen, der darauf wartet, auf jede der Ansichten zu klicken:

```
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 Klicken Sie auf Listener

```
public  class RecyclerTouchListener implements RecyclerView.OnItemTouchListener {

    private GestureDetector gestureDetector;
    private RecyclerTouchListener.ClickListener clickListener;

    public RecyclerTouchListener(Context context, final RecyclerView recyclerView, final
RecyclerTouchListener.ClickListener clickListener) {
        this.clickListener = clickListener;

        gestureDetector = new GestureDetector(context, new
GestureDetector.SimpleOnGestureListener() {
            @Override
            public boolean onSingleTapUp(MotionEvent e) {
                return true;
            }
            @Override
            public void onLongPress(MotionEvent e) {
                View child = recyclerView.findChildViewUnder(e.getX(), e.getY());
                if (child != null && clickListener != null) {
                    clickListener.onLongClick(child, recyclerView.getChildPosition(child));
                }
            }
        });
    }


    @Override
    public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {
        View child = rv.findChildViewUnder(e.getX(), e.getY());
        if (child != null && clickListener != null && gestureDetector.onTouchEvent(e)) {
            clickListener.onClick(child, rv.getChildPosition(child));
        }
        return false;
    }

    @Override
    public void onTouchEvent(RecyclerView rv, MotionEvent e) {

    }

    @Override
    public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {

    }

    public interface ClickListener {
        void onLongClick(View child, int childPosition);

        void onClick(View child, int childPosition);
    }
}
```

In der Hauptaktivität

```
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) {


     }
    }));
```

RecyclerView onClickListeners online lesen:
https://riptutorial.com/de/android/topic/96/recyclerview-onclicklisteners

Kapitel 184: RecyclerView und LayoutManager

Examples

GridLayoutManager mit dynamischer Spannenzahl

Beim Erstellen eines Recyclerview mit einem Gridlayout-Layout-Manager müssen Sie die Spannweitenanzahl im Konstruktor angeben. Die Spannweite bezieht sich auf die Anzahl der Spalten. Dies ist ziemlich klobig und berücksichtigt keine größeren Bildschirmgrößen oder Bildschirmausrichtungen. Ein Ansatz besteht darin, mehrere Layouts für die verschiedenen Bildschirmgrößen zu erstellen. Ein weiterer dynamischerer Ansatz ist unten zu sehen.

Zuerst erstellen wir eine benutzerdefinierte RecyclerView-Klasse wie folgt:

```
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);
        }
    }
}
```

Diese Klasse bestimmt, wie viele Spalten in das Recycling-Fenster passen. Um es zu verwenden, müssen Sie es wie folgt in Ihre layout.xml einfügen:

```
<?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"
    />
```

Beachten Sie, dass wir das columnWidth-Attribut verwenden. Die Recyclingübersicht benötigt es, um festzustellen, wie viele Spalten in den verfügbaren Platz passen.

In Ihrer Aktivität / Ihrem Fragment erhalten Sie lediglich einen Verweis auf das Recylerview und legen einen Adapter darauf (und alle Elementdekorationen oder Animationen, die Sie hinzufügen möchten) fest. **SETZEN SIE KEINEN LAYOUT MANAGER**

```
RecyclerView recyclerView = (RecyclerView) findViewById(R.id.auto_fit_recycler_view);
recyclerView.setAdapter(new MyAdapter());
```

(wobei MyAdapter Ihre Adapterklasse ist)

Sie haben jetzt eine Übersicht, in der der Spancount (dh die Spalten) an die Bildschirmgröße angepasst wird. Als letzte Ergänzung möchten Sie möglicherweise die Spalten im Recycling-Fenster zentrieren (standardmäßig sind sie an layout_start ausgerichtet). Sie können dies tun, indem Sie die AutofitRecyclerView-Klasse ein wenig ändern. Beginnen Sie mit der Erstellung einer inneren Klasse in der Recyclingübersicht. Dies ist eine Klasse, die sich von GridLayoutManager erstreckt. Es wird links und rechts genug aufgefüllt, um die Zeilen zu zentrieren:

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

Wenn Sie den LayoutManager in AutofitRecyclerView festlegen, verwenden Sie den CenteredGridLayoutManager wie folgt:

```
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);
}
```

Und das ist es! Sie verfügen über eine dynamische, zentrierte Rasterlayoutmanager-basierte Recyclingübersicht.

Quellen:

- Chiu-Ki Chans Platzinsel-Blog
- Paketüberfluss

Headeransicht zum Recycling hinzufügen mit dem Gridlayout-Manager

Um einem Recycling-Fenster mit einem Rasterlayout einen Header hinzuzufügen, muss zunächst dem Adapter mitgeteilt werden, dass die Header-Ansicht die erste Position ist - und nicht die Standardzelle für den Inhalt. Als Nächstes muss dem Layout-Manager mitgeteilt werden, dass die erste Position eine Spannweite haben muss, die der * Spannweite der gesamten Liste entspricht. *

Nehmen Sie eine reguläre RecyclerView.Adapter-Klasse und konfigurieren Sie sie wie folgt:

```
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
    }
}
```

Dann in der Aktivität / 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);
```

Der gleiche Ansatz kann verwendet werden, um zusätzlich zu oder anstelle einer Kopfzeile eine Fußzeile hinzuzufügen.

Quelle: Chiu-Ki Chans Platzinsel-Blog

Einfache Liste mit LinearLayoutManager

In diesem Beispiel wird eine Liste von Orten mit Bild und Namen `ArrayList` indem eine `ArrayList` mit benutzerdefinierten `Place` Objekten als Datensatz verwendet wird.

Aktivitätslayout

Das Layout der Aktivität / des Fragments oder des RecyclerView muss nur das RecyclerView enthalten. Es ist kein ScrollView oder ein bestimmtes Layout erforderlich.

```
<?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>
```

Definieren Sie das Datenmodell

Sie können jeden Klassen- oder primitiven Datentyp als Modell verwenden, z. B. `int` , `String` , `float[]` oder `CustomObject` . Die RecyclerView verweist auf eine `List` dieser Objekte / Primitiven.

Wenn ein Listenelement auf verschiedene Datentypen verweist, z. B. Text, Zahlen, Bilder (wie in diesem Beispiel mit Orten), ist es oft ratsam, ein benutzerdefiniertes Objekt zu verwenden.

```
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;
    }
}
```

Listenelement-Layout

Sie müssen eine XML-Layoutdatei angeben, die für jedes Listenelement verwendet wird. In diesem Beispiel wird eine `ImageView` für das Image und eine `TextView` für den Namen verwendet. Das `LinearLayout` positioniert die `ImageView` links und die `TextView` rechts neben dem Image.

```
<?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>
```

Erstellen Sie einen RecyclerView-Adapter und einen ViewHolder

Als Nächstes müssen Sie den `RecyclerView.Adapter` und den `RecyclerView.ViewHolder` erben. Eine übliche Klassenstruktur wäre:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    public class ViewHolder extends RecyclerView.ViewHolder {
        // ...
    }
}
```

Zuerst implementieren wir den `ViewHolder` . Es erbt nur den Standardkonstruktor und speichert die erforderlichen Ansichten in einigen Feldern:

```
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);
    }
}
```

Der Konstruktor des Adapters legt das verwendete Dataset fest:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    private List<Place> mPlaces;

    public PlaceListAdapter(List<Place> contacts) {
        mPlaces = contacts;
    }

    // ...
}
```

Um unser benutzerdefiniertes Listenelement-Layout zu verwenden, überschreiben wir die Methode `onCreateViewHolder(...)` . In diesem Beispiel heißt die Layoutdatei `place_list_item.xml` .

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(
                R.layout.place_list_item,
                parent,
                false
        );
        return new ViewHolder(view);
    }

    // ...
}
```

Im `onBindViewHolder(...)` legen wir tatsächlich den Inhalt der Ansichten fest. Wir erhalten das verwendete Modell, indem wir es in der `List` an der angegebenen Position finden und dann in den Ansichten des `ViewHolder` Bild und Namen `ViewHolder` .

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public void onBindViewHolder(PlaceListAdapter.ViewHolder viewHolder, int position) {
```

```
        Place place = mPlaces.get(position);

        viewHolder.nameView.setText(place.getName());
        viewHolder.imageView.setImageBitmap(place.getImage());
    }

    // ...
}
```

Wir müssen auch `getItemCount()` implementieren, das einfach die Größe der `List getItemCount()` .

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public int getItemCount() {
        return mPlaces.size();
    }

    // ...
}
```

(Zufallsdaten erzeugen)

Für dieses Beispiel generieren wir einige zufällige Orte.

```
@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;
}
```

Verbinden Sie die RecyclerView mit dem PlaceListAdapter und der Datenmenge

Das Anschließen eines `RecyclerView` mit einem Adapter ist sehr einfach. Sie müssen den `LinearLayoutManager` als Layout-Manager einstellen, um das `LinearLayoutManager` zu erreichen.

```
@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));
}
```

Erledigt!

StaggeredGridLayoutManager

1. Erstellen Sie Ihre RecyclerView in Ihrer Layout-XML-Datei:

```
<android.support.v7.widget.RecyclerView
            android:id="@+id/recycleView"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
```

2. Erstellen Sie Ihre Modellklasse zum Speichern Ihrer Daten:

```
     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. Erstellen Sie eine Layoutdatei für RecyclerView-Elemente:

```
<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. Erstellen Sie die Adapterklasse für die RecyclerView:

```
 public class PintrestAdapter extends
RecyclerView.Adapter<PintrestAdapter.PintrestViewHolder>{
    private ArrayList<PintrestItem>images;
    Picasso picasso;
    Context context;
    public PintrestAdapter(ArrayList<PintrestItem>images,Context context){
        this.images=images;
        picasso=Picasso.with(context);
        this.context=context;

    }

    @Override
    public PintrestViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view=
LayoutInflater.from(parent.getContext()).inflate(R.layout.pintrest_layout_item,parent,false);

        return new PintrestViewHolder(view);
    }

    @Override
    public void onBindViewHolder(PintrestViewHolder holder, int position) {
          picasso.load(images.get(position).getUrl()).into(holder.imageView);
          holder.tv.setText(images.get(position).getName());
    }

    @Override
    public int getItemCount() {
        return images.size();
    }

    public class PintrestViewHolder extends RecyclerView.ViewHolder{
        ImageView imageView;
        TextView tv;
        public PintrestViewHolder(View itemView) {
            super(itemView);
            imageView=(ImageView)itemView.findViewById(R.id.imageView);
            tv=(TextView)itemView.findViewById(R.id.name);

        }
    }
}
```

5. Instanziere die RecyclerView in deiner Aktivität oder deinem Fragment:

```
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() );
```

Vergessen Sie nicht, die Picasso-Abhängigkeit in Ihre build.gradle-Datei aufzunehmen:

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

RecyclerView und LayoutManager online lesen:
https://riptutorial.com/de/android/topic/6772/recyclerview-und-layoutmanager

Kapitel 185: RecyclerView-Dekorationen

Syntax

- RecyclerView AddItemDecoration (RecyclerView.ItemDecoration-Dekoration)
- RecyclerView AddItemDecoration (RecyclerView.ItemDecoration-Dekoration, int-Index)

Parameter

Parameter	Einzelheiten
Dekoration	die Elementdekoration, die der `RecyclerView` hinzugefügt werden soll
Index	der Index in der Dekorationsliste für diese `RecyclerView` . In dieser Reihenfolge werden `getItemOffset` und `onDraw` aufgerufen. Spätere Anrufe überholen möglicherweise die vorherigen.

Bemerkungen

Dekorationen sind statisch

Da Dekorationen nur gezeichnet werden, ist es nicht möglich, Klicklistener oder andere UI-Funktionen hinzuzufügen.

Mehrere Dekorationen

Das Hinzufügen mehrerer Dekorationen zu einem `RecyclerView` wird in einigen Fällen funktionieren, aber es gibt derzeit keine öffentliche API, um andere mögliche Dekorationen beim Messen oder Zeichnen zu berücksichtigen. Sie können die Ansichtsgrenzen oder die Ansicht verzierte Grenzen erhalten, wobei die dekorierten Grenzen die Summe aller angewendeten Dekorationsversätze sind.

Andere verwandte Themen:

RecyclerView
RecyclerView onClickListeners

Offizieller Javadoc

https://developer.android.com/reference/android/support/v7/widget/RecyclerView.ItemDecoration.html

Examples

Separator zeichnen

Dies wird eine Linie am unteren Rand jeder Ansicht zeichnen, die letzte als Trennzeichen zwischen Elementen.

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

Artikelbezogene Ränder bei ItemDecoration

Sie können `RecyclerView.ItemDecoration` , um jedes Element in einer RecyclerView mit zusätzlichen Rändern zu versehen. Dies kann in einigen Fällen sowohl Ihre Adapterimplementierung als auch Ihre XML für die Elementansicht bereinigen.

```
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);
    }
}
```

Um die Dekoration zu aktivieren, fügen Sie sie einfach Ihrem RecyclerView hinzu:

```
// in your onCreate()
RecyclerView rv = (RecyclerView) findItemById(R.id.myList);
```

```
rv.addItemDecoration(new MyItemDecoration(context));
```

Trenner zu RecyclerView hinzufügen

Zunächst müssen Sie eine Klasse erstellen, die `RecyclerView.ItemDecoration` :

```
public class SimpleBlueDivider extends RecyclerView.ItemDecoration {
private Drawable mDivider;

public SimpleBlueDivider(Context context) {
    mDivider = context.getResources().getDrawable(R.drawable.divider_blue);
}

@Override
public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {
    //divider padding give some padding whatever u want or disable
    int left =parent.getPaddingLeft()+80;
    int right = parent.getWidth() - parent.getPaddingRight()-30;

    int childCount = parent.getChildCount();
    for (int i = 0; i < childCount; i++) {
        View child = parent.getChildAt(i);

        RecyclerView.LayoutParams params = (RecyclerView.LayoutParams)
child.getLayoutParams();

        int top = child.getBottom() + params.bottomMargin;
        int bottom = top + mDivider.getIntrinsicHeight();

        mDivider.setBounds(left, top, right, bottom);
        mDivider.draw(c);
    }
}

}
```

`divider_blue.xml` zu deinem Zeichenordner hinzu:

```
<?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>
```

Dann benutze es wie:

```
recyclerView.addItemDecoration(new SimpleBlueDivider(context));
```

Das Ergebnis wird wie folgt aussehen:


28% 14:20
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)

Dieses Bild ist nur ein Beispiel für die Funktionsweise von Teilern. Wenn Sie beim Hinzufügen von Teilern die Angaben zum Material Design beachten möchten, schauen Sie sich diesen Link an: Teiler und danke @Brenden Kromhout, indem Sie einen Link angeben .

So fügen Sie Teiler mit und DividerItemDecoration hinzu

Die `DividerItemDecoration` ist eine `RecyclerView.ItemDecoration` , die als Trenner zwischen Elementen verwendet werden kann.

```
DividerItemDecoration mDividerItemDecoration = new DividerItemDecoration(context,
             mLayoutManager.getOrientation());
recyclerView.addItemDecoration(mDividerItemDecoration);
```

Es unterstützt beide Orientierungen mit `DividerItemDecoration.VERTICAL` und `DividerItemDecoration.HORIZONTAL` .

ItemOffsetDecoration für GridLayoutManager in RecycleView

Das folgende Beispiel hilft, einem Element in GridLayout den gleichen Platz zu geben.

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;
    }
```

```
}
```

Sie können ItemDecoration wie folgt aufrufen.

```
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);
```

und Beispiel Artikelversatz

```
<dimen name="item_offset">5dp</dimen>
```

RecyclerView-Dekorationen online lesen: https://riptutorial.com/de/android/topic/506/recyclerview-dekorationen

Kapitel 186: Reich

Einführung

Realm Mobile Database ist eine Alternative zu SQLite. Die Realm Mobile-Datenbank ist viel schneller als ein ORM und häufig auch schneller als das rohe SQLite.

Leistungen

Offline-Funktionalität, schnelle Abfragen, sicheres Threading, plattformübergreifende Apps, Verschlüsselung, reaktive Architektur.

Bemerkungen

Wenn Sie Realm verwenden, müssen Sie daran denken, dass Sie keine Instanzen von RealmObjects, RealmResults und Realm zwischen Threads übergeben dürfen. Wenn Sie eine Abfrage für einen bestimmten Thread benötigen, öffnen Sie eine Realm-Instanz in diesem Thread. Bei Beendigung des Threads sollten Sie das Realm schließen.

> ***RECHTLICHER HINWEIS*** : Sie verstehen, dass die Software möglicherweise kryptografische Funktionen enthält, die Exportbeschränkungen unterliegen können, und Sie ***versichern*** und garantieren, dass **Sie sich nicht in einem Land befinden** , das Exportbeschränkungen oder Embargo der USA unterliegt, **einschließlich Kuba, Iran, Nord Korea, Sudan, Syrien oder die Krimregion** , und Sie befinden sich nicht auf der Liste der abgelehnten Personen, nicht verifizierten Parteien des Handelsministeriums oder einer eingeschränkten Einheit.

Examples

Realm zu Ihrem Projekt hinzufügen

Fügen Sie der **Projektstufe** `build.gradle` die folgende Abhängigkeit `build.gradle` .

```
dependencies {
    classpath "io.realm:realm-gradle-plugin:3.1.2"
}
```

Fügen Sie oben in Ihrer `build.gradle` Datei auf **App-** Ebene `build.gradle` :

```
apply plugin: 'realm-android'
```

Schließen Sie eine Gradle-Synchronisierung ab, und Sie haben jetzt Realm als Abhängigkeit zu Ihrem Projekt hinzugefügt!

Realm erfordert vor der Verwendung einen ersten Aufruf seit 2.0.0. Sie können dies in Ihrer

`Application` Klasse oder in der `onCreate` Methode Ihrer ersten Activity `onCreate` .

```
Realm.init(this); // added in Realm 2.0.0
Realm.setDefaultConfiguration(new RealmConfiguration.Builder().build());
```

Realm-Modelle

Realm-Modelle müssen die `RealmObject` Basisklasse erweitern, sie definieren das Schema der zugrunde liegenden Datenbank.

Unterstützte Feldtypen sind `boolean` , `byte` , `short` , `int` , `long` , `float` , `double` , `String` , `Date` , `byte[]` , Verknüpfungen zu anderen `RealmObject` und `RealmList<T extends RealmModel>` .

```
public class Person extends RealmObject {
    @PrimaryKey //primary key is also implicitly an @Index
                //it is required for `copyToRealmOrUpdate()` to update the object.
    private long id;

    @Index //index makes queries faster on this field
    @Required //prevents `null` value from being inserted
    private String name;

    private RealmList<Dog> dogs; //->many relationship to Dog

    private Person spouse; //->one relationship to Person

    @Ignore
    private Calendar birthday; //calendars are not supported but can be ignored

    // getters, setters
}
```

Wenn Sie Ihrem RealmObject ein neues Feld hinzufügen (oder entfernen) (oder Sie eine neue RealmObject-Klasse hinzufügen oder ein vorhandenes löschen), ist eine **Migration** erforderlich. Sie können `deleteIfMigrationNeeded()` in Ihrem `RealmConfiguration.Builder` festlegen oder die erforderliche Migration definieren. Eine Migration ist auch erforderlich, wenn Sie die `@Required` `@Index` oder `@PrimaryKey` `@Required` oder `@Index` `@PrimaryKey` Annotation hinzufügen (oder entfernen).

Beziehungen müssen manuell festgelegt werden. Sie basieren NICHT automatisch auf Primärschlüsseln.

Seit 0.88.0 ist es auch möglich, öffentliche Felder anstelle von privaten Feldern / Getters / Setters in RealmObject-Klassen zu verwenden.

Es ist auch möglich zu implementieren `RealmModel` statt erstreckt `RealmObject` , wenn die Klasse auch mit Anmerkungen versehen ist `@RealmClass` .

```
@RealmClass
public class Person implements RealmModel {
    // ...
}
```

In diesem Fall werden Methoden wie `person.deleteFromRealm()` oder `person.addChangeListener()` durch `RealmObject.deleteFromRealm(person)` und `RealmObject.addChangeListener(person)` .

Einschränkungen bestehen darin, dass durch ein `RealmObject` nur `RealmObject` erweitert werden kann und `final` , `volatile` und `transient` Felder nicht unterstützt werden.

Es ist wichtig, dass eine *verwaltete* RealmObject-Klasse nur in einer Transaktion geändert werden kann. Ein *verwaltetes* RealmObject kann nicht zwischen Threads übergeben werden.

Liste der Grundelemente (RealmList)

Realm unterstützt derzeit nicht das Speichern einer Liste von Grundelementen. Es ist auf ihrer ToDo-Liste (GitHub-Ausgabe Nr. 575), aber in der Zwischenzeit ist hier eine Problemumgehung.

Erstellen Sie eine neue Klasse für Ihren primitiven Typ, dies verwendet Integer, aber ändern Sie sie für das, was Sie speichern möchten.

```
public class RealmInteger extends RealmObject {
    private int val;

    public RealmInteger() {
    }

    public RealmInteger(int val) {
        this.val = val;
    }

    // Getters and setters
}
```

Sie können dies jetzt in Ihrem `RealmObject` .

```
public class MainObject extends RealmObject {

    private String name;
    private RealmList<RealmInteger> ints;

    // Getters and setters
}
```

Wenn Sie `GSON` , um Ihr `RealmObject` , müssen Sie einen benutzerdefinierten `RealmObject` hinzufügen.

```
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();
```

Try-With-Ressourcen

```
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
}
```

Die Try with-Ressourcen können nur von KITKAT verwendet werden (minSDK 19)

Sortierte Abfragen

Um eine Abfrage zu sortieren, sollten Sie anstelle von `findAll()` `findAllSorted()` .

```
RealmResults<SomeObject> results = realm.where(SomeObject.class)
                                        .findAllSorted("sortField", Sort.ASCENDING);
```

Hinweis:

`sort()` gibt ein völlig neues RealmResults zurück, das sortiert ist, aber ein Update dieses RealmResults setzt es zurück. Wenn Sie `sort()` , sollten Sie es immer in Ihrem `RealmChangeListener` neu sortieren, den `RealmChangeListener` aus den vorherigen `RealmResults` und den zurückgegebenen neuen `RealmResults` . Die Verwendung von `sort()` für ein `RealmResults` von einer noch nicht geladenen `RealmResults` Abfrage zurückgegeben wird, `RealmResults` fehl.

`findAllSorted()` gibt die Ergebnisse immer nach Feld sortiert zurück, auch wenn es aktualisiert wird. Es wird empfohlen, `findAllSorted()` .

Async-Abfragen

Jede synchrone Abfragemethode (wie `findAll()` oder `findAllSorted()`) hat ein asynchrones Gegenstück (`findAllAsync()` / `findAllSortedAsync()`).

Asynchrone Abfragen lagern die Auswertung der `RealmResults` in einen anderen Thread aus. Um diese Ergebnisse im aktuellen Thread zu erhalten, muss der aktuelle Thread ein Looper-Thread sein (read: async-Abfragen funktionieren normalerweise nur im UI-Thread).

```
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);
```

Verwenden von Realm mit RxJava

Für Abfragen stellt Realm die Methode `realmResults.asObservable()` zur Verfügung. Das Beobachten von Ergebnissen ist nur für Looper-Threads (normalerweise der UI-Thread) möglich.

Damit dies funktioniert, muss Ihre Konfiguration Folgendes enthalten

```
realmConfiguration = new RealmConfiguration.Builder(context)       //
                         .rxFactory(new RealmObservableFactory()) //
                            //...
                         .build();
```

Anschließend können Sie Ihre Ergebnisse als beobachtbare verwenden.

```
Observable<RealmResults<SomeObject>> observable = results.asObservable();
```

Bei asynchronen Abfragen sollten Sie die Ergebnisse nach `isLoaded()` filtern, damit Sie nur dann ein Ereignis erhalten, wenn die Abfrage ausgeführt wurde. Dieses `filter()` wird nicht für synchrone Abfragen benötigt (`isLoaded()` gibt bei Synchronisationsabfragen immer `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));
```

Für Schreibvorgänge sollten Sie entweder die Methode `executeTransactionAsync()` verwenden oder

eine Realm-Instanz im Hintergrundthread öffnen, die Transaktion synchron ausführen und dann die Realm-Instanz schließen.

```
public Subscription loadObjectsFromNetwork() {
    return objectApi.getObjects()
        .subscribeOn(Schedulers.io())
        .subscribe(response -> {
            try(Realm realmInstance = Realm.getDefaultInstance()) {
                realmInstance.executeTransaction(realm ->
realm.insertOrUpdate(response.objects));
            }
        });
}
```

Grundlegende Verwendung

Instanz einrichten

Um Realm zu verwenden, müssen Sie zuerst eine Instanz davon erhalten. Jede Realm-Instanz ist einer Datei auf der Festplatte zugeordnet. Der einfachste Weg, um eine Instanz zu erhalten, lautet wie folgt:

```
// 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();
```

Die Methode `Realm.getInstance()` erstellt die Datenbankdatei, falls sie noch nicht erstellt wurde. Andernfalls wird die Datei geöffnet. Das `RealmConfiguration` Objekt steuert alle Aspekte der `RealmConfiguration` eines Realms - ob in einer `inMemory()` -Datenbank, Name der Realm-Datei, ob der Realm gelöscht werden soll, wenn eine Migration erforderlich ist, Anfangsdaten usw.

Bitte beachten Sie, dass Aufrufe von `Realm.getInstance()` Referenzzähler sind (jeder Aufruf erhöht einen Zähler) und der Zähler wird beim `realm.close()` dekrementiert.

Eine Instanz schließen

Auf Hintergrund - Threads, ist es *sehr wichtig* , die Realm - Instanz (en) zu ***schließen*** , sobald es nicht mehr verwendet wird (zum Beispiel Transaktion abgeschlossen ist und die Thread - Ausführung endet). Wenn Sie nicht alle Realm-Instanzen im Hintergrund-Thread schließen, führt dies zu einem Versionspinning und kann zu einer erheblichen Vergrößerung der Dateigröße führen.

```
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
```

Es ist erwähnenswert, dass Sie oberhalb von API Level 19 diesen Code durch Folgendes ersetzen können:

```
try(Realm realm = Realm.getDefaultInstance()) {
    // ...
}
```

Modelle

Der nächste Schritt wäre das Erstellen Ihrer Modelle. Hier könnte eine Frage gestellt werden: "Was ist ein Modell?". Ein Modell ist eine Struktur, die Eigenschaften eines Objekts definiert, das in der Datenbank gespeichert wird. Im Folgenden modellieren wir beispielsweise ein Buch.

```
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;
    }
}
```

Beachten Sie, dass Ihre Modelle die RealmObject-Klasse erweitern sollten. Der Primärschlüssel wird auch durch `@PrimaryKey` Annotation `@PrimaryKey` angegeben. Primärschlüssel können Null sein, aber nur ein Element kann `null` als Primärschlüssel haben. Sie können auch die Annotation `@Ignore` für die Felder verwenden, die nicht auf dem Datenträger gespeichert werden sollen:

```
@Ignore
private String isbn;
```

Daten einfügen oder aktualisieren

Um ein Buchobjekt in Ihrer Realm-Datenbankinstanz zu speichern, können Sie zuerst eine Instanz Ihres Modells erstellen und diese dann mit der `copyToRealm` Methode in der Datenbank `copyToRealm` . Zum Erstellen oder Aktualisieren können Sie `copyToRealmOrUpdate` . (Eine schnellere Alternative ist das neu hinzugefügte `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);
    }
});
```

Beachten Sie, dass alle Änderungen an Daten in einer Transaktion erfolgen müssen. Eine andere Methode zum Erstellen eines Objekts ist das folgende Muster:

```
Book book = realm.createObject(Book.class, primaryKey);
...
```

Datenbank abfragen

- Alle Bücher:

  ```
  RealmResults<Book> results = realm.where(Book.class).findAll();
  ```

- Alle Bücher mit einer ID größer als 10:

```
RealmResults<Book> results = realm.where(Book.class)
                                  .greaterThan("id", 10)
                                  .findAll();
```

- Bücher von `'Taylor Swift'` oder `'%Peter%'` :

```
RealmResults<Book> results = realm.where(Book.class)
                                  .beginGroup()
                                      .equalTo("author", "Taylor Swift")
                                      .or()
                                      .contains("author", "Peter")
                                  .endGroup().findAll();
```

Objekt löschen

Zum Beispiel möchten wir alle Bücher von Taylor Swift löschen:

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

Reich online lesen: https://riptutorial.com/de/android/topic/3187/reich

Kapitel 187: RenderScript

Einführung

RenderScript ist eine Skriptsprache, mit der Sie leistungsstarkes Grafik-Rendering und reinen Rechencode schreiben können. Es bietet die Möglichkeit, leistungskritischen Code zu schreiben, den das System später in nativen Code für den Prozessor kompiliert, auf dem es ausgeführt werden kann. Dies kann die CPU, eine Multi-Core-CPU oder sogar die GPU sein. Worauf es letztlich läuft, hängt von vielen Faktoren ab, die dem Entwickler nicht ohne weiteres zur Verfügung stehen, sondern auch von der Architektur, die der interne Plattformcompiler unterstützt.

Examples

Fertig machen

RenderScript ist ein Framework für die parallele Hochleistungsberechnung unter Android. Skripts, die Sie schreiben, werden auf allen verfügbaren Prozessoren (z. B. CPU, GPU usw.) parallel ausgeführt, sodass Sie sich auf die Aufgabe konzentrieren können, die Sie ausführen möchten, anstatt wie geplant und ausgeführt wird.

Skripts werden in einer C99-Sprache geschrieben (C99 ist eine alte Version des C-Programmiersprachenstandards). Für jedes Skript wird eine Java-Klasse erstellt, mit der Sie problemlos mit RenderScript in Ihrem Java-Code interagieren können.

Einrichten Ihres Projekts

Es gibt zwei verschiedene Möglichkeiten, auf RenderScript in Ihrer App zuzugreifen, mit den Android Framework-Bibliotheken oder der Support Library. Auch wenn Sie keine Geräte vor API Level 11 ansprechen möchten, sollten Sie immer die Support Library-Implementierung verwenden, da die Gerätekompatibilität für viele verschiedene Geräte gewährleistet ist. Um die Support Library-Implementierung zu verwenden, müssen Sie mindestens Build-Tools Version `18.1.0` !

Jetzt können Sie die build.gradle -Datei Ihrer Anwendung einrichten:

```
android {
    compileSdkVersion 24
    buildToolsVersion '24.0.1'

    defaultConfig {
        minSdkVersion 8
        targetSdkVersion 24

        renderscriptTargetApi 18
        renderscriptSupportModeEnabled true
    }
}
```

- `renderscriptTargetApi` : Dies sollte auf die früheste API-Version festgelegt werden, die alle erforderlichen RenderScript-Funktionen bietet.
- `renderscriptSupportModeEnabled` : Dies ermöglicht die Verwendung der RenderScript-Implementierung der Support Library.

Wie RenderScript funktioniert

Ein typisches RenderScript besteht aus zwei Dingen: Kerneln und Funktionen. Eine Funktion ist genau das, was sie sich anhört - sie akzeptiert eine Eingabe, macht etwas mit dieser Eingabe und gibt eine Ausgabe zurück. Ein Kernel ist, woher die wirkliche Kraft von RenderScript kommt.

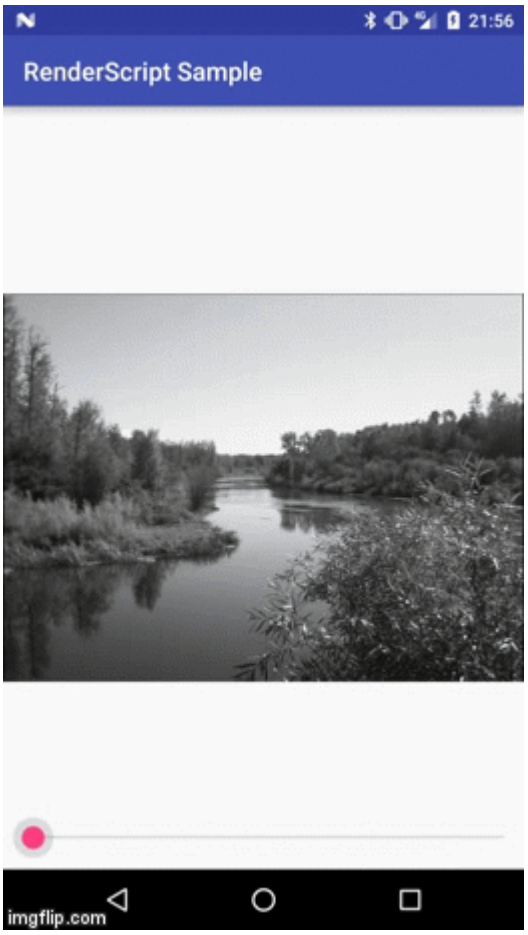
Ein Kernel ist eine Funktion, die für jedes Element in einer `Allocation` . Eine `Allocation` kann verwendet werden, um Daten wie eine `Bitmap` oder ein `byte` Array an ein `RenderScript` und sie werden auch verwendet, um ein Ergebnis aus einem Kernel zu erhalten. Kernel können entweder eine `Allocation` als Eingabe und eine andere als Ausgabe verwenden oder sie können die Daten in nur einer `Allocation` ändern.

Sie können Ihre eigenen Kernels schreiben, aber es gibt auch viele vordefinierte Kernel, mit denen Sie gängige Operationen wie Gaußsche Bildunschärfe ausführen können.

Wie bereits erwähnt, wird für jede RenderScript-Datei eine Klasse generiert, um damit zu interagieren. Diese Klassen beginnen immer mit dem Präfix `ScriptC_` gefolgt vom Namen der RenderScript-Datei. Wenn Ihre RenderScript-Datei beispielsweise als `example` wird, heißt die generierte Java-Klasse `ScriptC_example` . Alle vordefinierten Skripts beginnen einfach mit dem Präfix `Script` - beispielsweise heißt das Gaußsche Bild-Unschärfeskript `ScriptIntrinsicBlur` .

Schreiben Sie Ihr erstes RenderScript

Das folgende Beispiel basiert auf einem Beispiel für GitHub. Es führt grundlegende Bildmanipulationen durch, indem die Sättigung eines Bildes geändert wird. Sie können den Quellcode **hier finden** und herausfinden, ob Sie selbst damit herumspielen wollen. Hier ist ein kurzer Hinweis darauf, wie das Ergebnis aussehen soll:



RenderScript-Zwischenablage

RenderScript-Dateien befinden sich im Ordner `src/main/rs` in Ihrem Projekt. Jede Datei hat die Dateierweiterung `.rs` und muss oben zwei `#pragma` Anweisungen enthalten:

```
#pragma version(1)
#pragma rs java_package_name(your.package.name)
```

- `#pragma version(1)` : Hier können Sie die Version von RenderScript festlegen, die Sie verwenden. Derzeit gibt es nur Version 1.

- `#pragma rs java_package_name(your.package.name)` : Hiermit kann der Paketname der generierten Java-Klasse festgelegt werden, um mit diesem bestimmten RenderScript zu interagieren.

Es gibt ein weiteres `#pragma` Sie normalerweise in jeder Ihrer RenderScript-Dateien `#pragma` sollten. Mit diesem Parameter wird die Gleitkomma-Genauigkeit festgelegt. Sie können die Gleitkommazahl auf drei verschiedene Ebenen einstellen:

- `#pragma rs_fp_full` : Dies ist die strengste Einstellung mit der höchsten Genauigkeit, und es ist auch der Standardwert, wenn nichts angegeben wird. Sie sollten dies verwenden, wenn Sie eine hohe Gleitkomma-Genauigkeit benötigen.
- `#pragma rs_fp_relaxed` : Dies gewährleistet zwar eine nicht ganz so hohe Gleitkomma-Genauigkeit, aber bei manchen Architekturen ermöglicht es eine Reihe von Optimierungen,

die dazu führen können, dass Ihre Skripts schneller ausgeführt werden.
- `#pragma rs_fp_imprecise` : Dies sorgt für eine noch geringere Genauigkeit und sollte verwendet werden, wenn die Gleitkomma-Genauigkeit für Ihr Skript keine Rolle spielt.

Die meisten Skripts können nur `#pragma rs_fp_relaxed` sei denn, Sie benötigen eine hohe Gleitkomma-Genauigkeit.

Globale Variablen

Wie in C-Code können Sie nun globale Variablen oder Konstanten definieren:

```
const static float3 gMonoMult = {0.299f, 0.587f, 0.114f};

float saturationLevel = 0.0f;
```

Die Variable `gMonoMult` ist vom Typ `float3` . Dies bedeutet, dass es sich um einen Vektor handelt, der aus 3 Float-Nummern besteht. Die andere `float` Variable mit dem Namen `saturationValue` ist nicht konstant. Daher können Sie sie zur Laufzeit auf einen gewünschten Wert setzen. Sie können Variablen wie diese in Ihren Kerneln oder Funktionen verwenden. Sie sind daher eine andere Möglichkeit, Eingaben in Ihre RenderScripts zu geben oder von ihnen zu empfangen. Für jede nicht konstante Variable wird eine Getter- und Setter-Methode für die zugehörige Java-Klasse generiert.

Kernel

Aber jetzt lass uns den Kernel implementieren. Für die Zwecke dieses Beispiels werde ich nicht die im Kernel verwendete Mathematik erläutern, um die Sättigung des Bildes zu ändern, sondern mich darauf konzentrieren, wie ein Kernel implementiert wird und wie er verwendet wird. Am Ende dieses Kapitels werde ich schnell erklären, was der Code in diesem Kernel tatsächlich tut.

Kernel im Allgemeinen

Schauen wir uns zuerst den Quellcode an:

```
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);
}
```

Wie Sie sehen, sieht es aus wie eine normale C-Funktion mit einer Ausnahme: Das `__attribute__((kernel))` zwischen dem Rückgabetyp und dem Methodennamen. Dies ist, was RenderScript sagt, dass diese Methode ein Kernel ist. Möglicherweise werden Sie auch feststellen, dass diese Methode einen `uchar4` Parameter akzeptiert und einen anderen `uchar4` Wert `uchar4` . `uchar4` ist - wie die Variable `float3` , die wir im `float3` Kapitel besprochen haben - ein Vektor. Es enthält 4 `uchar` Werte, bei denen es sich lediglich um Byte-Werte im Bereich von 0 bis 255 handelt.

Sie können auf diese einzelnen Werte auf viele verschiedene Arten zugreifen, z. B. würde `in.r` das Byte zurückgeben, das dem roten Kanal eines Pixels entspricht. Wir verwenden ein `uchar4` da jedes Pixel aus 4 Werten besteht - `r` für Rot, `g` für Grün, `b` für Blau und `a` für Alpha - und Sie können mit dieser Abkürzung darauf zugreifen. Mit RenderScript können Sie auch eine beliebige Anzahl von Werten aus einem Vektor übernehmen und daraus einen anderen Vektor erstellen. Beispielsweise würde `in.rgb` einen `uchar3` Wert zurückgeben, der nur den roten, grünen und blauen Teil des Pixels ohne den Alpha-Wert enthält.

Zur Laufzeit ruft RenderScript diese Kernel-Methode für jedes Pixel eines Bildes auf, weshalb der Rückgabewert und der Parameter nur ein `uchar4` Wert sind. RenderScript führt viele dieser Aufrufe parallel auf allen verfügbaren Prozessoren aus, weshalb RenderScript so leistungsfähig ist. Dies bedeutet auch, dass Sie sich nicht um Threading oder Thread-Sicherheit sorgen müssen. Sie können einfach alles, was Sie tun möchten, für jedes Pixel implementieren, und RenderScript kümmert sich um den Rest.

Wenn Sie einen Kernel in Java aufrufen, geben Sie zwei `Allocation` Variablen an, eine, die die Eingabedaten enthält, und eine, die die Ausgabe erhält. Ihre Kernel-Methode wird für jeden Wert in der Input `Allocation` aufgerufen und schreibt das Ergebnis in die Output `Allocation` .

RenderScript-Laufzeit-API-Methoden

Im obigen Kernel werden einige Methoden verwendet, die im Lieferumfang enthalten sind. RenderScript bietet viele solcher Methoden und sie sind für fast alles, was Sie mit RenderScript machen werden, von entscheidender Bedeutung. Darunter befinden sich Methoden für mathematische Operationen wie `sin()` und Hilfsmethoden wie `mix()` die zwei Werte nach anderen Werten mischen. Es gibt aber auch Methoden für komplexere Operationen im Umgang mit Vektoren, Quaternionen und Matrizen.

Die offizielle **RenderScript-Laufzeit-API-Referenz** ist die beste Ressource, wenn Sie mehr über eine bestimmte Methode erfahren möchten oder nach einer bestimmten Methode suchen, die eine allgemeine Operation wie das Berechnen des Punktprodukts einer Matrix durchführt. Diese Dokumentation finden Sie **hier** .

Kernel-Implementierung

Schauen wir uns nun die Besonderheiten des Kernels an. Hier ist die erste Zeile im Kernel:

```
float4 f4 = rsUnpackColor8888(in);
```

Die erste Zeile ruft die eingebaute Methode **`rsUnpackColor8888()`** die den `uchar4` Wert in einen `float4` Wert `float4` . Jeder Farbkanal wird auch in den Bereich `0.0f - 1.0f` wobei `0.0f` einem Byte-Wert von `0` und `1.0f` bis `255` . Der Hauptzweck besteht darin, die gesamte Mathematik in diesem Kernel viel einfacher zu machen.

```
float3 dotVector = dot(f4.rgb, gMonoMult);
```

Diese nächste Zeile verwendet die integrierte Methode **`dot()`** , um das Punktprodukt zweier

Vektoren zu berechnen. `gMonoMult` ist ein konstanter Wert, den wir oben ein paar Kapitel definiert haben. Da beide Vektoren dieselbe Länge haben müssen, um das Punktprodukt zu berechnen, und da wir nur die Farbkanäle und nicht den Alphakanal eines Pixels beeinflussen möchten, verwenden wir die `.rgb` , um einen neuen `float3` Vektor zu erhalten, der nur den rote, grüne und blaue Farbkanäle. Diejenigen von uns, die sich noch an die Schule erinnern, wie das Dot-Produkt funktioniert, werden schnell feststellen, dass das Dot-Produkt nur einen Wert und nicht einen Vektor liefern soll. Im obigen Code weisen wir das Ergebnis jedoch einem `float3` Vektor zu. Dies ist wieder eine Funktion von RenderScript. Wenn Sie einem Vektor eine eindimensionale Zahl zuweisen, werden alle Elemente im Vektor auf diesen Wert gesetzt. Das folgende Snippet weist `2.0f` jedem der drei Werte im `float3` Vektor `float3` :

```
float3 example = 2.0f;
```

Das Ergebnis des obigen Punktprodukts wird also jedem Element im darüber `float3` Vektor `float3` zugewiesen.

Jetzt kommt der Teil, in dem wir tatsächlich die globale Variable `saturationLevel` , um die Sättigung des Bildes zu ändern:

```
float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
```

Hierbei wird die eingebaute Methode **`mix()`** um die Originalfarbe mit dem oben erstellten Produktvektor zu mischen. Wie sie miteinander vermischt werden, hängt von der globalen Variable `saturationLevel` . Ein `saturationLevel` von `0.0f` , dass die resultierende Farbe keinen Teil der ursprünglichen Farbwerte hat und nur aus Werten im `dotVector` die zu einem Schwarzweiß- oder ausgegrauten Bild führen. Ein Wert von `1.0f` , dass die resultierende Farbe vollständig aus den ursprünglichen Farbwerten besteht. Werte über `1.0f` multiplizieren die Originalfarben, um sie heller und intensiver zu machen.

```
return rsPackColorTo8888(newColor);
```

Dies ist der letzte Teil im Kernel. **`rsPackColorTo8888()`** wandelt den `float3` Vektor zurück in einen `uchar4` Wert, der zurückgegeben wird. Die resultierenden Bytewerte werden auf einen Bereich zwischen 0 und 255 festgelegt, sodass Float-Werte über `1.0f` zu einem Byte-Wert von 255 führen und Werte unter `0.0` für einen Byte-Wert von `0` .

Und das ist die gesamte Kernel-Implementierung. Jetzt ist nur noch ein Teil übrig: Wie ruft man einen Kernel in Java auf?

RenderScript in Java aufrufen

Grundlagen

Wie bereits oben erwähnt, wird für jede RenderScript-Datei eine Java-Klasse generiert, mit der Sie mit Ihren Skripts interagieren können. Diese Dateien haben das Präfix `ScriptC_` gefolgt vom

Namen der RenderScript-Datei. Um eine Instanz dieser Klassen zu erstellen, benötigen Sie zunächst eine Instanz der `RenderScript` Klasse:

```
final RenderScript renderScript = RenderScript.create(context);
```

Die statische Methode `create()` kann verwendet werden, um eine `RenderScript` Instanz aus einem `Context` zu erstellen. Sie können dann die Java-Klasse instanziieren, die für Ihr Skript generiert wurde. Wenn Sie die RenderScript-Datei `saturation.rs` aufgerufen `ScriptC_saturation` wird die Klasse `ScriptC_saturation` :

```
final ScriptC_saturation script = new ScriptC_saturation(renderScript);
```

In dieser Klasse können Sie nun den Sättigungsgrad einstellen und den Kernel aufrufen. Der Setter, der für die Variable `saturationLevel` generiert wurde, hat das Präfix `set_` gefolgt vom Namen der Variablen:

```
script.set_saturationLevel(1.0f);
```

Es gibt auch einen Getter mit dem Präfix `get_` , mit dem Sie den aktuell eingestellten Sättigungspegel `get_` können:

```
float saturationLevel = script.get_saturationLevel();
```

Kerneln, die Sie in Ihrem RenderScript definieren, sind `forEach_` gefolgt vom Namen der Kernel-Methode vorangestellt. Der Kernel, den wir geschrieben haben, erwartet als Parameter eine Input `Allocation` und eine Output `Allocation` :

```
script.forEach_saturation(inputAllocation, outputAllocation);
```

Die `Allocation` muss das Eingangsbild enthalten, und nach dem `forEach_saturation` Verfahren die Ausgangszuweisung enthält die geänderten Bilddaten beendet ist.

Sobald Sie eine `Allocation` Instanz haben, können Sie Daten von und zu diesen `Allocations` kopieren, indem Sie die Methoden `copyFrom()` und `copyTo()` . Zum Beispiel können Sie ein neues Bild in Ihre Eingabe `Allocation kopieren, indem Sie Folgendes aufrufen:

```
inputAllocation.copyFrom(inputBitmap);
```

Auf dieselbe Weise können Sie das Ergebnisbild `copyTo()` indem Sie `copyTo()` in der Ausgabe `Allocation` aufrufen:

```
outputAllocation.copyTo(outputBitmap);
```

Allocation-Instanzen erstellen

Es gibt viele Möglichkeiten, eine `Allocation` zu erstellen. Sobald Sie eine `Allocation` Instanz haben,

können Sie neue Daten von und zu diesen `Allocations` mit `copyTo()` und `copyFrom()` wie oben beschrieben, aber um sie zu erstellen, müssen Sie zunächst wissen, mit welchen Daten Sie genau arbeiten. Beginnen wir mit der Eingabe `Allocation` :

Wir können die statische Methode `createFromBitmap()` , um schnell unsere Eingabe erstellen `Allocation` aus einer `Bitmap` :

```
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, image);
```

In diesem Beispiel nie das Eingangsbild ändert brauchen wir so nie die Eingabe ändern `Allocation` wieder. Wir können es jedes Mal wieder verwenden die `saturationLevel` ändert eine neue Ausgabe erstellen `Bitmap` .

Erstellen der Ausgabe Die `Allocation` ist etwas komplexer. Zuerst müssen wir einen sogenannten `Type` erstellen. Ein `Type` wird verwendet, um einer `Allocation` mitzuteilen, mit welchen Daten es sich handelt. Normalerweise verwendet man die `Type.Builder` Klasse, um schnell einen geeigneten `Type` zu erstellen. Schauen wir uns zuerst den Code an:

```
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
        .setX(inputBitmap.getWidth())
        .setY(inputBitmap.getHeight())
        .create();
```

Wir arbeiten mit einer normalen 32-Bit- `Bitmap` (oder mit anderen Worten 4 Byte) mit 4 Farbkanälen. Deshalb wählen wir `Element.RGBA_8888` , um den `Type` zu erstellen. Dann verwenden wir die Methoden `setX()` und `setY()` , um die Breite und Höhe des Ausgabebildes auf dieselbe Größe wie das Eingabebild festzulegen. Die Methode `create()` erstellt dann den `Type` mit den von uns angegebenen Parametern.

Sobald wir den richtigen `Type` haben, können wir die Ausgabe- `Allocation` mit der statischen Methode `createTyped()` :

```
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);
```

Jetzt sind wir fast fertig. Wir brauchen auch einen Ausgang `Bitmap` , in dem wir die Daten aus der Ausgabe kopieren `Allocation` . Dazu müssen wir die statische Methode `createBitmap()` einen neuen leeren erstellen `Bitmap` mit der gleichen Größe und Konfiguration wie der Eingang `Bitmap` .

```
final Bitmap outputBitmap = Bitmap.createBitmap(
        inputBitmap.getWidth(),
        inputBitmap.getHeight(),
        inputBitmap.getConfig()
);
```

Und damit haben wir alle Puzzleteile, um unser RenderScript auszuführen.

Vollständiges Beispiel

Lassen Sie uns dies in einem Beispiel zusammenfassen:

```
// 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);
```

Fazit

Mit dieser Einführung sollten Sie alle eigenen RenderScript-Kernel für die einfache Bildbearbeitung schreiben. Es gibt jedoch ein paar Dinge, die Sie beachten sollten:

- **RenderScript funktioniert nur in Anwendungsprojekten** : Derzeit können RenderScript-Dateien nicht Teil eines Bibliotheksprojekts sein.
- **Achten Sie auf Speicher** : RenderScript ist sehr schnell, kann jedoch auch speicherintensiv sein. `RenderScript` keinem Zeitpunkt sollte es mehr als eine Instanz von `RenderScript` . Sie sollten auch so viel wie möglich wiederverwenden. Normalerweise müssen Sie Ihre `Allocation` Instanzen nur einmal erstellen und können sie in Zukunft wiederverwenden. Das gleiche gilt für die Ausgabe `Bitmaps` oder Script - Instanzen. Verwenden Sie so viel wie möglich.
- **Machen Sie Ihre Arbeit im Hintergrund** : Wieder ist RenderScript sehr schnell, aber in keiner Weise sofort. Jeder Kernel, besonders komplexe, sollten in einer `AsyncTask` oder etwas

Ähnlichem vom UI-Thread ausgeführt werden. In den meisten Fällen müssen Sie sich jedoch nicht um Speicherlecks kümmern. Alle renderbezogenen Klassen nur die Anwendung verwenden `Context` und damit verursachen keine Speicherlecks. Sie müssen sich jedoch immer noch um die üblichen Dinge wie auslaufendes `View` , `Activity` oder eine `Context` kümmern, die Sie selbst verwenden.

- **Verwenden Sie integrierte Funktionen** : Es gibt viele vordefinierte Skripts, die Aufgaben wie Verwischen, Verwischen, Konvertieren und Ändern der Größe von Bildern ausführen. Es gibt viele weitere integrierte Methoden, die Sie bei der Implementierung Ihrer Kernel unterstützen. Wenn Sie etwas tun wollen, gibt es wahrscheinlich ein Skript oder eine Methode, die bereits das tut, was Sie versuchen. Das Rad nicht neu erfinden.

Wenn Sie schnell loslegen und mit aktuellem Code herumspielen möchten, sollten Sie sich das GitHub-Beispielprojekt ansehen, in dem das in diesem Tutorial beschriebene Beispiel genau implementiert ist. Das Projekt finden Sie **hier** . Viel Spaß mit RenderScript!

Verwischen Sie ein Bild

In diesem Beispiel wird veranschaulicht, wie Sie mit der Renderscript-API ein Bild verwischen (mit Bitmap). In diesem Beispiel wird ScriptInstrinsicBlur verwendet, das von der Android Renderscript-API (API> = 17) bereitgestellt wird.

```
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;
    }
}
```

Jedes Skript verfügt über einen Kern, der die Daten verarbeitet, und es wird im Allgemeinen über die Methode `forEach` aufgerufen.

```
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
    }
}
```

Damit ist das Beispiel hier abgeschlossen. Es wird empfohlen, die Verarbeitung in einem Hintergrundthread durchzuführen.

Verwischen Sie eine Ansicht

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;
    }
}
```

Verwendungszweck:

```
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);
```

RenderScript online lesen: https://riptutorial.com/de/android/topic/5214/renderscript

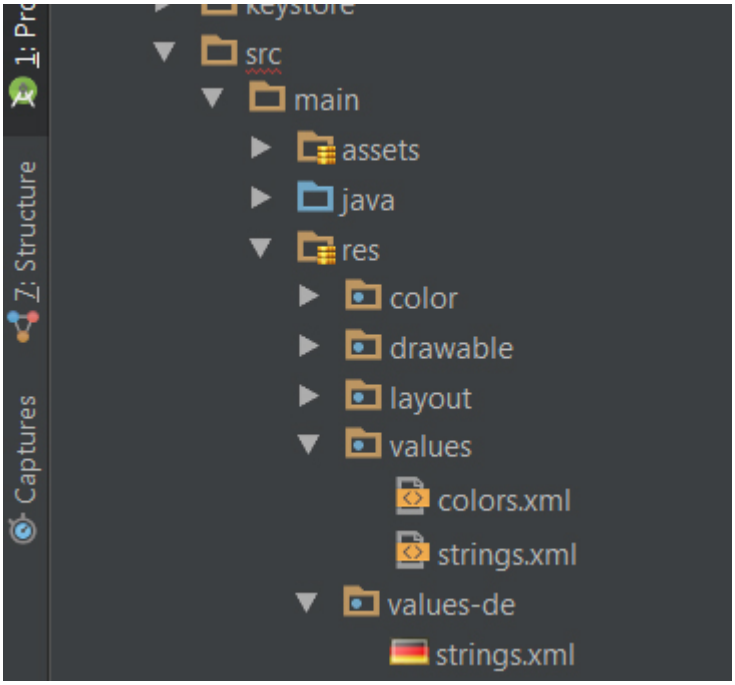
Kapitel 188: Ressourcen

Examples

Übersetzen Sie eine Zeichenfolge

Zeichenfolgen können internationalisiert werden, indem für jede unterstützte Sprache eine andere Zeichenfolge.xml definiert wird.

Sie fügen eine neue Sprache hinzu, indem Sie ein neues Werteverzeichnis mit dem ISO-Sprachcode als Suffix erstellen. Wenn Sie beispielsweise ein deutsches Set hinzufügen, könnte Ihre Struktur wie folgt aussehen:



Wenn das System nach der angeforderten Zeichenfolge sucht, prüft es zuerst die sprachspezifische XML-Datei. Wird sie nicht gefunden, wird der Wert aus der Standarddatei strings.xml zurückgegeben. Der Schlüssel bleibt für jede Sprache gleich und nur der Wert ändert sich.

Beispielinhalt:

/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>
```

Zeichenketten definieren

Zeichenfolgen werden normalerweise in der Ressourcendatei `strings.xml` gespeichert. Sie werden mit einem `<string>` XML-Element definiert.

Der Zweck von strings.xml ist es, Internationalisierung zu ermöglichen. Sie können für jeden Sprachcode eine Zeichenfolge.xml definieren. Wenn das System nach dem String 'app_name' sucht, prüft es zuerst die der aktuellen Sprache entsprechende XML-Datei. Wenn es nicht gefunden wird, sucht es nach dem Eintrag in der Standarddatei strings.xml. Dies bedeutet, dass Sie nur einige Ihrer Strings lokalisieren können, andere jedoch nicht.

/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>
```

Sobald eine Zeichenfolge in einer XML-Ressourcendatei definiert ist, kann sie von anderen Teilen der App verwendet werden.

Die XML-Projektdateien einer App können ein `<string>` -Element verwenden, indem sie auf `@string/string_name` . Beispielsweise enthält die Manifestdatei (/manifests/AndroidManifest.xml) einer App standardmäßig die folgende Zeile in Android Studio:

```
android:label="@string/app_name"
```

Dies weist Android an, nach einer `<string>` -Ressource mit dem Namen "app_name" zu suchen, die als Name für die App verwendet werden soll, wenn sie installiert oder in einem Startprogramm angezeigt wird.

Ein anderes Mal, wenn Sie eine `<string>` -Ressource aus einer XML-Datei in Android verwenden würden, wäre dies eine Layoutdatei. Das folgende Beispiel stellt eine Textansicht dar, die die `hello_world` definierte `hello_world` Zeichenfolge anzeigt:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world"/>
```

Sie können auch auf `<string>` -Ressourcen aus dem Java-Teil Ihrer App zugreifen. Um unseren `hello_world` String innerhalb einer Activity-Klasse von oben `hello_world` , verwenden Sie:

```
String helloWorld = getString(R.string.hello_world);
```

String-Array definieren

Um ein String-Array zu definieren, schreiben Sie in eine Ressourcendatei

res / values / filename.xml

```
<string-array name="string_array_name">
    <item>text_string</item>
    <item>@string/string_id</item>
</string-array>
```

zum Beispiel

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>
```

und benutze es aus Java gerne

```
String[] strings = getResources().getStringArray(R.array.string_array_example;
Log.i("TAG",Arrays.toString(strings)));
```

Ausgabe

```
I/TAG: [HelloWorld, Hello World!]
```

Bemaßungen definieren

Dimensionen werden normalerweise in einem Ressourcendateinamen `dimens.xml` . Sie werden mit einem `<dimen>` -Element definiert.

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>
```

Sie können verschiedene Einheiten verwenden:

- **sp:** Skalenunabhängige Pixel. Für Schriftarten
- **dp:** Dichteunabhängige Pixel. Für alles andere
- **pt:** Punkte
- **px:** Pixel
- **mm:** Millimeter
- **im:** Zoll

Dimensionen können jetzt in XML mit der Syntax `@dimen/name_of_the_dimension` .

Zum Beispiel:

```
<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>
```

Integer definieren

Integer werden normalerweise in einer Ressourcendatei namens `integers.xml` gespeichert, der Dateiname kann jedoch beliebig gewählt werden. Jede Ganzzahl wird mit einem `<integer>` -Element definiert, wie in der folgenden Datei gezeigt:

res / values / integers.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer name="max">100</integer>
</resources>
```

Integer können jetzt in XML mit der Syntax `@integer/name_of_the_integer` , wie im folgenden Beispiel gezeigt:

```
<ProgressBar
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:max="@integer/max"/>
```

Definieren Sie ein ganzzahliges Array

Um ein ganzzahliges Array zu definieren, schreiben Sie in eine Ressourcendatei

res / values / filename.xml

```
<integer-array name="integer_array_name">
    <item>integer_value</item>
```

```
    <item>@integer/integer_id</item>
</integer-array>
```

zum Beispiel

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>
```

und benutze es aus Java gerne

```
int[] values = getResources().getIntArray(R.array.fibo);
Log.i("TAG",Arrays.toString(values)));
```

Ausgabe

```
I/TAG: [0, 1, 1, 2, 3, 5]
```

Farben definieren

Farben werden normalerweise in einer Ressourcendatei mit dem Namen `colors.xml` im Ordner `/res/values/` gespeichert.

Sie werden durch `<color>` -Elemente definiert:

```
<?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>
```

Farben werden durch hexadezimale Farbwerte für jeden Farbkanal `(0 - FF)` in einem der folgenden Formate dargestellt:

- `#RGB`
- `#ARGB`
- `#RRGGBB`
- `#AARRGGBB`

Legende

- Ein Alpha-Kanal-0-Wert ist vollständig transparent, der FF-Wert ist undurchsichtig
- R - roter Kanal
- G - grüner Kanal
- B - blauer Kanal

Definierte Farben können in XML mit der folgenden Syntax `@color/name_of_the_color`

Zum Beispiel:

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/blackOverlay">
```

Farben im Code verwenden

In diesen Beispielen wird davon ausgegangen, dass `this` um eine Aktivitätsreferenz handelt. An seiner Stelle kann auch eine Kontextreferenz verwendet werden.

1.6

```
int color = ContextCompat.getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

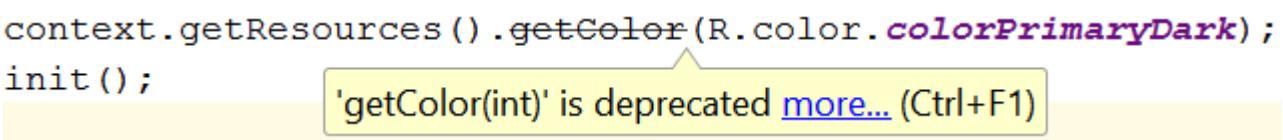
6,0

```
int color = this.getResources().getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

In der obigen Deklaration `colorPrimary` werden `colorPrimaryDark` und `colorAccent` verwendet, um Materialdesign-Farben zu definieren, die zum Definieren eines benutzerdefinierten Android-`styles.xml` in `styles.xml` . Sie werden automatisch hinzugefügt, wenn ein neues Projekt mit Android Studio erstellt wird.

Ressourcen ohne "veraltete" Warnungen erhalten

Bei Verwendung der Android-API 23 oder höher ist eine solche Situation häufig zu sehen:



Diese Situation wird durch die strukturelle Änderung der Android-API in Bezug auf das Abrufen der Ressourcen verursacht.
Nun die Funktion:

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

sollte benutzt werden. Die `android.support.v4` Bibliothek hat jedoch eine andere Lösung.

Fügen Sie der Datei `build.gradle` die folgende Abhängigkeit `build.gradle` :

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

Dann stehen alle Methoden aus der Support Library zur Verfügung:

```
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));
```

Darüber hinaus können weitere Methoden aus der Support-Bibliothek verwendet werden:

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

Definieren Sie eine Menüressource und verwenden Sie sie in Aktivität / Fragment

Definieren Sie ein Menü in **res / menu**

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/first_item_id"
        android:orderInCategory="100"
        android:title="@string/first_item_string"
        android:icon="@drawable/first_item_icon"
        app:showAsAction="ifRoom"/>

    <item
        android:id="@+id/second_item_id"
        android:orderInCategory="110"
        android:title="@string/second_item_string"
        android:icon="@drawable/second_item_icon"
        app:showAsAction="ifRoom"/>

</menu>
```

Weitere Konfigurationsoptionen finden Sie unter: Menüressource

Inside- **`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);
}
```

Um die obigen Methoden während der `getActivity().invalidateOptionsMenu();` der Ansicht aufzurufen, rufen Sie `getActivity().invalidateOptionsMenu();`

In **`Fragment`** ein zusätzlicher Aufruf erforderlich:

```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    setHasOptionsMenu(true);
    super.onCreateView(inflater, container, savedInstanceState);
}
```

String-Formatierung in strings.xml

Das Definieren von Strings in der Datei strings.xml ermöglicht auch die Formatierung von Strings. Der einzige Nachteil ist, dass der String mit dem folgenden Code behandelt werden muss, anstatt ihn einfach an ein Layout anzuhängen.

```
<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);
```

In obigem Beispiel
% 1 $ s
'%' trennt sich von normalen Zeichen,
'1' bezeichnet den ersten Parameter,
'$' wird als Trennzeichen zwischen Parameternummer und Typ verwendet.
's' bezeichnet den String-Typ ('d' wird für eine ganze Zahl verwendet)

Beachten Sie, dass `getString()` ist eine Methode der `Context` oder `Resources` , dh Sie können es direkt innerhalb eines verwenden können `Activity` oder sonst können Sie verwenden `getActivity().getString()` oder `getContext().getString()` sind.

Definieren Sie eine Farbstatusliste

Farbstatuslisten können als Farben verwendet werden, ändern sich jedoch je nach dem Status der Ansicht, für die sie verwendet werden.

Um eine zu definieren, erstellen Sie eine Ressourcendatei in `res/color/foo.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="#888888" android:state_enabled="false"/>
    <item android:color="@color/lightGray" android:state_selected="false"/>
    <item android:color="@android:color/white" />
</selector>
```

Elemente werden in der Reihenfolge bewertet, in der sie definiert sind, und es wird der erste Artikel verwendet, dessen angegebene Status mit dem aktuellen Status der Ansicht übereinstimmen. Daher empfiehlt es sich, am Ende ein Catch-All anzugeben, ohne dass Zustandsselektoren angegeben werden.

Jedes Element kann entweder ein Farb-Literal verwenden oder auf eine an anderer Stelle definierte Farbe verweisen.

String Plurals definieren

Um zwischen Plural- und Singularzeichenfolgen zu unterscheiden, können Sie in Ihrer Datei *strings.xml* einen Plural definieren und die unterschiedlichen Größen *auflisten* , wie im folgenden Beispiel gezeigt:

```
<?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>
```

Auf diese Definition kann aus Java-Code mit der Methode `getQuantityString()` der `Resources` Klasse `getQuantityString()` , wie im folgenden Beispiel gezeigt:

```
getResources().getQuantityString(R.plurals.hello_people, 3, 3);
```

Hierbei ist der erste Parameter `R.plurals.hello_people` der Ressourcenname. Der zweite Parameter (in diesem Beispiel `3`) wird verwendet, um die richtige `quantity` auszuwählen. Der dritte Parameter (in diesem Beispiel ebenfalls `3`) ist das Formatargument, das zum Ersetzen des Formatbezeichners `%d` .

Mögliche Mengenwerte (in alphabetischer Reihenfolge) sind:

```
few
many
one
other
two
zero
```

Es ist wichtig zu beachten, dass nicht alle Gebiete jede `quantity` . Zum Beispiel hat die chinesische Sprache kein Konzept `one` Elements. Englisch hat keinen `zero` , da er grammatikalisch dem `other` . Nicht unterstützte `quantity` werden von der IDE als Lint-Warnungen gekennzeichnet, verursachen jedoch keine Komplikationsfehler, wenn sie verwendet werden.

Importieren Sie ein in Ressourcen definiertes Array von Objekten

Es gibt Fälle, in denen benutzerdefinierte Objekte erstellt und in den Ressourcen der Anwendung definiert werden müssen. Solche Objekte können aus einfachen `Java` Typen bestehen, z. B. `Integer` , `Float` , `String` .

Hier ist das Beispiel, wie ein in Anwendungsressourcen definiertes Objekt importiert wird. Das Objekt `Category` enthält 3 Eigenschaften der Kategorie:

- ICH WÜRDE
- Farbe
- Name

Dieses `POJO` hat eine entsprechende Entsprechung in der Datei `categories.xml` , wobei für jedes Array dieselben Eigenschaften für jede Kategorie definiert sind.

1. **Erstellen Sie ein Modell für Ihr Objekt:**

```
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. **Erstellen Sie die Datei im Ordner `res/values` :**

```
categories.xml
```

3. **Stellen Sie jedes Modell zusammen, das aus Ressourcen besteht:**

```
<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. **Definieren Sie ein Array in der Ressourcendatei:**

```
<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. **Erstellen Sie eine Funktion, um sie zu importieren:**

```
@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 Patches

9 Flecken sind **streckbare** Bilder, bei denen die Bereiche, die gedehnt werden können, durch schwarze Markierungen auf einem transparenten Rand definiert werden.

Es gibt ein großes Tutorial hier .
Obwohl es so alt ist, ist es immer noch so wertvoll und es hat vielen von uns geholfen, die 9-Patch-Ausrüstung tief zu verstehen.

Leider wurde diese Seite in letzter Zeit vorübergehend nicht mehr angezeigt (derzeit ist sie wieder verfügbar).

Daher ist es notwendig, eine physische Kopie dieser Seite für Android-Entwickler auf unseren zuverlässigen Servern zu haben.

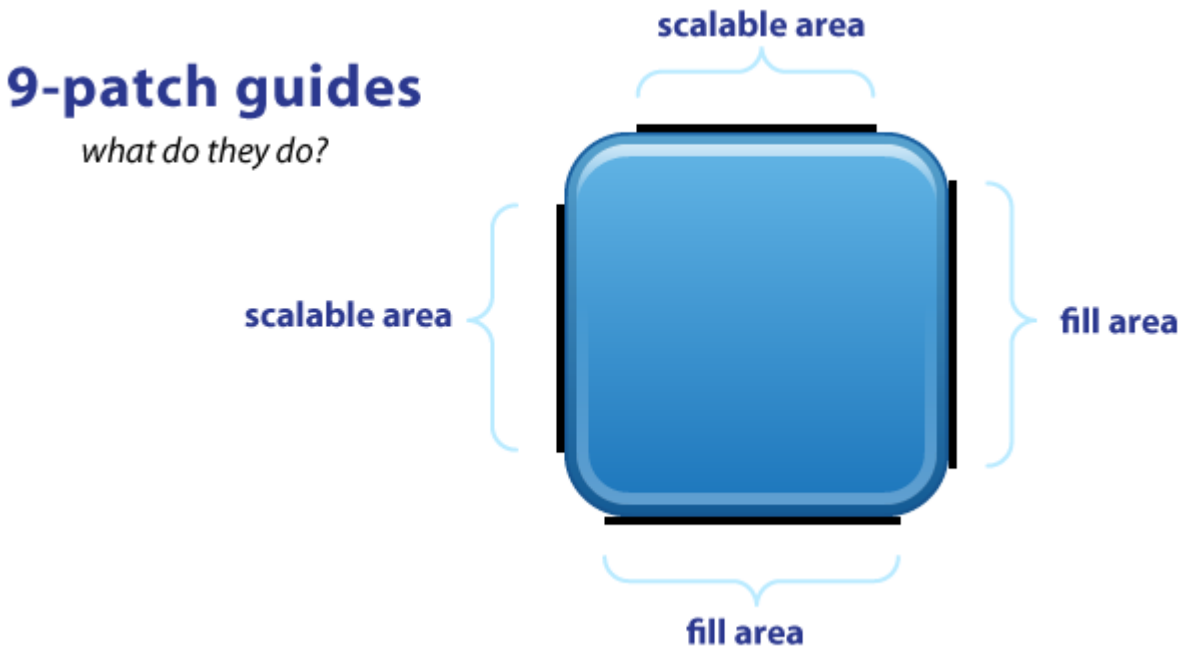
Hier ist es.

EIN EINFACHER HILFE ZUM 9-PATCH FÜR ANDROID UI 18. Mai 2011

Während ich an meiner ersten Android-App arbeitete, fand ich 9-Patch (aka 9.png) verwirrend und schlecht dokumentiert. Nach einer Weile habe ich endlich herausgefunden, wie es funktioniert und beschloss, etwas zusammen zu werfen, um anderen zu helfen, es herauszufinden.

Grundsätzlich verwendet 9-Patch png-Transparenz, um eine erweiterte Form von 9-Slice oder Scale9 zu erstellen. Bei den Hilfslinien handelt es sich um gerade schwarze 1-Pixel-Linien, die am Bildrand gezeichnet werden und die Skalierung und Füllung des Bildes bestimmen. Indem Sie Ihre Bilddatei name.9.png benennen, erkennt Android das 9.png-Format und verwendet die schwarzen Hilfslinien, um Ihre Bitmaps zu skalieren und zu füllen.

Hier ist eine grundlegende Übersichtskarte:



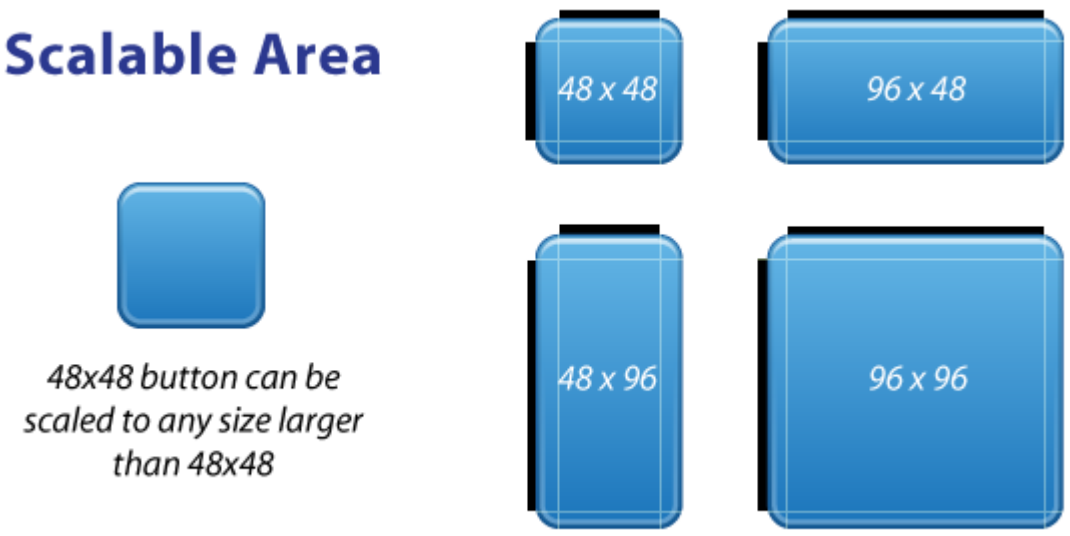
Wie Sie sehen, haben Sie auf jeder Seite Ihres Bildes eine Anleitung. Die TOP- und LEFT-Hilfslinien dienen zum Skalieren Ihres Bildes (dh 9-Slice), während die RECHTEN und UNTEN-Hilfslinien den Füllbereich definieren.

Die schwarzen Hilfslinien werden von Ihrem Bild abgeschnitten / entfernt - sie werden in der App nicht angezeigt. Die Hilfslinien müssen nur ein Pixel breit sein. Wenn Sie also eine 48 × 48-Taste wünschen, ist Ihr Png tatsächlich 50 × 50. Alles, was dicker als ein Pixel ist, bleibt Teil Ihres Bildes. (Meine Beispiele verfügen über 4 Pixel breite Hilfslinien zur besseren Sichtbarkeit. Sie sollten eigentlich nur 1 Pixel groß sein).

Ihre Guides müssen durchgehend schwarz sein (# 000000). Sogar ein geringfügiger Unterschied in Farbe (# 000001) oder Alpha führt zum Ausfall und zur normalen Dehnung. Dieser Fehler wird auch nicht offensichtlich sein *, er versagt lautlos! Ja. Ja wirklich. Jetzt wissen Sie.

Beachten Sie auch, dass der verbleibende Bereich der Einpixel-Kontur vollständig transparent sein muss. Dazu gehören die vier Ecken des Bildes - diese sollten immer klar sein. Dies kann ein größeres Problem sein, als Sie annehmen. Wenn Sie beispielsweise ein Bild in Photoshop skalieren, werden Anti-Alias-Pixel hinzugefügt, die fast unsichtbare Pixel enthalten können, was dazu führt, dass das Bild auch ausfällt *. Wenn Sie in Photoshop skalieren müssen, verwenden Sie die Einstellung "Nächster Nachbar" im Pulldown-Menü "Bild neu berechnen" (unten im Popup-Menü "Bildgröße"), um scharfe Kanten auf den Hilfslinien zu erhalten.

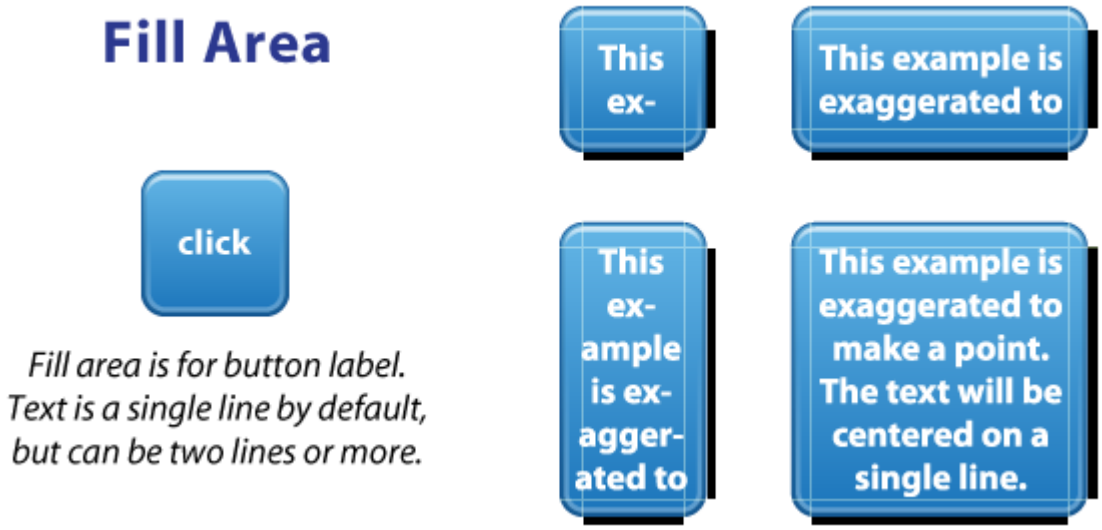
* (aktualisiert 1/2012) Dies ist eigentlich ein "Update" im neuesten Entwicklungskit. Zuvor manifestierte es sich, als alle Ihre anderen Bilder und Ressourcen plötzlich brechen, nicht das tatsächlich zerbrochene 9-Patch-Bild.



Die TOP- und LEFT-Hilfslinien werden verwendet, um den skalierbaren Teil Ihres Bildes zu definieren - LEFT für die Skalierungshöhe und TOP für die Skalierungsbreite. Wenn Sie ein Schaltflächenbild als Beispiel verwenden, bedeutet dies, dass sich die Schaltfläche innerhalb des schwarzen Bereichs horizontal und vertikal erstrecken kann. Alle anderen Elemente, z. B. die Ecken, bleiben gleich groß. Mit können Sie Schaltflächen verwenden, die auf jede Größe skaliert werden können und ein einheitliches Erscheinungsbild beibehalten.

Beachten Sie, dass 9-Patch-Bilder nicht verkleinert werden - sie werden nur vergrößert. Also fangen Sie am besten so klein wie möglich an.

Sie können auch Teile in der Mitte der Skalenlinie auslassen. Wenn Sie beispielsweise eine Schaltfläche mit einem scharfen, glänzenden Rand in der Mitte haben, können Sie einige Pixel in der Mitte der LEFT-Hilfslinie auslassen. Die horizontale Mittelachse Ihres Bildes wird nicht skaliert, sondern nur die darüber und darunter liegenden Teile, so dass Ihr scharfer Glanz nicht mit Anti-Aliasing oder Fuzzy-Effekt versehen wird.

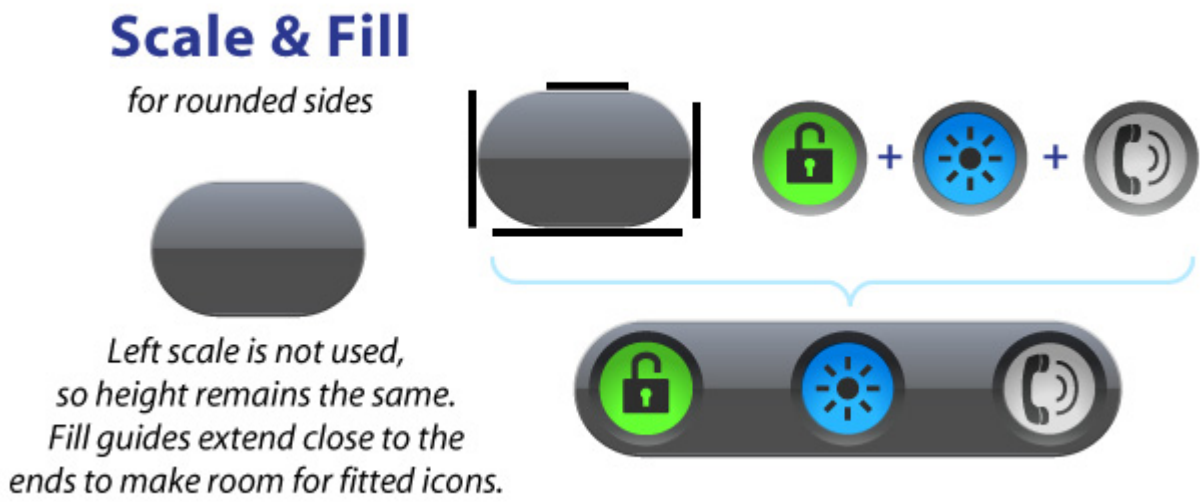


Hilfslinien für den Ausfüllbereich sind optional und bieten eine Möglichkeit, den Bereich für

Elemente wie Ihre Textbeschriftung festzulegen. Ausfüllen bestimmt, wie viel Platz in Ihrem Bild zum Platzieren von Text, Symbolen oder anderen Dingen vorhanden ist. 9-patch ist nicht nur für Knöpfe, sondern auch für Hintergrundbilder.

Das obige Beispiel für die Schaltfläche und das Etikett ist übertrieben, um die Idee des Füllens zu erläutern. Das Etikett ist nicht vollständig genau. Um ehrlich zu sein, habe ich noch nicht erfahren, wie Android mehrzeilige Beschriftungen ausführt, da es sich bei einer Schaltflächenbeschriftung normalerweise um eine einzelne Textzeile handelt.

Zum Abschluss möchten wir Ihnen zeigen, wie die Skalierungs- und Füllhilfslinien variieren können, beispielsweise ein LinearLayout mit Hintergrundbild und vollständig abgerundeten Seiten:



In diesem Beispiel wird die LEFT-Anleitung nicht verwendet, wir benötigen jedoch weiterhin eine Anleitung. Das Hintergrundbild wird nicht vertikal skaliert. es skaliert nur horizontal (basierend auf der TOP-Anleitung). Wenn Sie die Füllhilfslinien betrachten, erstrecken sich die Hilfslinien RECHTS und UNTEN über die Stelle, an der sie auf die gebogenen Kanten des Bildes treffen. Dadurch kann ich meine runden Knöpfe in der Nähe der Ränder des Hintergrunds platzieren, um ein enges, enges Aussehen zu erzielen.

So, das war es. 9-Patch ist super einfach, sobald Sie es bekommen. Die Skalierung ist zwar nicht perfekt, aber der Füllbereich und die mehrzeiligen Skalierungshilfslinien bieten mehr Flexibilität als herkömmliche 9-Slice- und Scale9-Formate. Probieren Sie es aus und Sie werden es schnell herausfinden.

Farbtransparenz (Alpha)

Hex-Deckkraftwerte

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        |
    -------------------------------
```

Wenn Sie 45% auf Rot setzen möchten.

```
<color name="red_with_alpha_45">#73FF0000</color>
```

Hexadezimalwert für Rot - # FF0000

Sie können 73 für 45% Deckkraft im Präfix hinzufügen - # 73FF0000

Mit der Datei strings.xml arbeiten

Eine Zeichenfolgenressource enthält Textzeichenfolgen für Ihre Anwendung mit optionaler Textgestaltung und -formatierung. Es gibt drei Arten von Ressourcen, die Ihrer Anwendung Strings zur Verfügung stellen können:

String

```
XML resource that provides a single string.
```

Syntax:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="string_name">text_string</string>
</resources>
```

Und diese Zeichenfolge im Layout verwenden:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/string_name" />
```

String Array

```
XML resource that provides an array of strings.
```

Syntax:

```
    <resources>
<string-array name="planets_array">
    <item>Mercury</item>
    <item>Venus</item>
    <item>Earth</item>
    <item>Mars</item>
</string-array>
```

Verwendungszweck

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

Anzahl Zeichenketten (Plurals)

```
XML resource that carries different strings for pluralization.
```

Syntax:

```
<?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>
```

Verwendungszweck:

```
int count = getNumberOfsongsAvailable();
Resources res = getResources();
String songsFound = res.getQuantityString(R.plurals.plural_name, count, count);
```

Ressourcen online lesen: https://riptutorial.com/de/android/topic/108/ressourcen

Kapitel 189: Retrofit2 mit RxJava

Examples

Retrofit2 mit RxJava

Fügen Sie zunächst relevante Abhängigkeiten in die Datei build.gradle ein.

```
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'
    ....
}
```

Dann erstellen Sie das Modell, das Sie erhalten möchten:

```
public class Server {
    public String name;
    public String url;
    public String apikey;
    public List<Site> siteList;
}
```

Erstellen Sie eine Schnittstelle, die Methoden für den Datenaustausch mit Remote-Server enthält:

```
public interface ApiServerRequests {

    @GET("api/get-servers")
    public Observable<List<Server>> getServers();
}
```

Dann erstellen Sie eine `Retrofit` Instanz:

```
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;
}
```

Rufen Sie dann an beliebiger Stelle des Codes die Methode auf:

```
apiRequests.getServers()
   .subscribeOn(Schedulers.io()) // the observable is emitted on io thread
   .observerOn(AndroidSchedulers.mainThread()) // Methods needed to handle request in
background thread
   .subscribe(new Subscriber<List<Server>>() {
       @Override
       public void onCompleted() {

       }

       @Override
       public void onError(Throwable e) {

       }

       @Override
       public void onNext(List<Server> servers) {
           //A list of servers is fetched successfully
       }
   });
```

Retrofit mit RxJava, um Daten asynchron abzurufen

RxJava ist eine Java VM-Implementierung von Reactive Extensions aus dem GitHub-Repo von RxJava *: Eine Bibliothek zum Erstellen asynchroner und ereignisbasierter Programme unter Verwendung beobachtbarer Sequenzen. Es erweitert das Observer-Muster, um Sequenzen von Daten / Ereignissen zu unterstützen, und fügt Operatoren hinzu, mit denen Sie deklarativ Sequenzen zusammenstellen können, während Sie Bedenken über Dinge wie Low-Level-Threading, Synchronisation, Thread-Sicherheit und gleichzeitige Datenstrukturen abschaffen.*

Retrofit ist ein typsicherer HTTP-Client für Android und Java. Damit können Entwickler das gesamte Netzwerk wesentlich einfacher machen. Als Beispiel werden wir JSON herunterladen und in RecyclerView als Liste anzeigen.

Fertig machen:

Fügen Sie der Build.gradle-Datei auf App-Ebene RxJava-, RxAndroid- und Retrofit-Abhängigkeiten hinzu: `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"
```

Definieren Sie ApiClient und ApiInterface, um Daten vom Server auszutauschen

```
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;
}}
```

öffentliche Schnittstelle ApiInterface {

```
@GET("repos/{org}/{repo}/issues")
Observable<List<Issue>> getIssues(@Path("org") String organisation,
                                  @Path("repo") String repositoryName,
                                  @Query("page") int pageNumber);}
```

Beachten Sie, dass getRepos () ein Observable zurückgibt und nicht nur eine Liste von Problemen.

Definieren Sie die Modelle

Ein Beispiel dafür wird gezeigt. Sie können kostenlose Dienste wie JsonSchema2Pojo oder dieses nutzen.

```
public class Comment {

@SerializedName("url")
@Expose
private String url;
@SerializedName("html_url")
@Expose
private String htmlUrl;

//Getters and Setters
}
```

Retrofit-Instanz erstellen

```
ApiInterface apiService = ApiClient.getInstance().create(ApiInterface.class);
```

Verwenden Sie dann diese Instanz, um Daten vom Server abzurufen

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

Jetzt haben Sie Daten erfolgreich mit Retrofit und RxJava von einem Server abgerufen.

Beispiel für verschachtelte Anforderungen: Mehrere Anforderungen, kombinieren Sie die Ergebnisse

Angenommen, wir verfügen über eine API, mit der wir Objekt-Metadaten in einer einzelnen Anforderung (`getAllPets`) `getAllPets` , und andere Anforderungen, die vollständige Daten einer einzelnen Ressource (`getSinglePet`) enthalten. Wie können wir alle in einer einzigen Kette abfragen?

```
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;
    }
}
```

}

Retrofit2 mit RxJava online lesen: https://riptutorial.com/de/android/topic/7632/retrofit2-mit-rxjava

Kapitel 190: RoboGuice

Examples

Einfaches Beispiel

RoboGuice ist ein Framework, das die Einfachheit und Leichtigkeit von Dependency Injection mit Googles Guice-Bibliothek für Android ermöglicht.

```
@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 );
    }
}
```

Installation für Gradle-Projekte

Fügen Sie dem Abhängigkeitsabschnitt Ihrer Gradle Build-Datei den folgenden Pom hinzu:

```
project.dependencies {
    compile 'org.roboguice:roboguice:3.+'
    provided 'org.roboguice:roboblender:3.+'
}
```

@ContentView-Anmerkung

Die @ContentView-Annotation kann verwendet werden, um die Entwicklung von Aktivitäten weiter zu erleichtern und die setContentView-Anweisung zu ersetzen:

```
@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-Anmerkung

Sie können beliebige Ressourcentypen, Strings, Animationen, Drawables usw. einfügen.

Um Ihre erste Ressource in eine Aktivität einzuspeisen, müssen Sie:

- Vererben von RoboActivity
- Kommentieren Sie Ihre Ressourcen mit @InjectResource

Beispiel

```
@InjectResource(R.string.app_name) String name;

@InjectResource(R.drawable.ic_launcher) Drawable icLauncher;

@InjectResource(R.anim.my_animation) Animation myAnimation;
```

@InjectView-Anmerkung

Sie können jede Ansicht mithilfe der Annotation @InjectView einfügen:

Sie müssen:

- Vererben von RoboActivity
- Legen Sie Ihre Inhaltsansicht fest
- Kommentieren Sie Ihre Ansichten mit @InjectView

Beispiel

```
@InjectView(R.id.textView1) TextView textView1;

@InjectView(R.id.textView2) TextView textView2;

@InjectView(R.id.imageView1) ImageView imageView1;
```

Einführung in RoboGuice

`RoboGuice` ist ein Framework, das die Einfachheit und Leichtigkeit von Dependency Injection mit Googles Guice-Bibliothek für Android ermöglicht.

RoboGuice 3 reduziert Ihren Anwendungscode. Weniger Code bedeutet weniger Möglichkeiten für Fehler. Ihr Code wird dadurch auch leichter nachvollzogen - Ihr Code ist nicht mehr mit den Mechanismen der Android-Plattform übersät, sondern kann sich jetzt auf die eigentliche Logik konzentrieren, die für Ihre Anwendung spezifisch ist.

Um Ihnen eine Idee zu geben, werfen Sie einen Blick auf dieses einfache Beispiel einer typischen Android- `Activity` :

```
class AndroidWay extends Activity {
        TextView name;
        ImageView thumbnail;
        LocationManager loc;
        Drawable icon;
        String myName;
```

```
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            name      = (TextView) findViewById(R.id.name);
            thumbnail = (ImageView) findViewById(R.id.thumbnail);
            loc       = (LocationManager) getSystemService(Activity.LOCATION_SERVICE);
            icon      = getResources().getDrawable(R.drawable.icon);
            myName    = getString(R.string.app_name);
            name.setText( "Hello, " + myName );
        }
    }
```

Dieses Beispiel umfasst 19 Zeilen Code. Wenn Sie versuchen, `onCreate()` , müssen Sie fünf Zeilen der Boilerplate-Initialisierung überspringen, um die einzige zu finden, die wirklich wichtig ist: `name.setText()` . Komplexe Aktivitäten können zu einem viel größeren Teil dieses Initialisierungscodes führen.

Vergleichen Sie dies mit derselben App, die mit `RoboGuice` :

```
 @ContentView(R.layout.main)
    class RoboWay extends RoboActivity {
        @InjectView(R.id.name)             TextView name;
        @InjectView(R.id.thumbnail)        ImageView thumbnail;
        @InjectResource(R.drawable.icon)   Drawable icon;
        @InjectResource(R.string.app_name) String myName;
        @Inject                            LocationManager loc;

        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            name.setText( "Hello, " + myName );
        }
    }
```

Das Ziel von RoboGuice ist es, Ihren Code zu Ihrer App zu machen, statt sich mit dem gesamten Initialisierungs- und Lebenszykluscode zu befassen, den Sie normalerweise in Android pflegen müssen.

Anmerkungen:

@ContentView-Anmerkung:

Die @ContentView-Annotation kann verwendet werden, um die Entwicklung von Aktivitäten weiter zu erleichtern und die setContentView-Anweisung zu ersetzen:

```
@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-Annotation:

Zuerst benötigen Sie eine Aktivität, die von RoboActivity erbt. Wenn Sie sich in Ihrem res / anim-Ordner eine Animation my_animation.xml befinden, können Sie diese nun mit einer Anmerkung referenzieren:

```
public class MyActivity extends RoboActivity {
    @InjectResource(R.anim.my_animation) Animation myAnimation;
    // the rest of your code
}
```

@Inject-Anmerkung:

Sie stellen sicher, dass Ihre Aktivitäten von RoboActivity ausgehen, und kommentieren Ihren System-Service-Member mit @Inject. Den Rest erledigt Roboguice.

```
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();
```

Neben Ansichten, Ressourcen, Diensten und anderen androidspezifischen Dingen kann RoboGuice alte Java-Objekte einfügen. Standardmäßig ruft Roboguice in Ihrem POJO keinen Argumentkonstruktor auf

```
class MyActivity extends RoboActivity {
    @Inject Foo foo; // this will basically call new Foo();
}
```

RoboGuice online lesen: https://riptutorial.com/de/android/topic/2563/roboguice

Kapitel 191: Robolectric

Einführung

Bei Unit-Tests wird ein Code entnommen und unabhängig getestet, ohne dass andere Abhängigkeiten oder Teile des Systems (z. B. die Datenbank) ausgeführt werden.

Robolectric ist ein Unit-Test-Framework, das das Android-SDK-Gefäß deaktiviert, sodass Sie die Entwicklung Ihrer Android-App testen können. Die Tests werden innerhalb von Sekunden in der JVM auf Ihrer Workstation ausgeführt.

Wenn Sie beide kombinieren, können Sie schnelle Tests mit dem JVN ausführen, wobei die Android-APIs verwendet werden.

Examples

Robolectric Test

```
@RunWith(RobolectricTestRunner.class)
public class MyActivityTest {

  @Test
  public void clickingButton_shouldChangeResultsViewText() throws Exception {
    MyActivity activity = Robolectric.setupActivity(MyActivity.class);

    Button button = (Button) activity.findViewById(R.id.button);
    TextView results = (TextView) activity.findViewById(R.id.results);

    button.performClick();
    assertThat(results.getText().toString()).isEqualTo("Robolectric Rocks!");
  }
}
```

Aufbau

Um robolectric zu konfigurieren, fügen Sie der `@Config` oder -methode die @ `@Config` Anmerkung hinzu.

Mit benutzerdefinierter Anwendungsklasse ausführen

```
@RunWith(RobolectricTestRunner.class)
@Config(application = MyApplication.class)
public final class MyTest {
}
```

Legen Sie das Ziel-SDK fest

```
@RunWith(RobolectricTestRunner.class)
@Config(sdk = Build.VERSION_CODES.LOLLIPOP)
public final class MyTest {
}
```

Mit benutzerdefiniertem Manifest ausführen

Wenn angegeben, sieht robolectric relativ zum aktuellen Verzeichnis. Der Standardwert ist `AndroidManifest.xml`

Ressourcen und Assets werden relativ zum Manifest geladen.

```
@RunWith(RobolectricTestRunner.class)
@Config(manifest = "path/AndroidManifest.xml")
public final class MyTest {
}
```

Verwenden Sie Qualifikatoren

Mögliche Qualifikationsmerkmale finden Sie in Android-Dokumenten .

```
@RunWith(RobolectricTestRunner.class)
public final class MyTest {

    @Config(qualifiers = "sw600dp")
    public void testForTablet() {
    }
}
```

Robolectric online lesen: https://riptutorial.com/de/android/topic/8743/robolectric

Kapitel 192: Rückruf-URL

Examples

Callback-URL-Beispiel mit Instagram OAuth

Eine der Anwendungsfälle von *Callback-URLs* ist OAuth. Lassen Sie uns tun dies mit einer Instagram Anmeldung: Wenn der Benutzer ihre Anmeldeinformationen ein und klickt auf die Schaltfläche *Anmelden,* Instagram die Anmeldeinformationen validieren und gibt eine `access_token` . Wir brauchen das `access_token` in unserer App.

Damit unsere App sich solche Links anhören kann, müssen wir unserer `Activity` eine Rückruf-URL hinzufügen. Wir können dies tun, indem Sie unserer `Activity` einen `<intent-filter/>` hinzufügen, der auf diese Callback-URL reagiert. Angenommen, unsere Rückruf-URL lautet `appSchema://appName.com` . Dann müssen Sie der gewünschten `Activity` in der Datei *Manifest.xml* die folgenden Zeilen hinzufügen:

```
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.BROWSABLE"/>
<data android:host="appName.com" android:scheme="appSchema"/>
```

Erklärung der obigen Zeilen:

- `<category android:name="android.intent.category.BROWSABLE"/>` dass die `<category android:name="android.intent.category.BROWSABLE"/>` von einem Webbrowser gestartet wird, um Daten anzuzeigen, auf die durch einen Link verwiesen wird.
- `<data android:host="appName.com" android:scheme="appSchema"/>` gibt unser Schema und den Host unserer Callback-URL an.
- Alles in allem bewirken diese Zeilen, dass die spezifische `Activity` geöffnet wird, wenn die Rückruf-URL in einem Browser aufgerufen wird.

Um nun den Inhalt der URL in Ihrer `Activity` `onResume()` , müssen Sie die `onResume()` -Methode wie folgt überschreiben:

```
@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.
}
```

Nun haben Sie das `access_token` von Instagram abgerufen, das in verschiedenen API-Endpunkten von Instagram verwendet wird.

Rückruf-URL online lesen: https://riptutorial.com/de/android/topic/4790/ruckruf-url

Kapitel 193: Rückwärtskompatible Apps erstellen

Examples

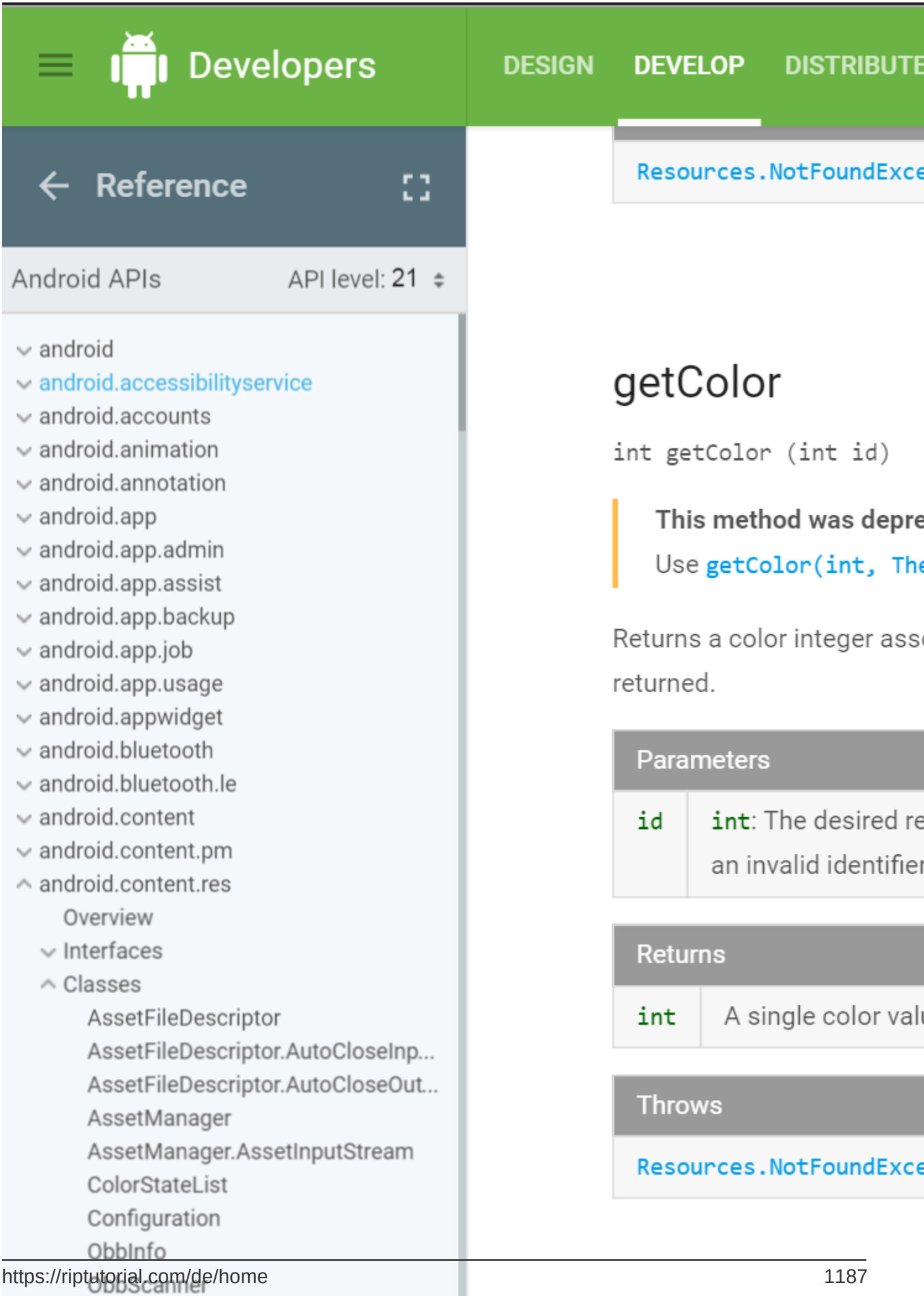
Wie veraltete API behandeln

Es ist unwahrscheinlich, dass ein Entwickler während eines Entwicklungsprozesses auf eine veraltete API stößt. Ein veraltetes Programmelement kann von Programmierern nicht verwendet werden, normalerweise, weil es gefährlich ist oder weil es eine bessere Alternative gibt. Compiler und Analysatoren (wie LINT) warnen, wenn ein nicht mehr unterstütztes Programmelement verwendet oder in nicht veraltetem Code überschrieben wird.

Eine veraltete API wird in Android Studio normalerweise mit einem Strikeout identifiziert. Im folgenden Beispiel ist die Methode `.getColor(int id)` veraltet:

```
getResources().getColor(R.color.colorAccent));
```

Wenn möglich, sollten Entwickler alternative APIs und Elemente verwenden. Sie können die Abwärtskompatibilität einer Bibliothek überprüfen, indem Sie die Android-Dokumentation für die Bibliothek aufrufen und den Abschnitt "In API-Ebene x" prüfen:


Developers
DESIGN
DEVELOP
DISTRIBUTE
Reference
Android APIs
API level: 21
android
android.accessibilityservice
android.accounts
android.animation
android.annotation
android.app
android.app.admin
android.app.assist
android.app.backup
android.app.job
android.app.usage
android.appwidget
android.bluetooth
android.bluetooth.le
android.content
android.content.pm
android.content.res
Overview
Interfaces
Classes
AssetFileDescriptor
AssetFileDescriptor.AutoCloseInp...
AssetFileDescriptor.AutoCloseOut...
AssetManager
AssetManager.AssetInputStream
ColorStateList
Configuration
ObbInfo
ObbScanner
Resources.NotFoundExce
getColor
int getColor (int id)
This method was depre
Use getColor(int, The
Returns a color integer ass
returned.
Parameters
id
int: The desired re
an invalid identifier
Returns
int
A single color valu
Throws
Resources.NotFoundExce

https://riptutorial.com/de/android/topic/4291/ruckwartskompatible-apps-erstellen

Kapitel 194: Rundfunkempfänger

Einführung

BroadcastReceiver (receiver) ist eine Android-Komponente, mit der Sie sich für System- oder Anwendungsereignisse registrieren können. Alle registrierten Empfänger für ein Ereignis werden von der Android-Laufzeitumgebung benachrichtigt, sobald dieses Ereignis auftritt.

B. eine Sendung mit der Meldung, dass der Bildschirm ausgeschaltet ist, der Akku schwach ist oder ein Bild aufgenommen wurde.

Anwendungen können auch Broadcasts initiieren, z. B. um anderen Anwendungen mitzuteilen, dass einige Daten auf das Gerät heruntergeladen wurden und zur Verfügung stehen.

Examples

Einführung in den Broadcast-Empfänger

Ein Broadcast-Empfänger ist eine Android-Komponente, mit der Sie sich für System- oder Anwendungsereignisse registrieren können.

Ein Empfänger kann über die `AndroidManifest.xml` Datei oder dynamisch über die `Context.registerReceiver()` -Methode registriert werden.

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

Hier habe ich ein Beispiel für `ACTION_BOOT_COMPLETED` das vom System ausgelöst wird, sobald der Android-Startvorgang abgeschlossen ist.

Sie können einen Empfänger in einer Manifestdatei wie folgt registrieren:

```
<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>
```

Jetzt wird das Gerät gebootet, die `onReceive()` -Methode wird aufgerufen und Sie können Ihre

Arbeit erledigen (z. B. einen Dienst starten, einen Alarm starten).

BroadcastReceiver-Grundlagen

BroadcastReceiver werden verwendet, um Broadcast- Intents zu empfangen, die vom Android-Betriebssystem, anderen Apps oder innerhalb derselben App gesendet werden.

Jeder Intent wird mit einem *Intent-Filter erstellt* , der eine String- *Aktion* erfordert. Zusätzliche Informationen können im Intent konfiguriert werden.

Ebenso registrieren sich BroadcastReceivers, um Intents mit einem bestimmten Intent-Filter zu empfangen. Sie können programmgesteuert registriert werden:

```
mContext.registerReceiver(new BroadcastReceiver() {
    @Override
   public void onReceive(Context context, Intent intent) {
      //Your implementation goes here.
   }
}, new IntentFilter("Some Action"));
```

oder in der Datei `AndroidManifest.xml` :

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="Some Action"/>
    </intent-filter>
</receiver>
```

Um die Absicht zu erhalten, setzen Sie die Aktion auf eine von Android OS, einer anderen App oder API oder in Ihrer eigenen Anwendung dokumentierte Aktion mit `sendBroadcast`

```
mContext.sendBroadcast(new Intent("Some Action"));
```

Darüber hinaus kann der Intent Informationen wie Strings, Primitives und *Parcelables enthalten* , die in `onReceive` angezeigt werden können.

LocalBroadcastManager verwenden

LocalBroadcastManager wird verwendet, um Broadcast Intents innerhalb einer Anwendung zu senden, ohne sie unerwünschten Listenern auszusetzen.

Die Verwendung von *LocalBroadcastManager* ist effizienter und sicherer als die `context.sendBroadcast()` Verwendung von `context.sendBroadcast()` , da Sie sich keine Sorgen um Broadcasts machen müssen, die von anderen Anwendungen gefälscht werden, was ein Sicherheitsrisiko darstellt.

Hier ist ein einfaches Beispiel für das Senden und Empfangen von lokalen Sendungen:

```
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-Rundfunkempfänger

Fügen Sie der Manifestdatei die Berechtigung hinzu

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

In deinem Fragment (oder in deiner Aktivität)

- Fügen Sie die Empfängermethode hinzu

```
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;
    }
};
```

Sendung registrieren

- Aufruf dieser Methode bei onResume ()

```
private void registerBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    manager.registerReceiver(mBluetoothStatusChangedReceiver, new
```

```
IntentFilter(Constants.BROADCAST_BLUETOOTH_STATE));
}
```

Broadcast abmelden

- Aufruf dieser Methode bei onPause ()

```
private void unregisterBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    // Beacon□□□
    manager.unregisterReceiver(mBluetoothStatusChangedReceiver);
}
```

Programmgesteuertes Aktivieren und Deaktivieren eines Broadcast-Empfängers

Um einen `BroadcastReceiver` zu aktivieren oder zu deaktivieren, benötigen wir einen Verweis auf den `PackageManager` und benötigen ein `ComponentName` Objekt, das die Klasse des Empfängers enthält, den wir aktivieren / deaktivieren möchten:

```
ComponentName componentName = new ComponentName(context, MyBroadcastReceiver.class);
PackageManager packageManager = context.getPackageManager();
```

Jetzt können wir die folgende Methode aufrufen, um den `BroadcastReceiver` zu aktivieren:

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP);
```

Oder Sie können stattdessen `COMPONENT_ENABLED_STATE_DISABLED` , um den Empfänger zu deaktivieren:

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
    PackageManager.DONT_KILL_APP);
```

BroadcastReceiver zur Behandlung von BOOT_COMPLETED-Ereignissen

Das folgende Beispiel zeigt, wie ein `BroadcastReceiver` wird, der `BOOT_COMPLETED` Ereignisse empfangen `BOOT_COMPLETED` . Auf diese Weise können Sie einen `Service` starten oder eine `Activity` starten, sobald das Gerät hochgefahren wurde.

Sie können auch `BOOT_COMPLETED` Ereignisse verwenden, um Ihre Alarme wiederherzustellen, da sie beim Ausschalten des Geräts zerstört werden.

HINWEIS: Der Benutzer muss die Anwendung mindestens einmal gestartet haben, bevor Sie die Aktion `BOOT_COMPLETED` empfangen `BOOT_COMPLETED` .

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.test.example" >
    ...
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    ...

    <application>
        ...

        <receiver android:name="com.test.example.MyCustomBroadcastReceiver">
        <intent-filter>
                <!-- REGISTER TO RECEIVE BOOT_COMPLETED EVENTS -->
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

MyCustomBroadcastReceiver.java

```
public class MyCustomBroadcastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if(action != null) {
            if (action.equals(Intent.ACTION_BOOT_COMPLETED) ) {
                // TO-DO: Code to handle BOOT COMPLETED EVENT
                // TO-DO: I can start an service.. display a notification... start an activity
            }
        }
    }
}
```

Beispiel für einen LocalBroadcastManager

Ein BroadcastReceiver ist im Wesentlichen ein Mechanismus, um Intents durch das Betriebssystem zu leiten, um bestimmte Aktionen auszuführen. Ein klassisches Definitionswesen

> "Ein Broadcast-Empfänger ist eine Android-Komponente, mit der Sie sich für System- oder Anwendungsereignisse registrieren können."

LocalBroadcastManager ist eine Möglichkeit, Broadcasts innerhalb eines Anwendungsprozesses zu senden oder zu empfangen. Dieser Mechanismus hat viele Vorteile

1. Da die Daten im Anwendungsprozess verbleiben, können die Daten nicht durchgesickert werden.
2. LocalBroadcasts werden schneller aufgelöst, da die Auflösung einer normalen Übertragung zur Laufzeit im gesamten Betriebssystem erfolgt.

Ein einfaches Beispiel für einen LocalBroastManager ist:

SenderActivity

```
  Intent intent = new Intent("anEvent");
  intent.putExtra("key", "This is an event");
  LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
```

ReceiverActivity

1. Einen Empfänger registrieren

```
LocalBroadcastManager.getInstance(this).registerReceiver(aLBReceiver,
              new IntentFilter("anEvent"));
```

2. Ein konkretes Objekt zum Durchführen einer Aktion, wenn der Empfänger aufgerufen wird

```
private BroadcastReceiver aLBReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        // perform action here.
    }
};
```

3. Registrierung aufheben, wenn die Ansicht nicht mehr sichtbar ist.

```
@Override
protected void onPause() {
    // Unregister since the activity is about to be closed.
    LocalBroadcastManager.getInstance(this).unregisterReceiver(aLBReceiver);
    super.onDestroy();
}
```

Kommunizieren Sie zwei Aktivitäten über einen benutzerdefinierten Broadcast-Empfänger

Sie können zwei Aktivitäten kommunizieren, sodass Aktivität A über ein Ereignis in Aktivität B benachrichtigt werden kann.

Tätigkeit 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
    }
};
```

Tätigkeit B

```
final String eventName = "your.package.goes.here.EVENT";

private void launchEvent() {
    Intent eventIntent = new Intent(eventName);
    this.sendBroadcast(eventIntent);
}
```

Natürlich können Sie der Sendung weitere Informationen hinzufügen und der Absicht, die zwischen den Aktivitäten übertragen wird, weitere Extras hinzufügen. Nicht hinzugefügt, um das Beispiel so einfach wie möglich zu halten.

Sticky Broadcast

Wenn wir die Methode sendStickyBroadcast (intent) verwenden, ist die entsprechende Absicht klebrig, dh die von Ihnen gesendete Absicht bleibt nach Abschluss der Übertragung erhalten. Ein StickyBroadcast ist, wie der Name schon sagt, ein Mechanismus zum Lesen der Daten aus einer Sendung, nachdem die Sendung abgeschlossen ist. Dies kann in einem Szenario verwendet werden, in dem Sie den Wert eines Schlüssels in der Absicht vor dem Start der `Activity's onCreate()` einer `Activity's onCreate()` überprüfen möchten.

```
Intent intent = new Intent("com.org.action");
intent.putExtra("anIntegerKey", 0);
sendStickyBroadcast(intent);
```

Geordnete Sendungen verwenden

Bestellte Broadcasts werden verwendet, wenn Sie eine Priorität für Broadcast-Listener festlegen müssen.

In diesem Beispiel `firstReceiver` Broadcast immer vor einem `secondReceiver` :

```
final int highPriority = 2;
final int lowPriority = 1;
final String action = "action";

// intent filter for first receiver with high priority
final IntentFilter firstFilter = new IntentFilter(action);
```

```
first Filter.setPriority(highPriority);
final BroadcastReceiver firstReceiver = new MyReceiver();

// intent filter for second receiver with low priority
final IntentFilter secondFilter = new IntentFilter(action);
secondFilter.setPriority(lowPriority);
final BroadcastReceiver secondReceiver = new MyReceiver();

// register our receivers
context.registerReceiver(firstReceiver, firstFilter);
context.registerReceiver(secondReceiver, secondFilter);

// send ordered broadcast
context.sendOrderedBroadcast(new Intent(action), null);
```

Außerdem kann der Rundfunkempfänger die bestellte Sendung abbrechen:

```
@Override
public void onReceive(final Context context, final Intent intent) {
    abortBroadcast();
}
```

In diesem Fall erhalten alle Empfänger mit niedrigerer Priorität keine Broadcast-Nachricht.

Android-Status angehalten

Ab Android 3.1 werden alle Anwendungen nach der Installation in den Stoppzustand versetzt. Im gestoppten Zustand wird die Anwendung aus keinem Grund ausgeführt, außer durch manuelles Starten einer Aktivität oder durch eine **explizite** Absicht, die eine Aktivität, einen Dienst oder einen Broadcast anspricht.

Bitte beachten Sie beim Schreiben einer System-App, die APKs direkt installiert, dass die neu installierte APP keine Broadcasts empfängt, bis sie in einen nicht angehaltenen Status versetzt wird.

Eine einfache Möglichkeit, eine App zu aktivieren, ist das Senden einer expliziten Übertragung an diese App. `INSTALL_REFERRER` meisten Apps `INSTALL_REFERRER` implementieren, können wir es als Verbindungspunkt verwenden

Scannen Sie das Manifest der installierten App und senden Sie eine explizite Übertragung an jeden Empfänger:

```
Intent intent = new Intent();
intent.addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES);
intent.setComponent(new ComponentName(packageName, fullClassName));
sendBroadcast(intent);
```

Rundfunkempfänger online lesen: https://riptutorial.com/de/android/topic/1460/rundfunkempfanger

Kapitel 195: Schnelle Möglichkeit, Retrolambda auf einem Android-Projekt einzurichten.

Einführung

Retrolambda ist eine Bibliothek, die die Verwendung von Java 8-Lambda-Ausdrücken, Methodenreferenzen und try-with-resources-Anweisungen auf Java 7, 6 oder 5 ermöglicht.

Das Retrolambda-Plug-In von Gradle ermöglicht die Integration von Retrolambda in einen Gradle-basierten Build. Dies ermöglicht beispielsweise die Verwendung dieser Konstrukte in einer Android-Anwendung, da die Android-Standardentwicklung derzeit noch keine Unterstützung für Java 8 bietet.

Examples

Setup und Beispiel zur Verwendung:

Setup-Schritte:

1. Laden Sie jdk8 herunter und installieren Sie es.
2. Fügen Sie dem Hauptbuild.gradle Ihres Projekts Folgendes hinzu

```
buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
        classpath 'me.tatarka:gradle-retrolambda:3.2.3'
    }
}
```

3. Fügen Sie dies jetzt dem build.gradle Ihres Anwendungsmoduls hinzu

```
apply plugin: 'com.android.application' // or apply plugin: 'java'
apply plugin: 'me.tatarka.retrolambda'
```

4. Fügen Sie diese Zeilen zum build.gradle Ihres Anwendungsmoduls hinzu, um die IDE über die Sprachebene zu informieren:

```
android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
```

```
    }
}
```

Beispiel:

Also Dinge wie diese:

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        log("Clicked");
    }
});
```

Werde das:

```
button.setOnClickListener(v -> log("Clicked"));
```

Schnelle Möglichkeit, Retrolambda auf einem Android-Projekt einzurichten. online lesen: https://riptutorial.com/de/android/topic/8822/schnelle-moglichkeit--retrolambda-auf-einem-android-projekt-einzurichten-

Kapitel 196: Schnittstellen

Examples

Benutzerdefinierter Listener

Schnittstelle definieren

```
//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);
}
```

Listener erstellen

Im nächsten Schritt müssen wir eine Instanzvariable im Objekt definieren, die den Callback über `MyCustomListener` . Und fügen Sie Setter für unseren Hörer hinzu.

```
public class SampleClassB {
    private MyCustomListener listener;

    public void setMyCustomListener(MyCustomListener listener) {
        this.listener = listener;
    }
}
```

Listener implementieren

In einer anderen Klasse können wir jetzt eine Instanz von `SampleClassB` erstellen.

```
public class SomeActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
    }
}
```

Als nächstes können wir unseren Listener auf zwei Arten auf `sampleClass` :

implementiert in unserer Klasse `MyCustomListener` :

```
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) {

    }
}
```

oder einfach eine anonyme innere Klasse instanziieren:

```
public class SomeActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
        sampleClass.setMyCustomListener(new MyCustomListener() {

            @Override
            public void onSuccess(List<Bitmap> bitmapList) {

            }

            @Override
            public void onFailure(String error) {

            }
        });
    }
}
```

Trigger Listener

```
public class SampleClassB {
    private MyCustomListener listener;

    public void setMyCustomListener(MyCustomListener listener) {
        this.listener = listener;
    }

    public void doSomething() {
        fetchImages();
    }

    private void fetchImages() {
        AsyncImagefetch imageFetch = new AsyncImageFetch();
        imageFetch.start(new Response<Bitmap>() {
```

```
            @Override
            public void onDone(List<Bitmap> bitmapList, Exception e) {
                //do some stuff if needed

                //check if listener is set or not.
                if(listener == null)
                    return;
                //Fire proper event. bitmapList or error message will be sent to
                //class which set listener.
                if(e == null)
                    listener.onSuccess(bitmapList);
                else
                    listener.onFailure(e.getMessage());
            }
        });
    }
}
```

Grundlegender Hörer

Das "Listener" - oder "Observer" -Muster ist die häufigste Strategie zum Erstellen asynchroner Rückrufe in der Android-Entwicklung.

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

Nun zu Ihrer Aktivität:

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

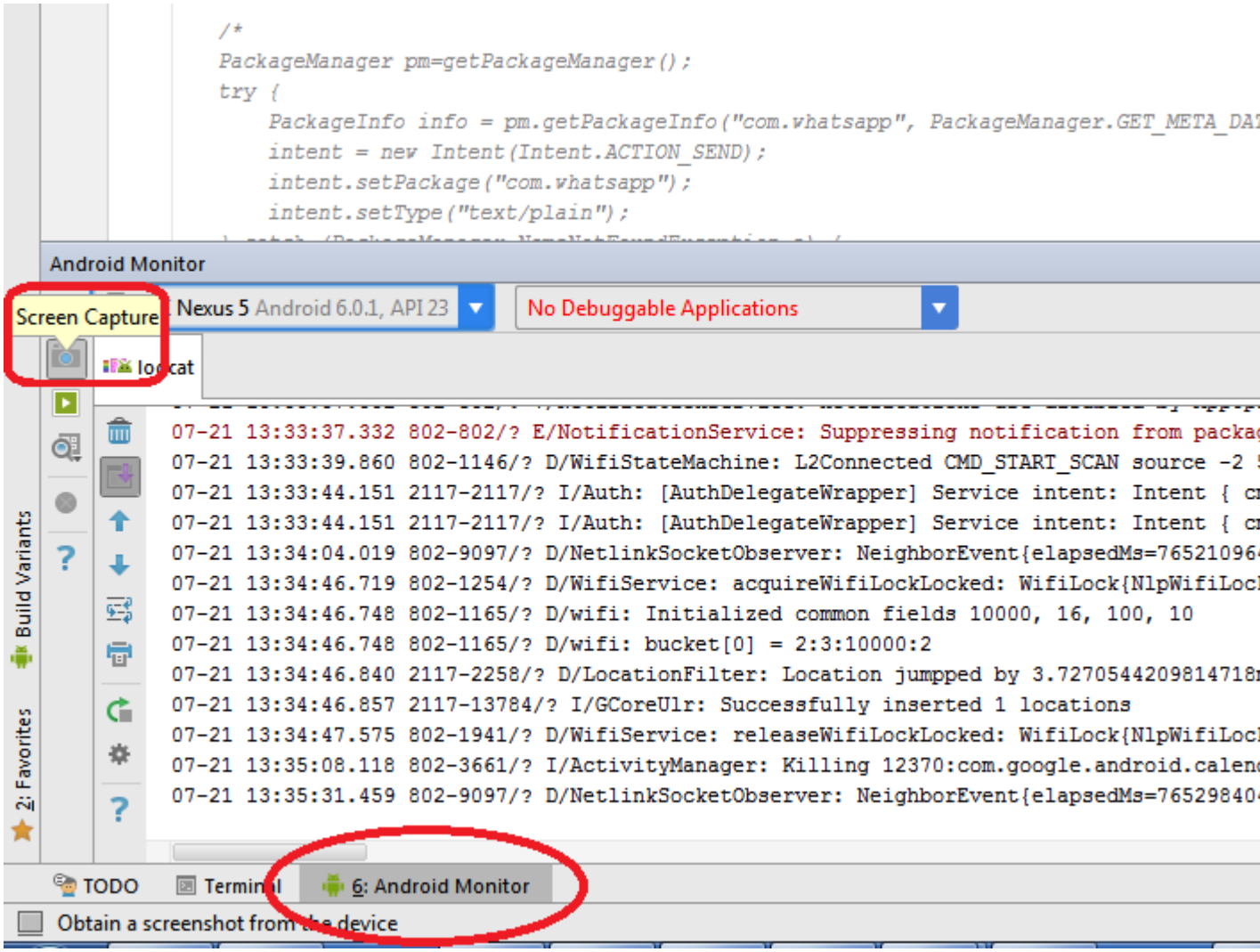
Schnittstellen online lesen: https://riptutorial.com/de/android/topic/1785/schnittstellen

Kapitel 197: Screenshots aufnehmen

Examples

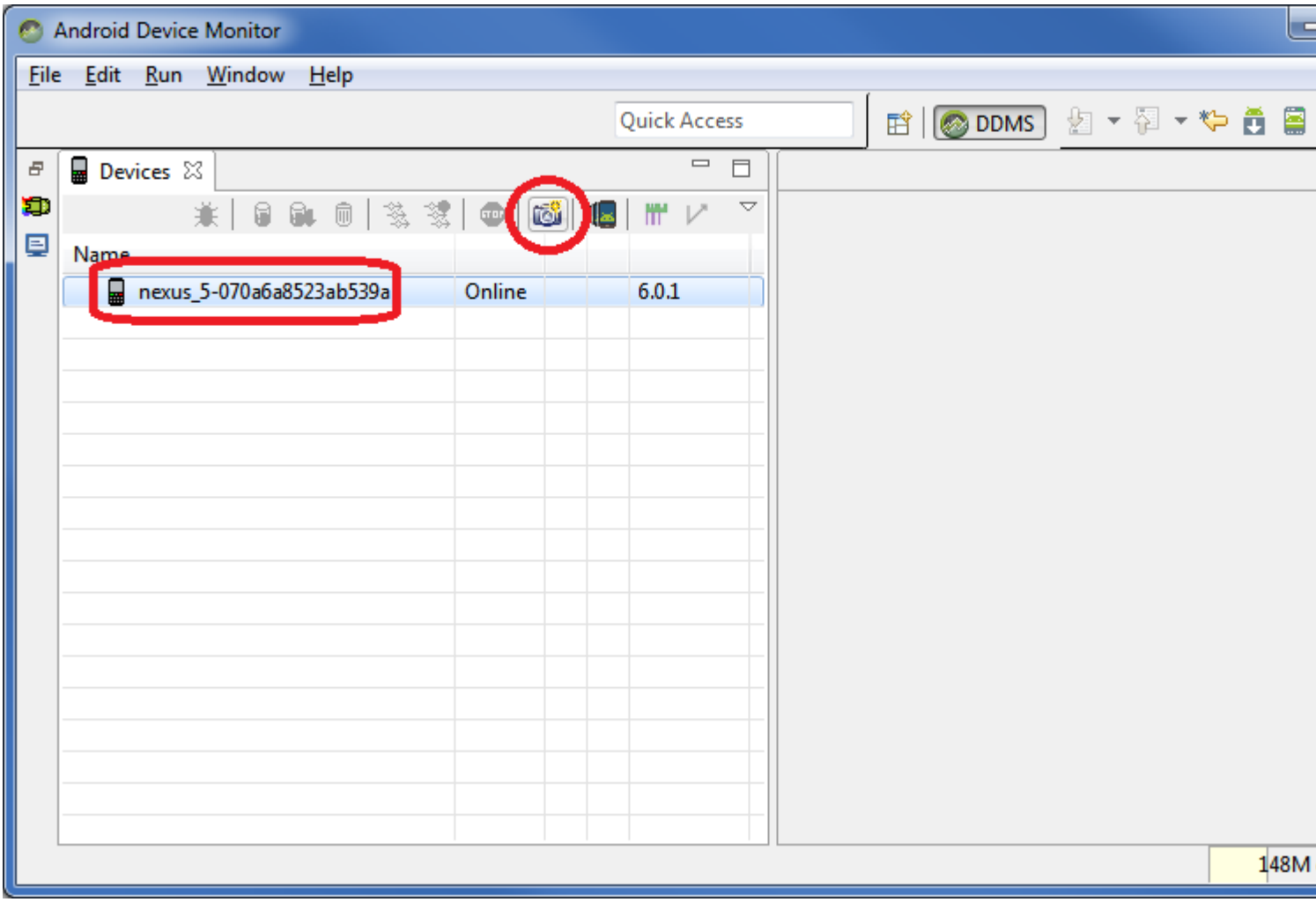
Screenshot über Android Studio aufnehmen

1. Öffnen Sie die Android Monitor-Registerkarte
2. Klicken Sie auf die Schaltfläche Screen Capture



Screenshot über Android-Gerätemonitor aufnehmen

1. Öffnen Sie den Android- *Gerätemonitor* (z. *B. C: <ANDROID_SDK_LOCATION> \ tools \ monitor.bat*).
2. Wählen sie ihren Gerätetyp
3. Klicken Sie auf die Schaltfläche Screen Capture



Screenshot über ADB aufnehmen

Das folgende Beispiel speichert einen Screenshot im internen Speicher von Devices.

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

Screenshot über ADB aufnehmen und direkt auf Ihrem PC speichern

Wenn Sie Linux (oder Windows mit Cygwin) verwenden, können Sie Folgendes ausführen:

```
adb shell screencap -p | sed 's/\r$//' > screenshot.png
```

Einen Screenshot einer bestimmten Ansicht erstellen

Wenn Sie einen Screenshot einer bestimmten View `v` erstellen möchten, können Sie den folgenden Code verwenden:

```
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();
}
```

Screenshots aufnehmen online lesen: https://riptutorial.com/de/android/topic/4506/screenshots-aufnehmen

Kapitel 198: SearchView

Examples

Appcompat SearchView mit dem RxBindings-Watcher

build.gradle :

```
dependencies {
    compile 'com.android.support:appcompat-v7:23.3.0'
    compile 'com.jakewharton.rxbinding:rxbinding-appcompat-v7:0.4.0'
}
```

menu / menu.xml :

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/action_search" android:title="Search"
        android:icon="@android:drawable/ic_menu_search"
        app:actionViewClass="android.support.v7.widget.SearchView"
        app:showAsAction="always"/>
</menu>
```

MainActivity.java :

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);

    MenuItem searchMenuItem = menu.findItem(R.id.action_search);
    setupSearchView(searchMenuItem );

    return true;
}

private void setupSearchView(MenuItem searchMenuItem) {
    SearchView searchView = (SearchView) searchMenuItem.getActionView();
    searchView.setQueryHint(getString(R.string.search_hint)); // your hint here

    SearchAdapter searchAdapter = new SearchAdapter(this);
    searchView.setSuggestionsAdapter(searchAdapter);

    // optional: set the letters count after which the search will begin to 1
    // the default is 2
    try {
        int autoCompleteTextViewID =
getResources().getIdentifier("android:id/search_src_text", null, null);
        AutoCompleteTextView searchAutoCompleteTextView = (AutoCompleteTextView)
searchView.findViewById(autoCompleteTextViewID);
        searchAutoCompleteTextView.setThreshold(1);
    } catch (Exception e) {
        Logs.e(TAG, "failed to set search view letters threshold");
```

```
    }

    searchView.setOnSearchClickListener(v -> {
        // optional actions to search view expand
    });
    searchView.setOnCloseListener(() -> {
        // optional actions to search view close
        return false;
    });

    RxSearchView.queryTextChanges(searchView)
            .doOnEach(notification -> {
                CharSequence query = (CharSequence) notification.getValue();
                searchAdapter.filter(query);
            })
            .debounce(300, TimeUnit.MILLISECONDS) // to skip intermediate letters
            .flatMap(query -> MyWebService.search(query)) // make a search request
            .retry(3)
            .subscribe(results -> {
                searchAdapter.populateAdapter(results);
            });

    //optional: collapse the searchView on close
    searchView.setOnQueryTextFocusChangeListener((view, queryTextFocused) -> {
        if (!queryTextFocused) {
            collapseSearchView();
        }
    });
}
```

SearchAdapter.java

```
public class SearchAdapter extends CursorAdapter {
    private List<SearchResult> items = Collections.emptyList();

    public SearchAdapter(Activity activity) {
        super(activity, null, CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
    }

    public void populateAdapter(List<SearchResult> items) {
        this.items = items;
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            c.addRow(new Object[]{i});
        }
        changeCursor(c);
        notifyDataSetChanged();
    }

    public void filter(CharSequence query) {
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            SearchResult result = items.get(i);
            if (result.getText().startsWith(query.toString())) {
                c.addRow(new Object[]{i});
            }
        }
        changeCursor(c);
        notifyDataSetChanged();
    }
```

```
    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        ViewHolder holder = (ViewHolder) view.getTag();
        int position = cursor.getPosition();
        if (position < items.size()) {
            SearchResult result = items.get(position);
            // bind your view here
        }
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        LayoutInflater inflater = (LayoutInflater) context
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        View v = inflater.inflate(R.layout.search_list_item, parent, false);
        ViewHolder holder = new ViewHolder(v);

        v.setTag(holder);
        return v;
    }

    private static class ViewHolder {
        public final TextView text;

        public ViewHolder(View v) {
            this.text= (TextView) v.findViewById(R.id.text);
        }
    }
}
```

Suchansicht in der Symbolleiste mit Fragment

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);
    }
}
```

Referenz-Screenshot:

Hello Android

menu.xml extrahiert wurde, müssen wir `menu.xml` , dass es vollständig von dem Stil abhängt, der auf die zugrunde liegende Toolbar angewendet wird. Um die Symbolleiste zu gestalten, führen Sie die folgenden Schritte aus.

Erstellen Sie einen Stil in der `styles.xml`

```
<style name="ActionBarThemeOverlay">
     <item name="android:textColorPrimary">@color/prim_color</item>
     <item name="colorControlNormal">@color/normal_color</item>
     <item name="colorControlHighlight">@color/high_color</item>
     <item name="android:textColorHint">@color/hint_color</item>
</style>
```

Wenden Sie den Stil auf die Symbolleiste an.

```
<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" />
```

Dadurch erhalten alle Ansichten, die der Symbolleiste entsprechen, die gewünschte Farbe (Zurück-Schaltfläche, Menüsymbole und SearchView).

SearchView online lesen: https://riptutorial.com/de/android/topic/4786/searchview

Kapitel 199: SensorManager

Examples

Sensorereignisse abrufen

Abrufen von Sensorinformationen von den integrierten Sensoren:

```
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
    }
}
```

Sensortransformation zum Weltkoordinatensystem

Die von Android zurückgegebenen Sensorwerte beziehen sich auf das Koordinatensystem des Telefons (z. B. + Y zeigt zur Oberseite des Telefons). Wir können diese Sensorwerte mithilfe der Rotationsmatrix des Sensor-Managers in ein Weltkoordinatensystem (z. B. + Y-Punkte in Richtung des magnetischen Nordens, tangential zum Boden) umwandeln

Zuerst müssen Sie die Matrizen / Arrays deklarieren und initialisieren, in denen Daten gespeichert werden (dies ist beispielsweise in der Methode `onCreate` möglich):

```
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];
```

Als Nächstes müssen wir Änderungen in den Sensorwerten erkennen, in den entsprechenden Arrays speichern (wenn wir sie später verwenden möchten) und dann die Rotationsmatrix und die resultierende Transformation in Weltkoordinaten berechnen:

```
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];

}
```

Stellen Sie mit dem Beschleunigungssensor fest, ob Ihr Gerät statisch ist oder nicht

Fügen Sie der `onCreate()` / `onResume()` Methode den folgenden Code `onCreate()` :

```
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);
```

Möglicherweise müssen Sie die Empfindlichkeit anpassen, indem Sie die `movementThreshold` durch Ausprobieren anpassen. Überschreiben Sie anschließend die `onSensorChanged()` -Methode wie folgt:

```
@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);
    }
}
```

Wenn Sie verhindern möchten, dass Ihre App auf Geräten ohne Beschleunigungssensor installiert wird, müssen Sie Ihrem Manifest die folgende Zeile hinzufügen:

```
<uses-feature android:name="android.hardware.sensor.accelerometer" />
```

SensorManager online lesen: https://riptutorial.com/de/android/topic/3344/sensormanager

Kapitel 200: ShortcutManager

Examples

Dynamic Launcher-Verknüpfungen

```
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));
```

Wir können alle dynamischen Verknüpfungen einfach entfernen, indem Sie Folgendes aufrufen: -

```
 shortcutManager.removeAllDynamicShortcuts();
```

Wir können vorhandene Dynamische Shorcuts mit aktualisieren

```
shortcutManager.updateShortcuts(Arrays.asList(shortcut);
```

Bitte beachten Sie, dass `setDynamicShortcuts(List)` verwendet wird, um die gesamte Liste der dynamischen Verknüpfungen neu zu definieren, `addDynamicShortcuts(List)` dient zum Hinzufügen dynamischer Verknüpfungen zu einer Liste dynamischer Verknüpfungen

ShortcutManager online lesen: https://riptutorial.com/de/android/topic/7661/shortcutmanager

Kapitel 201: Sichere SharedPreferences

Einführung

Gemeinsame Einstellungen sind auf **Schlüsselwerten basierende XML-Dateien** . Es befindet sich unter `/data/data/package_name/shared_prefs/<filename.xml>` .

So kann ein Benutzer mit root-Berechtigungen zu diesem Speicherort navigieren und seine Werte ändern. Wenn Sie die Werte in Ihren gemeinsamen Einstellungen schützen möchten, können Sie einen einfachen Verschlüsselungs- und Entschlüsselungsmechanismus schreiben.

Sie sollten wissen, dass Shared Preferences niemals sicher erstellt wurden. Es ist nur eine einfache Methode, um Daten zu speichern.

Syntax

1. public static String encrypt (String-Eingabe);
2. public static String decrypt (Zeicheneingabe);

Parameter

Parameter	Definition
Eingang	Stringwert zum Verschlüsseln oder Entschlüsseln.

Bemerkungen

Gemeinsame Einstellungen wurden nie als sicher erstellt, es ist nur eine einfache Methode, um Daten zu erhalten.

Es ist keine gute Idee, gemeinsam genutzte Voreinstellungen zum Speichern kritischer Informationen wie Benutzeranmeldeinformationen zu verwenden. Um Benutzeranmeldeinformationen (z. B. Kennwörter) zu speichern, müssen Sie andere Methoden verwenden, beispielsweise den `AccountManager` Android.

Examples

Sichern einer gemeinsamen Einstellung

Einfacher Codec

Um das Funktionsprinzip zu veranschaulichen, können wir die einfache Ver- und Entschlüsselung wie folgt verwenden.

```
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));
}
```

Implementierungstechnik

```
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);
```

Sichere SharedPreferences online lesen: https://riptutorial.com/de/android/topic/9887/sichere-sharedpreferences

Kapitel 202: Sichere SharedPreferences

Einführung

Gemeinsame Einstellungen sind auf **Schlüsselwerten basierende XML-Dateien** . Es befindet sich unter / data / data / package_name / shared_prefs / <Dateiname.xml>.

So kann ein Benutzer mit root-Berechtigungen zu diesem Speicherort navigieren und seine Werte ändern. Wenn Sie die Werte in Ihren gemeinsamen Einstellungen schützen möchten, können Sie einen einfachen Verschlüsselungs- und Entschlüsselungsmechanismus schreiben.

Sie sollten wissen, dass Shared Preferences niemals sicher erstellt wurden. Es ist nur eine einfache Methode, um Daten zu speichern.

Syntax

1. public static String encrypt (String-Eingabe);
2. public static String decrypt (Zeicheneingabe);

Parameter

Parameter	Definition
Eingang	Stringwert zum Verschlüsseln oder Entschlüsseln.

Bemerkungen

Gemeinsame Einstellungen wurden nie als sicher erstellt, es ist nur eine einfache Methode, um Daten zu erhalten.

Es ist keine gute Idee, gemeinsam genutzte Voreinstellungen zum Speichern kritischer Informationen wie Benutzeranmeldeinformationen zu verwenden. Um Benutzeranmeldeinformationen (z. B. Kennwörter) zu speichern, müssen Sie andere Methoden verwenden, beispielsweise den `AccountManager` Android.

Examples

Sichern einer gemeinsamen Einstellung

Einfacher Codec

Um das Funktionsprinzip zu veranschaulichen, können wir die einfache Ver- und Entschlüsselung wie folgt verwenden.

```
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));
}
```

Implementierungstechnik

```
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);
```

Sichere SharedPreferences online lesen: https://riptutorial.com/de/android/topic/9890/sichere-sharedpreferences

Kapitel 203: Sicherheit

Examples

Überprüfen der App-Signatur - Manipulationserkennung

Diese Technik beschreibt, wie Sie sicherstellen können, dass Ihr .apk mit Ihrem Entwicklerzertifikat signiert wurde, und nutzt die Tatsache, dass das Zertifikat konsistent bleibt und nur Sie Zugriff darauf haben. Wir können diese Technik in 3 einfache Schritte aufteilen:

- Finden Sie die Signatur Ihres Entwicklerzertifikats.
- Betten Sie Ihre Signatur in eine String-Konstante in Ihre App ein.
- Stellen Sie sicher, dass die Signatur zur Laufzeit unserer Embedded Developer-Signatur entspricht.

Hier ist das Code-Snippet:

```
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;

}
```

Sicherheit online lesen: https://riptutorial.com/de/android/topic/4664/sicherheit

Kapitel 204: Singleton-Klasse für Toast-Nachricht erstellen

Einführung

Toastmeldungen sind die einfachste Art, dem Benutzer eine Rückmeldung zu geben. Standardmäßig bietet Android eine graue Nachrichtentaste, in der Sie die Nachricht und die Dauer der Nachricht einstellen können. Wenn Sie eine anpassbarere und wiederverwendbare Toastnachricht erstellen müssen, können Sie sie mithilfe eines benutzerdefinierten Layouts selbst implementieren. Noch wichtiger ist bei der Implementierung, dass die Verwendung des Singelton-Entwurfsmusters die Wartung und Entwicklung der benutzerdefinierten Toast-Nachrichtenklasse vereinfacht.

Syntax

- Toast Toast (Kontext-Kontext)
- void setDuration (int duration)
- void setGravity (int Schwerkraft, int xOffset, int yOffset)
- SetView void (View View)
- void show ()

Parameter

Parameter	Einzelheiten
Kontext	Relevanter Kontext, der Ihre Toastnachricht anzeigen muss. Wenn Sie dies in der Aktivität verwenden, übergeben Sie "dieses" Schlüsselwort oder wenn Sie in fragement als "getActivity ()" verwenden.
Aussicht	Erstellen Sie eine benutzerdefinierte Ansicht, und übergeben Sie das Ansichtsobjekt an diese.
Schwere	Übergeben Sie die Schwerkraftposition des Toasters. Alle Positionen wurden unter der Schwerkraftklasse als statische Variablen hinzugefügt. Die häufigsten Positionen sind Gravity.TOP, Gravity.BOTTOM, Gravity.LEFT, Gravity.RIGHT.
xOffset	Horizontaler Versatz der Toastnachricht.
yOffset	Vertikaler Versatz der Toastnachricht.
Dauer	Dauer der Toastshow. Wir können entweder Toast.LENGTH_SHORT oder Toast.LENGTH_LONG einstellen

Bemerkungen

Die Toast-Nachricht ist eine einfache Möglichkeit, dem Benutzer eine Rückmeldung zu geben, wenn etwas passiert. Wenn Sie eine fortgeschrittenere Methode zur Rückmeldung benötigen, können Sie Dialoge oder eine Snackbar verwenden.

Weitere Informationen zur Toastnachricht finden Sie in dieser Dokumentation.
https://developer.android.com/reference/android/widget/Toast.html

Examples

Erstellen Sie eine eigene Singleton-Klasse für Toastmassagen

So erstellen Sie Ihre eigene Singleton-Klasse für Toast-Nachrichten: Wenn Ihre Anwendung Erfolg, Warnung und die Gefahrennachrichten für verschiedene Anwendungsfälle anzeigen muss, können Sie diese Klasse verwenden, nachdem Sie sie an Ihre eigenen Spezifikationen angepasst haben.

```
    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();
        }
    }
```

Singleton-Klasse für Toast-Nachricht erstellen online lesen:
https://riptutorial.com/de/android/topic/10843/singleton-klasse-fur-toast-nachricht-erstellen

Kapitel 205: So bewahren Sie Passwörter sicher auf

Examples

Verwendung von AES für gesalzene Kennwortverschlüsselung

In diesem Beispiel wird der AES-Algorithmus zum Verschlüsseln von Kennwörtern verwendet. Die Salzlänge kann bis zu 128 Bit betragen.

Wir verwenden die `SecureRandom` Klasse zum Generieren eines Salt, das mit dem Kennwort kombiniert wird, um einen geheimen Schlüssel zu generieren. Die verwendeten Klassen sind bereits in den Android-Paketen `javax.crypto` und `java.security` .

Nachdem ein Schlüssel generiert wurde, müssen wir ihn in einer Variablen beibehalten oder speichern. Wir speichern es unter den gemeinsamen `S_KEY` im Wert `S_KEY` . Anschließend wird ein Kennwort mit der Methode `doFinal` der `Cipher` Klasse verschlüsselt, sobald es in `ENCRYPT_MODE` initialisiert `ENCRYPT_MODE` . Als Nächstes wird das verschlüsselte Kennwort von einem Byte-Array in eine Zeichenfolge konvertiert und in den gemeinsam genutzten Einstellungen gespeichert. Der zum Generieren eines verschlüsselten Passworts verwendete Schlüssel kann verwendet werden, um das Passwort auf ähnliche Weise zu entschlüsseln:

```
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;
```

```
    }
}
```

So bewahren Sie Passwörter sicher auf online lesen:
https://riptutorial.com/de/android/topic/9093/so-bewahren-sie-passworter-sicher-auf

Kapitel 206: Sofortiges Ausführen in Android Studio

Bemerkungen

Instant Run ist ein erweitertes Verhalten für die Befehle run und debug, das ein schnelleres Debugging ermöglicht, da kein kompletter Build und keine erneute Installation für alle im Code der App vorgenommenen Änderungen erforderlich sind.

> Instant Run wurde in Android Studio 2.0 eingeführt und ist ein Verhalten für die Befehle Ausführen und Debuggen, das die Zeit zwischen Aktualisierungen Ihrer App erheblich reduziert. Der erste Build kann zwar länger dauern, aber Instant Run gibt nachfolgende Aktualisierungen an Ihre App weiter, ohne einen neuen APK zu erstellen, sodass Änderungen viel schneller sichtbar werden.
>
> Instant Run wird nur unterstützt, wenn Sie die Debug-Build-Variante bereitstellen, Android Plugin für Gradle Version 2.0.0 oder höher verwenden und minSdkVersion in der Build-Gradle-Datei auf Modulebene der App auf 15 oder höher festlegen. Setzen Sie für die beste Leistung minSdkVersion auf 21 oder höher.
>
> Nach der Bereitstellung einer App wird ein kleines, gelbes Thunderbolt-Symbol in der Schaltfläche Ausführen (oder in der Debug-Schaltfläche) angezeigt, das darauf hinweist, dass Instant Run beim nächsten Klicken auf die Schaltfläche zum Aktualisieren bereit ist. Anstatt eine neue APK zu erstellen, werden nur diese neuen Änderungen ausgeführt. In einigen Fällen muss die App nicht einmal neu gestartet werden, sondern zeigt sofort die Auswirkungen dieser Codeänderungen.
>
> Instant Run überträgt aktualisierten Code und Ressourcen auf Ihr angeschlossenes Gerät oder Emulator, indem Sie einen Hot-Swap, einen Warm-Swap oder einen Cold-Swap durchführen. Die Art des Tauschvorgangs wird automatisch anhand der von Ihnen vorgenommenen Änderung bestimmt. Das Video oben zeigt interessante Details darüber, wie das alles unter der Haube funktioniert. Eine kurze Zusammenfassung des Verhaltens von Instant Run, wenn Sie bestimmte Codeänderungen auf ein Zielgerät verschieben, finden Sie in der folgenden Tabelle.

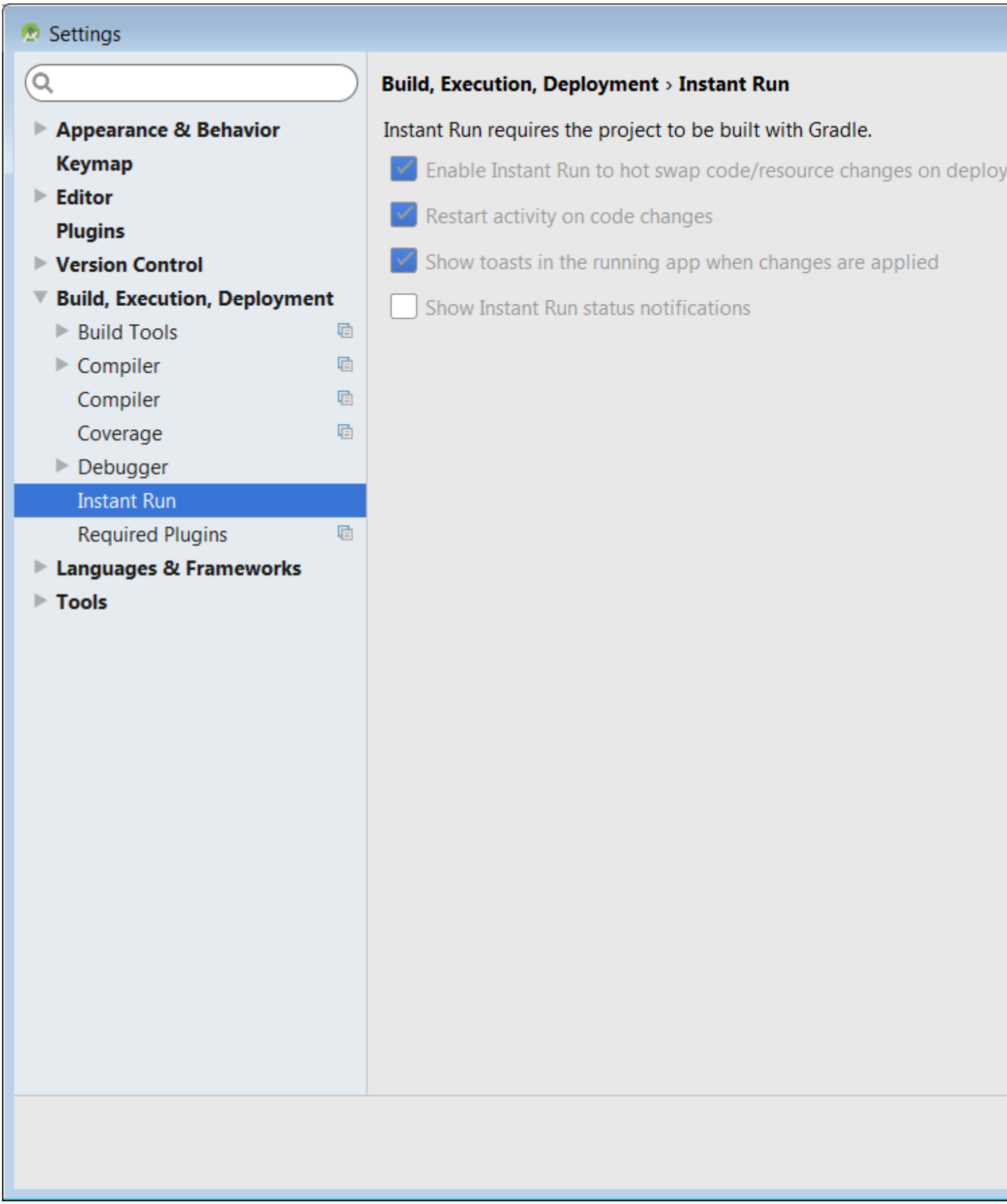
Dokumentation

Examples

Aktivieren oder deaktivieren Sie Instant Run

1. Öffnen Sie das Dialogfeld Einstellungen oder Voreinstellungen:
   - Unter Windows oder Linux wählen Sie im Hauptmenü `File` > `Settings` .
   - Wählen Sie unter Mac OSX im Hauptmenü `Android Studio` > `Preferences` aus.

2. Navigieren Sie zu `Build, Execution, Deployment` > `Compiler` .
3. Geben Sie in das Textfeld neben Befehlszeilenoptionen Ihre Befehlszeilenoptionen ein.
4. Klicken Sie auf OK, um zu speichern und zu beenden.



Die oberste Option ist Instant Run. Aktivieren / Deaktivieren Sie dieses Kontrollkästchen.

Dokumentation

Arten von Code-Swaps in Instant Run

Mit Instant Run können Sie drei Arten von Code-Swaps verwenden, um das schnellere Debuggen und Ausführen von Apps in Ihrem Code in Android Studio zu unterstützen.

- Heißer Tausch
- Warm Swap
- Cold Swap

Wann werden die einzelnen Swaps ausgelöst?

HOT SWAP wird ausgelöst, wenn die Implementierung einer vorhandenen Methode geändert wird.

WARM SWAP wird ausgelöst, wenn eine vorhandene Ressource geändert oder entfernt wird (alles im Ordner res)

COLD SWAP bei **jeder** Änderung des Strukturcodes im Code Ihrer App, z

1. Hinzufügen, Entfernen oder Ändern:

- eine Anmerkung
- ein Instanzfeld
- ein statisches Feld
- eine statische Methodensignatur
- eine Instanzmethoden-Signatur

2. Ändern Sie, von welcher übergeordneten Klasse die aktuelle Klasse erbt
3. Ändern Sie die Liste der implementierten Schnittstellen
4. Ändern Sie den statischen Initialisierer einer Klasse
5. Layoutelemente, die dynamische Ressourcen-IDs verwenden, neu anordnen

Was passiert, wenn ein Code ausgetauscht wird?

HOT SWAP- Änderungen sind sofort sichtbar - sobald der nächste Aufruf der Methode erfolgt, deren Implementierung geändert wird.

WARM SWAP startet die aktuelle Aktivität neu

COLD SWAP startet die gesamte App neu (ohne Neuinstallation)

Nicht unterstützter Code ändert sich bei Verwendung von Instant Run

Es gibt ein paar Änderungen, bei denen Instant keinen Trick zeigt, und eine vollständige Erstellung und Neuinstallation Ihrer App wird genau so erfolgen, wie dies vor der Geburt von Instant Run der Fall war.

1. Ändern Sie das App-Manifest
2. Ändern Sie die vom App-Manifest referenzierten Ressourcen
3. Ändern eines Android-Widget-Benutzeroberflächenelements (erfordert ein Bereinigen und Wiederholen)

Dokumentation

Sofortiges Ausführen in Android Studio online lesen:
https://riptutorial.com/de/android/topic/2119/sofortiges-ausfuhren-in-android-studio

Kapitel 207: SpannableString

Syntax

- `char charAt (int i)`
- `boolean equals (Object o)`
- `void getChars (int start, int end, char[] dest, int off)`
- `int getSpanEnd (Object what)`
- `int getSpanFlags (Object what)`
- `int getSpanStart (Object what)`
- `T[] getSpans (int queryStart, int queryEnd, Class<T> kind)`
- `int hashCode ()`
- `int length ()`
- `int nextSpanTransition (int start, int limit, Class kind)`
- void removeSpan (Objekt was)
- `void setSpan (Object what, int start, int end, int flags)`
- `CharSequence subSequence (int start, int end)`
- `String toString ()`
- `SpannableString valueOf (CharSequence source)`

Examples

Hinzufügen von Stilen zu einer Textansicht

Im folgenden Beispiel erstellen wir eine Aktivität, um eine einzelne TextView anzuzeigen.

Die TextView verwendet einen `SpannableString` als Inhalt, der einige der verfügbaren Stile veranschaulicht.

Hier 'was wir mit dem Text machen werden:

- Mach es größer
- Fett gedruckt
- Unterstreichen
- Kursiv
- Durchgestrichen
- Farbig
- Hervorgehoben
- Als hochgestellt anzeigen
- Als tiefgestellt anzeigen
- Als Link anzeigen
- Machen Sie es anklickbar.

```
@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);

}
```

Und das Ergebnis wird so aussehen:

Mehrfachschnur mit Mehrfarben

Methode: **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;
}
```

Verwendungszweck:

```
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`
```

SpannableString online lesen: https://riptutorial.com/de/android/topic/10553/spannablestring

Kapitel 208: Speicherlecks

Examples

Häufige Speicherlecks und deren Behebung

1. Korrigieren Sie Ihre Kontexte:

Versuchen Sie es mit dem entsprechenden Kontext: Zum Beispiel, da ein Toast in vielen Aktivitäten statt in nur einer `getApplicationContext()` , verwenden Sie `getApplicationContext()` für Toast. Da Dienste weiterhin ausgeführt werden können, obwohl eine Aktivität beendet ist, starten Sie einen Dienst mit:

```
Intent myService = new Intent(getApplicationContext(), MyService.class);
```

Verwenden Sie diese Tabelle als Kurzanleitung für den geeigneten Kontext:

	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	

Originalartikel zum Kontext hier .

2. Statischer Verweis auf Kontext

Ein schwerwiegender Speicherleckfehler besteht darin, einen statischen Verweis auf `View` behalten. Jede `View` hat einen inneren Bezug zum `Context` . Dies bedeutet, dass eine alte Aktivität mit ihrer gesamten Ansichtshierarchie erst dann gesammelt wird, wenn die App beendet wird. Sie haben Ihre App zweimal im Speicher, wenn Sie den Bildschirm drehen.

Stellen Sie sicher, dass es absolut keine statischen Verweise auf Ansicht, Kontext oder deren Nachkommen gibt.

3. Prüfen Sie, ob Sie Ihre Dienste tatsächlich abschließen.

Ich habe zum Beispiel einen intentService, der die Google Location Service-API verwendet. Und ich habe vergessen, `googleApiClient.disconnect();` :

```
//Disconnect from API onDestroy()
if (googleApiClient.isConnected()) {
    LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient,
GoogleLocationService.this);
    googleApiClient.disconnect();
}
```

4. Überprüfen Sie die Verwendung von Bildern und Bitmaps:

Wenn Sie die **Picasso** - **Bibliothek** von Square verwenden, habe ich festgestellt, dass mir Speicherplatz verloren ging, weil ich nicht `.fit()` verwendet habe. `.fit()` mein Speicherbedarf von 50 MB im Durchschnitt drastisch auf weniger als 19 MB reduziert.

```
Picasso.with(ActivityExample.this)                    //Activity context
                .load(object.getImageUrl())
                .fit()                                 //This avoided the OutOfMemoryError
                .centerCrop()                          //makes image to not stretch
                .into(imageView);
```

5. Wenn Sie Broadcast-Receiver verwenden, heben Sie die Registrierung auf.

6. Wenn Sie `java.util.Observer` (Observer-Muster) verwenden:

`deleteObserver(observer);` Sie sicher, dass Sie `deleteObserver(observer);`

Vermeiden Sie undichte Aktivitäten mit AsyncTask

> **Ein Wort der Vorsicht** : Bei AsyncTask gibt es neben dem hier beschriebenen Speicherverlust viele Gotchas. Seien Sie also vorsichtig mit dieser API oder vermeiden Sie sie ganz, wenn Sie die Auswirkungen nicht vollständig verstehen. Es gibt viele Alternativen (Thread, EventBus, RxAndroid usw.).

Ein häufiger Fehler mit `AsyncTask` ist eine starke Bezugnahme auf den Host zu erfassen `Activity` (oder `Fragment`):

```
class MyActivity extends Activity {
  private AsyncTask<Void, Void, Void> myTask = new AsyncTask<Void, Void, Void>() {
    // Don't do this! Inner classes implicitly keep a pointer to their
    // parent, which in this case is the Activity!
  }
}
```

Dies ist ein Problem, da `AsyncTask` die übergeordnete `Activity` leicht `AsyncTask` kann, z. B. wenn während der Task eine Konfigurationsänderung vorgenommen wird.

Der richtige Weg, dies zu tun, ist, Ihre Aufgabe zu einer `static` Klasse zu machen, die das übergeordnete Element nicht erfasst und einen schwachen Verweis auf die `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
    }
  }
}
```

Anonymer Rückruf bei Aktivitäten

Jedes Mal, wenn Sie eine anonyme Klasse erstellen, behält sie einen impliziten Verweis auf die übergeordnete Klasse bei. Also wenn du schreibst:

```
public class LeakyActivity extends Activity
{

...

    foo.registerCallback(new BarCallback()
    {
        @Override
        public void onBar()
        {
            // do something
        }
    });
}
```

Sie senden tatsächlich einen Verweis auf Ihre LeakyActivity-Instanz an foo. Wenn der Benutzer von Ihrer LeakyActivity weg navigiert, kann diese Referenz verhindern, dass die LeakyActivity-Instanz Müll gesammelt wird. Dies ist ein schwerwiegendes Leck, da Aktivitäten auf ihre gesamte Ansichtshierarchie verweisen und daher recht große Objekte im Speicher sind.

So vermeiden Sie dieses Leck:

Natürlich können Sie anonyme Rückrufe in Aktivitäten vollständig vermeiden. Sie können auch alle Ihre Rückrufe in Bezug auf den Aktivitätslebenszyklus aufheben. wie so:

```
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);
    }
}
```

Aktivitätskontext in statischen Klassen

Häufig möchten Sie einige Android-Klassen in einfachere Dienstprogrammklassen einbetten. Diese Dienstprogrammklassen benötigen häufig einen Kontext, um auf das Android-Betriebssystem oder die Ressourcen Ihrer Apps zuzugreifen. Ein typisches Beispiel hierfür ist ein Wrapper für die SharedPreferences-Klasse. Um auf die gemeinsamen Einstellungen von Androids zuzugreifen, muss man schreiben:

```
context.getSharedPreferences(prefsName, mode);
```

Und so kann man versucht sein, die folgende Klasse zu erstellen:

```
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);
    }
}
```

Wenn Sie jetzt `init()` mit Ihrem Aktivitätskontext aufrufen, behält der LeakySharedPrefsWrapper einen Verweis auf Ihre Aktivität bei und verhindert, dass Müll gesammelt wird.

Wie zu vermeiden:

Wenn Sie statische `context.getApplicationContext();` aufrufen, können Sie den Kontext der Anwendung mithilfe von `context.getApplicationContext();` senden `context.getApplicationContext();`

Beim Erstellen statischer Hilfsfunktionen können Sie den Anwendungskontext aus dem Kontext extrahieren, den Sie erhalten (Aufruf von getApplicationContext () im Anwendungskontext gibt den Anwendungskontext zurück). Die Korrektur für unseren Wrapper ist also einfach:

```
public static void init(Context context)
{
    sContext = context.getApplicationContext();
}
```

Wenn der Anwendungskontext nicht für Ihren Anwendungsfall geeignet ist, können Sie in jede Dienstprogrammfunktion einen Context-Parameter einschließen. Vermeiden Sie es, Verweise auf diese Kontextparameter beizubehalten. In diesem Fall würde die Lösung so aussehen:

```
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);
}
```

Erkennen Sie Speicherverluste mit der LeakCanary-Bibliothek

LeakCanary ist eine Open Source Java-Bibliothek zur Erkennung von Speicherverlusten in Ihren Debug-Builds.

`build.gradle` einfach die Abhängigkeiten im `build.gradle` :

```
 dependencies {
   debugCompile 'com.squareup.leakcanary:leakcanary-android:1.5.1'
   releaseCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
   testCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
 }
```

Dann in Ihrer `Application` Klasse:

```
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);
  }
}
```

Jetzt zeigt LeakCanary automatisch eine Benachrichtigung an, wenn in Ihrem **Debug-** Build ein Aktivitätsspeicherleck entdeckt wird.

HINWEIS: Der Versionscode enthält keinen anderen Verweis auf LeakCanary als die beiden leeren Klassen, die in der `leakcanary-android-no-op` sind.

Vermeiden Sie undichte Aktivitäten mit Zuhörern

Wenn Sie einen Listener in einer Aktivität implementieren oder erstellen, achten Sie immer auf den Lebenszyklus des Objekts, für das der Listener registriert ist.

Stellen Sie sich eine Anwendung vor, bei der wir verschiedene Aktivitäten / Fragmente haben, die daran interessiert sind, wann ein Benutzer an- oder abgemeldet ist. Eine Möglichkeit, dies zu tun, besteht darin, über eine Einzelinstanz eines `UserController` zu verfügen, die abonniert werden kann, um benachrichtigt zu werden, wenn sich der Status des Benutzers ändert:

```
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();
    }
}
```

Dann gibt es zwei Aktivitäten, `SignInActivity` :

```
public class SignInActivity extends Activity implements UserController.StateListener{

    UserController userController;

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

        this.userController = UserController.getInstance();
        this.userController.registerUserStateChangeListener(this);
    }

    @Override
    public void userLoggedIn() {
        startMainActivity();
    }

    @Override
    public void userLoggedOut() {
        showLoginForm();
    }

    ...

    public void onLoginClicked(View v) {
        userController.login();
    }
}
```

Und `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();
    }
}
```

In diesem Beispiel passiert `MainActivity` : Jedes Mal, wenn sich der Benutzer `MainActivity` und abmeldet, tritt eine `MainActivity` Instanz aus. Das Leck tritt auf, weil es einen Verweis auf die Aktivität in `UserController#listeners` .

Bitte beachten Sie: Selbst wenn wir eine anonyme innere Klasse als Zuhörer verwenden, leckt die Aktivität immer noch:

```
...
this.userController.registerUserStateChangeListener(new UserController.StateListener() {
    @Override
    public void userLoggedIn() {
        showUserAccount();
    }

    @Override
    public void userLoggedOut() {
        finish();
    }
});
...
```

Die Aktivität würde immer noch auslaufen, da die anonyme innere Klasse implizit auf die äußere Klasse verweist (in diesem Fall die Aktivität). Aus diesem Grund ist es möglich, Instanzmethoden aus der inneren Klasse in der äußeren Klasse aufzurufen. Tatsächlich sind die einzigen inneren Klassen, die keinen Bezug zur äußeren Klasse haben, **statische** innere Klassen.

Kurz gesagt, alle Instanzen nicht statischer innerer Klassen enthalten einen impliziten Verweis auf die Instanz der äußeren Klasse, die sie erstellt hat.

Es gibt zwei Hauptansätze, um dieses `WeakReference` zu lösen: entweder durch Hinzufügen einer Methode zum Entfernen eines Listeners aus `UserController#listeners` oder durch Verwendung einer `WeakReference` , um die Referenz der Listener zu `WeakReference` .

Alternative 1: Zuhörer entfernen

Beginnen wir mit dem Erstellen einer neuen Methode `removeUserStateChangeListener(StateListener listener)` :

```
public class UserController {
```

```
    ...

    public void registerUserStateChangeListener(StateListener listener) {
        listeners.add(listener);
    }

    public void removeUserStateChangeListener(StateListener listener) {
        listeners.remove(listener);
    }

    ...
}
```

Dann rufen wir diese Methode in der `onDestroy` Methode der Aktivität auf:

```
public class MainActivity extends Activity implements UserController.StateListener{
    ...

    @Override
    protected void onDestroy() {
        super.onDestroy();
        userController.removeUserStateChangeListener(this);
    }
}
```

Durch diese Änderung werden die Instanzen von `MainActivity` nicht mehr durchgesickert, wenn sich der Benutzer `MainActivity` und abmeldet. Wenn die Dokumentation jedoch nicht eindeutig ist, besteht die Gefahr, dass der nächste Entwickler, der `UserController` möglicherweise nicht mehr weiß, dass es erforderlich ist, die Registrierung des Listeners aufzuheben, wenn die Aktivität zerstört wird. Dies führt uns zur zweiten Methode, um diese Art von Lecks zu vermeiden.

Alternative 2: Verwendung schwacher Referenzen

Lassen Sie uns zunächst erklären, was eine schwache Referenz ist. Eine schwache Referenz enthält, wie der Name schon sagt, eine schwache Referenz auf ein Objekt. Verglichen mit einem normalen Instanzfeld, das eine starke Referenz ist, hindern schwache Referenzen den Garbage Collector (GC) nicht daran, die Objekte zu entfernen. In dem obigen Beispiel würde dies ermöglichen, dass `MainActivity` nach der `MainActivity` von `MainActivity` gesammelt wird, wenn der `UserController` `WeakReference` für die Referenz der Listener verwendet.

Kurz gesagt, eine schwache Referenz sagt dem GC, dass, wenn niemand anderes eine starke Referenz auf dieses Objekt hat, es fortfahren und es entfernen kann.

Lassen Sie uns `UserController` ändern, dass eine Liste von `WeakReference` , um die Listener zu `WeakReference` :

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

Bei dieser Änderung spielt es keine Rolle, ob die Listener entfernt werden oder nicht, da `UserController` keine starken Verweise auf die Listener enthält. Es ist jedoch umständlich, diesen Boilerplate-Code jedes Mal zu schreiben. Erstellen wir daher eine generische Klasse namens `WeakCollection` :

```
public class WeakCollection<T> {
    private LinkedList<WeakReference<T>> list;

    public WeakCollection() {
        this.list = new LinkedList<>();
    }
    public void put(T item){
        //Make sure that we don't re add an item if we already have the reference.
        List<T> currentList = get();
        for(T oldItem : currentList){
            if(item == oldItem){
                return;
```

```
            }
        }
        list.add(new WeakReference<T>(item));
    }

    public List<T> get() {
        List<T> ret = new ArrayList<>(list.size());
        List<WeakReference<T>> itemsToRemove = new LinkedList<>();
        for (WeakReference<T> ref : list) {
            T item = ref.get();
            if (item == null) {
                itemsToRemove.add(ref);
            } else {
                ret.add(item);
            }
        }
        for (WeakReference ref : itemsToRemove) {
            this.list.remove(ref);
        }
        return ret;
    }

    public void remove(T listener) {
        WeakReference<T> refToRemove = null;
        for (WeakReference<T> ref : list) {
            T item = ref.get();
            if (item == listener) {
                refToRemove = ref;
            }
        }
        if(refToRemove != null){
            list.remove(refToRemove);
        }
    }
}
```

Lassen Sie uns `UserController` jetzt erneut schreiben, um stattdessen `WeakCollection<T>` verwenden:

```
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();
        }
    }

    ...
}
```

Wie im obigen `WeakCollection<T>` gezeigt, entfernt die `WeakCollection<T>` den gesamten Boilerplate-Code, der zur Verwendung von `WeakReference` anstelle einer normalen Liste erforderlich ist. Um das `UserController#removeUserStateChangeListener(StateListener)` : Wenn ein Aufruf an `UserController#removeUserStateChangeListener(StateListener)` verpasst wird, gehen der Listener und alle darauf verweisenden Objekte nicht verloren.

Vermeiden Sie Speicherverluste mit Anonymous Class, Handler, Timer Task, Thread

In Android verwendet jeder Entwickler mindestens einmal in einem Projekt die `Anonymous Class` (lauffähig). Jede `Anonymous Class` hat einen Verweis auf ihr übergeordnetes Element (Aktivität). Wenn Sie eine lang andauernde Aufgabe ausführen, wird die übergeordnete Aktivität erst zerstört, wenn die Aufgabe beendet ist.
Beispiel verwendet den Handler und die anonyme `Runnable` Klasse. Der Speicher ist undicht, wenn wir die Aktivität beenden, bevor `Runnable` beendet ist.

```
  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...
```

Wie lösen wir es?

1. Machen Sie keine langen Operationen mit `Anonymous Class` oder wir benötigen eine `Static class` und übergeben Sie `WeakReference` (wie Aktivität, Ansicht ...). `Thread` ist mit der `Anonymous Class` identisch.
2. Brechen Sie den `Handler` , `Timer` wenn die Aktivität zerstört wird.

Speicherlecks online lesen: https://riptutorial.com/de/android/topic/2687/speicherlecks

Kapitel 209: Speichern von Dateien im internen und externen Speicher

Syntax

- FileOutputStream openFileInput (Name der Zeichenfolge)
- FileOutputStream openFileOutput (Stringname, int-Modus)
- Datei (Dateiname, Stringname)
- Datei (String-Pfad)
- Datei getExternalStoragePublicDirectory (String-Typ)
- Datei getExternalFilesDir (String-Typ)

Parameter

Parameter	Einzelheiten
Name	Der Name der zu öffnenden Datei. HINWEIS: Kann keine Pfadtrennzeichen enthalten
Modus	Betriebsart. Verwenden Sie `MODE_PRIVATE` für die Standardoperation und `MODE_APPEND` , um eine vorhandene Datei `MODE_APPEND` . Andere Modi umfassen `MODE_WORLD_READABLE` und `MODE_WORLD_WRITEABLE` , die beide in API 17 nicht mehr unterstützt werden.
dir	Verzeichnis der Datei, in der eine neue Datei erstellt werden soll
Pfad	Pfad zum Angeben des Speicherorts der neuen Datei
Art	Typ des abzurufenden Dateiverzeichnisses. Kann `null` oder eine der folgenden sein: `DIRECTORY_MUSIC` , `DIRECTORY_PODCASTS` , `DIRECTORY_RINGTONES` , `DIRECTORY_ALARMS` , `DIRECTORY_NOTIFICATIONS` , `DIRECTORY_PICTURES` oder `DIRECTORY_MOVIES`

Examples

Internen Speicher verwenden

Standardmäßig sind alle Dateien, die Sie im internen Speicher speichern, für Ihre Anwendung privat. Auf sie kann weder von anderen Anwendungen noch vom Benutzer unter normalen Umständen zugegriffen werden. **Diese Dateien werden gelöscht, wenn der Benutzer die Anwendung deinstalliert** .

So schreiben Sie Text in eine Datei

```
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();
}
```

So hängen Sie Text an eine vorhandene Datei an

Verwenden Sie `Context.MODE_APPEND` für den `openFileOutput` von `openFileOutput`

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

Externen Speicher verwenden

"Externer" Speicher ist eine andere Art von Speicher, mit dem Dateien auf dem Gerät des Benutzers gespeichert werden können. Es gibt einige wesentliche Unterschiede zum "internen" Speicher, nämlich:

- Es ist nicht immer verfügbar. Bei einem Wechselmedium (SD-Karte) kann der Benutzer den Speicher einfach entfernen.
- Es ist nicht privat. Der Benutzer (und andere Anwendungen) haben Zugriff auf diese Dateien.
- Wenn der Benutzer die App deinstalliert, werden die Dateien, die Sie in dem mit `getExternalFilesDir()` abgerufenen Verzeichnis `getExternalFilesDir()` , entfernt.

Um externen Speicher verwenden zu können, müssen wir zunächst die richtigen Berechtigungen erwerben. Sie müssen verwenden:

- `android.permission.WRITE_EXTERNAL_STORAGE` zum Lesen und Schreiben
- `android.permission.READ_EXTERNAL_STORAGE` zum Lesen

Um diese Berechtigungen zu erteilen, müssen Sie sie in Ihrer `AndroidManifest.xml` als solche identifizieren

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

> HINWEIS: Da es sich um gefährliche Berechtigungen handelt, wenn Sie **API-Level 23** oder höher verwenden, müssen Sie die **Berechtigungen zur Laufzeit** anfordern.

Bevor Sie versuchen, aus einem externen Speicher zu schreiben oder zu lesen, sollten Sie immer überprüfen, ob das Speichermedium verfügbar ist.

```
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
}
```

Beim Schreiben von Dateien auf den externen Speicher sollten Sie entscheiden, ob die Datei als öffentlich oder privat erkannt werden soll. Während diese beiden Dateitypen für den Benutzer und andere Anwendungen auf dem Gerät weiterhin verfügbar sind, besteht ein wesentlicher Unterschied zwischen ihnen.

Öffentliche Dateien sollten auf dem Gerät verbleiben, wenn der Benutzer die App deinstalliert. Ein Beispiel für eine Datei, die als öffentlich gespeichert werden soll, sind Fotos, die von Ihrer Anwendung aufgenommen werden.

Private Dateien sollten alle entfernt werden, wenn der Benutzer die App deinstalliert. Diese Dateitypen wären app-spezifisch und für den Benutzer oder andere Anwendungen nicht von Nutzen. Ex. temporäre Dateien, die von Ihrer Anwendung heruntergeladen / verwendet werden.

So erhalten Sie Zugriff auf das Verzeichnis `Documents` für öffentliche und private Dateien.

Öffentlichkeit

```
// 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();
```

Privatgelände

```
// 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: Interner und externer Speicher - Erläuterung der Terminologie

Android-Entwickler (hauptsächlich Anfänger) wurden hinsichtlich der Terminologie für interne und externe Speicherung verwirrt. Es gibt viele Fragen zu Stackoverflow, die das gleiche betreffen. Dies ist hauptsächlich darauf zurückzuführen, dass sich die *Terminologie* gemäß der offiziellen Dokumentation von Google / Android von der eines normalen Android-Betriebssystems unterscheidet. Daher dachte ich, die Dokumentation würde helfen.

Was wir denken - Benutzerterminologie (UT)

Interner Speicher (UT)	Externer Speicher (UT)
Interner Speicher des	herausnehmbare Secure Digital (SD) -Karte oder Micro-SD-

Interner Speicher (UT)	Externer Speicher (UT)
Telefons	Speicher
Beispiel: 32 GB interner Speicher des Nexus 6P.	**Beispiel:** Speicherplatz in herausnehmbaren SD-Karten, die von Anbietern wie Samsung, Sandisk, Strontium, Transcend usw. bereitgestellt werden

Laut Android Dokumentation / Anleitung - Googles Terminologie (GT)

Interner Speicher (GT):

> Standardmäßig sind im internen Speicher gespeicherte Dateien für Ihre Anwendung privat und andere Anwendungen können nicht auf sie zugreifen (auch nicht auf den Benutzer).

Externer Speicher (GT):

> Dies kann ein Wechseldatenträger (z. B. eine SD-Karte) oder ein interner (nicht entfernbarer) Speicher sein.

Externe Speicher (GT) können in zwei Typen eingeteilt werden:

Primärer externer Speicher	Sekundärer externer Speicher oder Wechselspeicher (GT)
Dies entspricht dem internen Speicher des Telefons (oder) des internen Speichers (UT).	Dies ist dasselbe wie ein entfernbarer Micro-SD-Kartenspeicher (oder) Externer Speicher (UT)
Beispiel: 32 GB interner Speicher des Nexus 6P.	**Beispiel:** Speicherplatz in herausnehmbaren SD-Karten, die von Anbietern wie Samsung, Sandisk, Strontium, Transcend usw. bereitgestellt werden
Auf diese Art von Speicher kann von Windows-PCs aus zugegriffen werden, indem Sie Ihr Telefon über ein USB-Kabel an den PC anschließen und in der USB-Optionsbenachrichtigung *Kamera (PTP)* auswählen.	Auf diese Art von Speicher kann von Windows-PCs aus zugegriffen werden, indem Sie Ihr Telefon über ein USB-Kabel an den PC anschließen und in der Benachrichtigung über USB-Optionen *Dateiübertragung* auswählen.

In einer Nussschale,

Externer Speicher (GT) = Interner Speicher (UT) und Externer Speicher (UT)

Wechselspeicher (GT) = Externer Speicher (UT)

Interner Speicher (GT) hat keine Laufzeit in UT.

Lassen Sie mich klar erklären,

Interner Speicher (GT): Standardmäßig sind auf dem internen Speicher gespeicherte Dateien für Ihre Anwendung privat, und andere Anwendungen können nicht auf sie zugreifen. Ihr App-Benutzer kann auch nicht mit dem Dateimanager darauf zugreifen. auch nach Aktivierung der Option "Alle Dateien anzeigen" im Dateimanager. Um auf Dateien im internen Speicher (GT) zugreifen zu können, müssen Sie Ihr Android-Telefon rooten. Wenn der Benutzer Ihre Anwendung deinstalliert, werden diese Dateien außerdem entfernt / gelöscht.

Interner Speicher (GT) ist also **NICHT das,** was wir als internen 32/64 GB-Speicher des Nexus 6P meinen

Im Allgemeinen würde der **Speicherort für den internen Speicher (GT)** lauten:
`/data/data/your.application.package.appname/someDirectory/`

Externer Speicher (GT):

> Jedes Android-kompatible Gerät unterstützt einen freigegebenen "externen Speicher", den Sie zum Speichern von Dateien verwenden können. Auf dem externen Speicher gespeicherte Dateien sind weltweit lesbar und können vom Benutzer geändert werden, wenn USB-Massenspeicher zum Übertragen von Dateien auf einem Computer aktiviert werden.

Speicherort des externen Speichers (GT): Er kann sich *irgendwo* in Ihrem internen Speicher (UT) oder in Ihrem Wechselspeicher (GT) befinden, z. B. einer Micro-SD-Karte. Dies hängt vom OEM Ihres Handys und auch von der Android OS-Version ab.

Um Dateien auf dem externen Speicher (GT) lesen oder schreiben zu können, muss Ihre App die `READ_EXTERNAL_STORAGE` oder `WRITE_EXTERNAL_STORAGE` erwerben.

Zum Beispiel:

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

> Wenn Sie Dateien sowohl lesen als auch schreiben müssen, müssen Sie nur die `WRITE_EXTERNAL_STORAGE` , da dies implizit auch Lesezugriff erfordert.

In **External Storage (GT)** können Sie auch **App-private** Dateien speichern

Aber,

> Wenn der Benutzer Ihre Anwendung deinstalliert, werden dieses Verzeichnis und der gesamte Inhalt gelöscht.

Wann müssen Sie **app-private** Dateien im **externen Speicher (GT)** speichern?

Wenn Sie mit Dateien arbeiten, die nicht für andere Apps vorgesehen sind (z. B. grafische Texturen oder Soundeffekte, die nur Ihre App verwendet), sollten Sie ein privates Speicherverzeichnis auf dem externen Speicher verwenden

Ab Android 4.4 ist für das Lesen oder Schreiben von Dateien in den privaten Verzeichnissen der App nicht die `READ_EXTERNAL_STORAGE` oder `WRITE_EXTERNAL_STORAGE` erforderlich. Sie können also `maxSdkVersion` , dass die Berechtigung nur für die niedrigeren Versionen von Android angefordert werden soll, indem Sie das Attribut `maxSdkVersion` hinzufügen:

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

Methoden zum Speichern im internen Speicher (GT):

Beide Methoden sind in der Context- Klasse vorhanden

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

File getFilesDir ()
```

Methoden zum Speichern im primären externen Speicher, dh internen Speicher (UT):

```
File getExternalStorageDirectory ()

File getExternalFilesDir (String type)

File getExternalStoragePublicDirectory (String type)
```

Am Anfang benutzten alle Environment.getExternalStorageDirectory () , das auf den **Stamm** des **primären externen Speichers** verwies. Daher wurde der primäre externe Speicher mit zufälligem Inhalt gefüllt.

Später wurden diese beiden Methoden hinzugefügt:

1. In der `Context` Klasse fügten sie getExternalFilesDir () hinzu und verweisen auf ein anwendungsspezifisches **Verzeichnis** im primären externen Speicher. Dieses Verzeichnis und sein Inhalt **werden gelöscht,** wenn die App deinstalliert wird.

2. Environment.getExternalStoragePublicDirectory () für zentralisierte Orte zum Speichern bekannter Dateitypen wie Fotos und Filme. Dieses Verzeichnis und sein Inhalt **werden NICHT gelöscht,** wenn die App deinstalliert wird.

Methoden zum Speichern in Wechselmedien (GT), z. B. Micro-SD-Karte

Vor dem **API-Level 19** gab es **keine offizielle Möglichkeit** , auf SD-Karte zu speichern. Aber viele könnten es mit inoffiziellen Bibliotheken oder APIs tun.

Offiziell wurde eine Methode in der `Context` Klasse in API Level 19 (Android-Version 4.4 - Kitkat) eingeführt.

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

> Sie gibt absolute Pfade an anwendungsspezifische Verzeichnisse auf allen freigegebenen / externen Speichergeräten zurück, in denen die Anwendung persistente Dateien ablegen kann, die ihr gehören. Diese Dateien sind in der Anwendung intern und für den Benutzer normalerweise nicht als Medium sichtbar.

Das heißt, es werden Pfade zu **beiden** Arten von externen Speichern (GT) zurückgegeben - internem Speicher und Micro-SD-Karte. Im Allgemeinen wäre der **zweite Pfad** der Speicherpfad der Micro-SD-Karte (jedoch nicht immer). Sie müssen es also überprüfen, indem Sie den Code mit dieser Methode ausführen.

Beispiel mit Code-Snippet:

Ich habe ein neues Android-Projekt mit leeren Aktivitäten erstellt und folgenden Code geschrieben

`protected void onCreate(Bundle savedInstanceState)` -Methode von `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);
```

Nach dem Ausführen des obigen Codes

Ausgabe:

```
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
```

Hinweis: Ich habe mein Telefon an einen Windows-PC angeschlossen. aktiviert beide Entwickleroptionen, USB-Debugging und führte diesen Code aus. Wenn Sie **Ihr Telefon nicht anschließen** ; Wenn Sie dies jedoch auf einem **Android-Emulator** ausführen, kann Ihre Ausgabe variieren. Mein Telefonmodell ist Coolpad Note 3 - läuft unter Android 5.1

Speicherorte auf meinem Handy:

Micro-SD-Speicherort : `/storage/sdcard1`

Interner Speicherort (UT) : `/storage/sdcard0` .

Beachten Sie, dass `/sdcard` & `/storage/emulated/0` `/sdcard` `/storage/emulated/0` auch auf Internal Storage (UT) `/sdcard` . Das sind aber Symlinks zu `/storage/sdcard0` .

Um die verschiedenen Speicherpfade in Android genau zu verstehen, gehen Sie bitte diese Antwort durch

Haftungsausschluss: Alle oben genannten Speicherpfade sind Pfade auf **meinem** Telefon. Ihre Dateien werden möglicherweise **nicht** in denselben Speicherpfaden gespeichert. Da sich die Speicherorte / Pfade auf anderen Mobiltelefonen je nach Hersteller, Hersteller und unterschiedlichen Versionen des Android-Betriebssystems unterscheiden können.

Datenbank auf SD-Karte speichern (Backup DB auf SD)

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

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

```
    if(!dir.exists()) // if there was no folder at this path , it create it .
    {
        dir.mkdirs();
    }

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

Rufen Sie diese Methode folgendermaßen auf:

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

Geräteverzeichnis abrufen:

Fügen Sie zuerst die Speicherberechtigung hinzu, um das Geräteverzeichnis zu lesen / abzurufen.

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

Erstellen Sie eine Modellklasse

```
//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;
    }
}
```

Erstellen Sie eine Liste mithilfe des Verzeichnismodells, um Verzeichnisdaten hinzuzufügen.

```
//define list to show directory

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

Rufen Sie das Verzeichnis mit folgender Methode ab.

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

Verzeichnisliste im Protokoll drucken

```
//print directory list in logs

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

Verwendungszweck

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

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

Um innere Dateien / Ordner eines bestimmten Verzeichnisses abzurufen, verwenden Sie dieselbe Methode. Ändern Sie einfach das Argument, übergeben Sie den aktuell ausgewählten Pfad als Argument und behandeln Sie die Antwort.

So erhalten Sie die Dateierweiterung:

```
private String getExtension(String filename) {

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

    return extension;
}
```

Speichern von Dateien im internen und externen Speicher online lesen: https://riptutorial.com/de/android/topic/150/speichern-von-dateien-im-internen-und-externen-speicher

Kapitel 210: Speisekarte

Syntax

- inflater.inflate (Menü R.menu.your_xml_file, Menü);

Parameter

Parameter	Beschreibung
`inflate(int menuRes, Menu menu)`	Inflation einer Menühierarchie aus der angegebenen XML-Ressource.
`getMenuInflater ()`	Gibt einen `MenuInflater` mit diesem Kontext zurück.
`onCreateOptionsMenu (Menu menu)`	Initialisieren Sie den Inhalt des Standardoptionen-Menüs der Aktivität. Sie sollten Ihre Menüpunkte in das Menü einfügen.
`onOptionsItemSelected (MenuItem item)`	Diese Methode wird aufgerufen, wenn ein Element in Ihrem Optionsmenü ausgewählt wird

Bemerkungen

Um mehr über **Menüs** zu erfahren, lesen Sie dies . Ich hoffe es hilft!

Examples

Optionsmenü mit Trennwänden

In Android gibt es ein Standard-Optionsmenü, das eine Reihe von Optionen annehmen kann. Wenn eine größere Anzahl von Optionen angezeigt werden muss, ist es sinnvoll, diese Optionen zu gruppieren, um die Übersichtlichkeit zu gewährleisten. Optionen können gruppiert werden, indem Trennlinien (dh horizontale Linien) zwischen ihnen platziert werden. Um Trennungen zu ermöglichen, kann das folgende Thema verwendet werden:

```
<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>
```

Durch Ändern des Themas können Teiler zu einem Menü hinzugefügt werden.

Wenden Sie die benutzerdefinierte Schriftart auf das Menü an

```
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);
}
```

und dann in der Aktivität:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    applyFontToMenu(menu,this);
    return true;
}
```

Ein Menü in einer Aktivität erstellen

Um ein eigenes Menü zu definieren, erstellen Sie eine XML-Datei im Verzeichnis `res/menu/` des Projekts und erstellen Sie das Menü mit den folgenden Elementen:

- `<menu>` : Definiert ein Menü, das alle Menüelemente enthält.
- `<item>` : Erstellt ein MenuItem, das ein einzelnes Element in einem Menü darstellt. Wir können auch ein verschachteltes Element erstellen, um ein Untermenü zu erstellen.

Schritt 1:

Erstellen Sie Ihre eigene XML-Datei wie folgt:

In `res/menu/main_menu.xml` :

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/aboutMenu"
```

```
        android:title="About" />
    <item
        android:id="@+id/helpMenu"
        android:title="Help" />
    <item
        android:id="@+id/signOutMenu"
        android:title="Sign Out" />
</menu>
```

Schritt 2:

Um das Optionsmenü anzugeben, überschreiben Sie `onCreateOptionsMenu()` in Ihrer *Aktivität* .

In dieser Methode können Sie Ihre `res/menu/main_menu.xml` aufpumpen (definiert in Ihrer XML-Datei, z. B. `res/menu/main_menu.xml`).

```
 @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }
```

Wenn der Benutzer ein Element aus dem Optionsmenü auswählt, ruft das System *die* überschriebene `onOptionsItemSelected()` Methode Ihrer *Aktivität auf* .

- Diese Methode übergibt das ausgewählte MenuItem.
- Sie können das Element identifizieren, indem Sie `getItemId()` aufrufen, das die eindeutige ID für das `getItemId()` zurückgibt (definiert durch das `android:id attribute` im Menü resource - `res/menu/main_menu.xml`). * /

```
 @Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.aboutMenu:
            Log.d(TAG, "Clicked on About!");
            // Code for About goes here
            return true;
        case R.id.helpMenu:
            Log.d(TAG, "Clicked on Help!");
            // Code for Help goes here
            return true;
        case R.id.signOutMenu:
            Log.d(TAG, "Clicked on Sign Out!");
            // SignOut method call goes here
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Einpacken

Ihr `Activity` sollte wie `Activity` aussehen:

```
public class MainActivity extends AppCompatActivity {

    private static final String TAG = "mytag";

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.aboutMenu:
                Log.d(TAG, "Clicked on About!");
                // Code for About goes here
                return true;
            case R.id.helpMenu:
                Log.d(TAG, "Clicked on Help!");
                // Code for Help goes here
                return true;
            case R.id.signOutMenu:
                Log.d(TAG, "User signed out");
                // SignOut method call goes here
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

Screenshot, wie Ihr eigenes Menü aussieht:

Speisekarte online lesen: https://riptutorial.com/de/android/topic/2028/speisekarte

Kapitel 211: Spinner

Examples

Einen Spinner zu Ihrer Aktivität hinzufügen

In /res/values/strings.xml:

```
<string-array name="spinner_options">
    <item>Option 1</item>
    <item>Option 2</item>
    <item>Option 3</item>
</string-array>
```

Im Layout XML:

```
<Spinner
    android:id="@+id/spinnerName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_options" />
```

In Aktivität:

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

Grundlegendes Spinner-Beispiel

Spinner Es ist eine Art Dropdown-Eingabe. Erstens im Layout

```
  <Spinner
    android:id="@+id/spinner"     <!-- id to refer this spinner from JAVA-->
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

</Spinner>
```

Jetzt zweitens Werte in Spinner auffüllen Es gibt hauptsächlich zwei Möglichkeiten, Werte in `spinner` .

1. Erstellen Sie aus XML selbst ein **Array.xml** im **Werte-** Verzeichnis unter **res** . Erstellen Sie dieses `array`

```
  <string-array name="defaultValue">
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
</string-array>
```

Fügen Sie nun diese Zeile in sppiner XML hinzu

```
            android:entries="@array/defaultValue"
```

2. Sie können auch Werte über JAVA hinzufügen

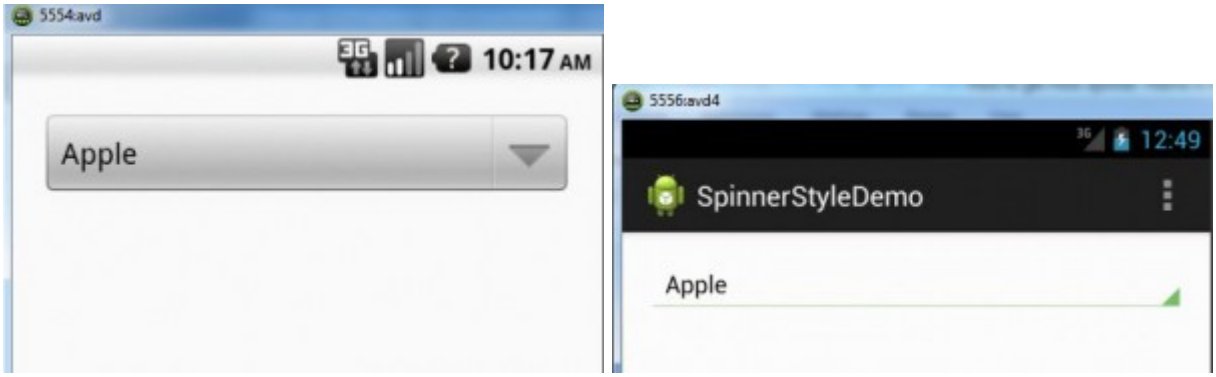
wenn Sie in `activity` cityArea = (Spinner) findViewById (R.id.cityArea) verwenden; sonst, wenn Sie in `fragment`

```
cityArea = (Spinner) findViewById(R.id.cityArea);
```

Erstellen Sie nun eine `arrayList` von `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));
```

Das wird aussehen



Je nach Android-Version wird es gerendert

Im Folgenden sind einige der Standarddesigns aufgeführt

Wenn eine App nicht explizit ein Design in ihrem Manifest anfordert, bestimmt Android System das Standarddesign basierend auf der targetSdkVersion der App, um die ursprünglichen Erwartungen der App zu erfüllen:

Android SDK-Version	Standardthema
Version <11	@android: style / Theme
Version zwischen 11 und 13	@android: style / Theme.Holo

Android SDK-Version	Standardthema
14 und höher	@android: style / Theme.DeviceDefault

`Spinner` kann leicht mit Hilfe von XML angepasst werden, z

```
 android:background="@drawable/spinner_background"

 android:layout_margin="16dp"

 android:padding="16dp"
```

Erstellen Sie einen benutzerdefinierten Hintergrund in XML und verwenden Sie ihn.

leicht die Position und andere Details des ausgewählten Elements im Spinner ermitteln

```
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) {

        }
    });
```

Ändern Sie die Textfarbe des ausgewählten Elements im Drehfeld

Dies kann auf zwei Arten in XML erfolgen

```
<item android:state_activated="true" android:color="@color/red"/>
```

Dadurch wird die Farbe des ausgewählten Elements im Popup geändert.

und von JAVA aus (im setOnItemSelectedListener (...))

```
 @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long
id) {
                 ((TextView) parent.getChildAt(0)).setTextColor(0x00000000);
// similarly change `background color` etc.
            }
```

Spinner online lesen: https://riptutorial.com/de/android/topic/3459/spinner

Kapitel 212: SQLite

Einführung

SQLite ist ein relationales Datenbankverwaltungssystem, das in C geschrieben ist . Um mit der Arbeit mit SQLite-Datenbanken innerhalb des Android-Frameworks zu beginnen, definieren Sie eine Klasse, die SQLiteOpenHelper erweitert , und passen Sie sie bei Bedarf an.

Bemerkungen

Die `SQLiteOpenHelper` Klasse definiert statische `onCreate()` und `onUpgrade()` . Diese Methoden werden in den entsprechenden Methoden einer `SQLiteOpenHelper` Unterklasse `SQLiteOpenHelper` , die Sie mit Ihren eigenen Tabellen anpassen.

Examples

Verwendung der SQLiteOpenHelper-Klasse

```
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;
            }
        }
    }
}
```

Daten in die Datenbank einfügen

```
// 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 () -Methode

`SQLiteOpenHelper` ist eine `SQLiteOpenHelper` zur Verwaltung der Datenbankerstellung und Versionsverwaltung.

In dieser Klasse ist die `onUpgrade()` Methode für das Upgrade der Datenbank verantwortlich, wenn Sie Änderungen am Schema vornehmen. Sie wird aufgerufen, wenn die Datenbankdatei bereits vorhanden ist, die Version jedoch niedriger ist als in der aktuellen Version der App angegeben. Für jede Datenbankversion müssen die von Ihnen vorgenommenen spezifischen Änderungen angewendet werden.

```
@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;
        }
    }
}
```

Daten von einem Cursor lesen

Hier ist ein Beispiel für eine Methode, die in einer Unterklasse `SQLiteOpenHelper` leben würde. Es verwendet den `searchTerm` String zum Filtern der Ergebnisse, durchläuft den Inhalt des Cursors und gibt diesen Inhalt in einer `List` mit `Product` .

Definieren Sie zunächst die `Product` POJO-Klasse , die der Container für jede aus der Datenbank abgerufene Zeile ist:

```
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;
  }
}
```

Definieren Sie dann die Methode, die die Datenbank abfragen soll, und geben Sie eine `List` der `Product` :

```
public List<Product> searchForProducts(String searchTerm) {

    // When reading data one should always just get a readable database.
    final SQLiteDatabase database = this.getReadableDatabase();

    final Cursor cursor = database.query(
            // Name of the table to read from
            TABLE_NAME,

            // String array of the columns which are supposed to be read
            new String[]{COLUMN_NAME, COLUMN_DESCRIPTION, COLUMN_VALUE},

            // The selection argument which specifies which row is read.
            // ? symbols are parameters.
            COLUMN_NAME + " LIKE ?",

            // The actual parameters values for the selection as a String array.
            // ? above take the value from here
            new String[]{"%" + searchTerm + "%"},

            // GroupBy clause. Specify a column name to group similar values
            // in that column together.
            null,

            // Having clause. When using the GroupBy clause this allows you to
            // specify which groups to include.
            null,

            // OrderBy clause. Specify a column name here to order the results
            // according to that column. Optionally append ASC or DESC to specify
            // an ascending or descending order.
            null
    );

    // To increase performance first get the index of each column in the cursor
    final int idIndex = cursor.getColumnIndex(COLUMN_ID);
    final int nameIndex = cursor.getColumnIndex(COLUMN_NAME);
    final int descriptionIndex = cursor.getColumnIndex(COLUMN_DESCRIPTION);
    final int valueIndex = cursor.getColumnIndex(COLUMN_VALUE);

    try {

        // If moveToFirst() returns false then cursor is empty
        if (!cursor.moveToFirst()) {
            return new ArrayList<>();
        }

        final List<Product> products = new ArrayList<>();

        do {
```

```
            // Read the values of a row in the table using the indexes acquired above
            final long id = cursor.getLong(idIndex);
            final String name = cursor.getString(nameIndex);
            final String description = cursor.getString(descriptionIndex);
            final float value = cursor.getFloat(valueIndex);

            products.add(new Product(id, name, description, value));

        } while (cursor.moveToNext());

        return products;

    } finally {
        // Don't forget to close the Cursor once you are done to avoid memory leaks.
        // Using a try/finally like in this example is usually the best way to handle this
        cursor.close();

        // close the database
        database.close();
    }
}
```

Erstellen Sie einen Vertrag, einen Helfer und einen Anbieter für SQLite in Android

DBContract.java

```
//Define the tables and columns of your local database
public final class DBContract {
/*Content Authority its a name for the content provider, is convenient to use the package app
name to be unique on the device */

    public static final String CONTENT_AUTHORITY = "com.yourdomain.yourapp";

    //Use CONTENT_AUTHORITY to create all the database URI's that the app will use to link the
content provider.
    public static final Uri BASE_CONTENT_URI = Uri.parse("content://" + CONTENT_AUTHORITY);

    /*the name of the uri that can be the same as the name of your table.
    this will translate to content://com.yourdomain.yourapp/user/ as a valid URI
    */
    public static final String PATH_USER = "User";

    // To prevent someone from accidentally instantiating the contract class,
    // give it an empty constructor.
    public DBContract () {}

    //Intern class that defines the user table
    public static final class UserEntry implements BaseColumns {
        public static final URI CONTENT_URI =
BASE_CONTENT_URI.buildUpon().appendPath(PATH_USER).build();

        public static final String CONTENT_TYPE =
ContentResolver.CURSOR_DIR_BASE_TYPE+"/"+CONTENT_AUTHORITY+"/"+PATH_USER;

        //Name of the table
        public static final String TABLE_NAME="User";
```

```
        //Columns of the user table
        public static final String COLUMN_Name="Name";
        public static final String COLUMN_Password="Password";

        public static Uri buildUri(long id){
            return ContentUris.withAppendedId(CONTENT_URI,id);
        }
}
```

DBHelper.java

```
public class DBHelper extends SQLiteOpenHelper{

//if you change the schema of the database, you must increment this number
private static final int DATABASE_VERSION=1;
static final String DATABASE_NAME="mydatabase.db";
private static DBHelper mInstance=null;
public static DBHelper getInstance(Context ctx){
    if(mInstance==null){
    mInstance= new DBHelper(ctx.getApplicationContext());
    }
    return mInstance;
}

public DBHelper(Context context){
    super(context,DATABASE_NAME,null,DATABASE_VERSION);
}

public int GetDatabase_Version() {
    return DATABASE_VERSION;
}

@Override
public void onCreate(SQLiteDatabase sqLiteDatabase){
//Create the table users
final String SQL_CREATE_TABLE_USERS="CREATE TABLE "+UserEntry.TABLE_NAME+ " ("+
UserEntry._ID+" INTEGER PRIMARY KEY, "+
UserEntry.COLUMN_Name+" TEXT , "+
UserEntry.COLUMN_Password+" TEXT "+
" ); ";

sqLiteDatabase.execSQL(SQL_CREATE_TABLE_USERS);

}

@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {
    sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + UserEntry.TABLE_NAME);
}

}
```

DBProvider.java

```
public class DBProvider extends ContentProvider {

    private static final UriMatcher sUriMatcher = buildUriMatcher();
    private DBHelper mDBHelper;
```

```
    private Context mContext;

    static final int USER = 100;

    static UriMatcher buildUriMatcher() {

        final UriMatcher matcher = new UriMatcher(UriMatcher.NO_MATCH);
        final String authority = DBContract.CONTENT_AUTHORITY;

        matcher.addURI(authority, DBContract.PATH_USER, USER);

        return matcher;
    }

    @Override
    public boolean onCreate() {
        mDBHelper = new DBHelper(getContext());
        return false;
    }

    public PeaberryProvider(Context context) {
        mDBHelper = DBHelper.getInstance(context);
        mContext = context;
    }

    @Override
    public String getType(Uri uri) {
        // determine what type of Uri is
        final int match = sUriMatcher.match(uri);

        switch (match) {
            case USER:
                return DBContract.UserEntry.CONTENT_TYPE;

            default:
                throw new UnsupportedOperationException("Uri unknown: " + uri);
        }
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String selection, String[]
selectionArgs,
                        String sortOrder) {
        Cursor retCursor;
        try {
            switch (sUriMatcher.match(uri)) {
                case USER: {
                    retCursor = mDBHelper.getReadableDatabase().query(
                            DBContract.UserEntry.TABLE_NAME,
                            projection,
                            selection,
                            selectionArgs,
                            null,
                            null,
                            sortOrder
                    );
                    break;
                }
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
```

```
        } catch (Exception ex) {
            Log.e("Cursor", ex.toString());
        } finally {
            mDBHelper.close();
        }
        return null;
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        final SQLiteDatabase db = mDBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        Uri returnUri;
        try {

            switch (match) {
                case USER: {
                    long _id = db.insert(DBContract.UserEntry.TABLE_NAME, null, values);
                    if (_id > 0)
                        returnUri = DBContract.UserEntry.buildUri(_id);
                    else
                        throw new android.database.SQLException("Error at inserting row in " +
uri);
                    break;
                }
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
            mContext.getContentResolver().notifyChange(uri, null);
            return returnUri;
        } catch (Exception ex) {
            Log.e("Insert", ex.toString());
            db.close();
        } finally {
            db.close();
        }
        return null;
    }

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        final SQLiteDatabase db = DBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        int deletedRows;
        if (null == selection) selection = "1";
        try {
            switch (match) {
                case USER:
                    deletedRows = db.delete(
                            DBContract.UserEntry.TABLE_NAME, selection, selectionArgs);
                    break;
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
            if (deletedRows != 0) {
                mContext.getContentResolver().notifyChange(uri, null);
            }
            return deletedRows;
        } catch (Exception ex) {
            Log.e("Insert", ex.toString());
        } finally {
```

```
            db.close();
        }
        return 0;
    }

    @Override
    public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)
{
        final SQLiteDatabase db = mDBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        int updatedRows;
        try {
            switch (match) {
                case USER:
                    updatedRows = db.update(DBContract.UserEntry.TABLE_NAME, values,
selection, selectionArgs);
                    break;
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
            if (updatedRows != 0) {
                mContext.getContentResolver().notifyChange(uri, null);
            }
            return updatedRows;
        } catch (Exception ex) {
            Log.e("Update", ex.toString());
        } finally {
            db.close();
        }
        return -1;
    }

}
```

Wie benutzt man:

```
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;
}
```

Eine Zeile in einer Tabelle aktualisieren

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

Eine Transaktion durchführen

Transaktionen können verwendet werden, um mehrere Änderungen an der Datenbank atomar vorzunehmen. Jede normale Transaktion folgt diesem Muster:

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

Das Aufrufen von `beginTransaction()` innerhalb einer aktiven Transaktion hat keine Auswirkungen.

Zeile (n) aus der Tabelle löschen

Alle Zeilen aus der Tabelle löschen

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

db.delete(TABLE_NAME, null, null);
db.close();
```

Löschen aller Zeilen aus der Tabelle und Abrufen der Anzahl der gelöschten Zeilen im Rückgabewert

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

int numRowsDeleted = db.delete(TABLE_NAME, String.valueOf(1), null);
db.close();
```

Zeilen mit `WHERE` Bedingung löschen

```
//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();
```

Bild in SQLite speichern

Einrichten der Datenbank

```
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);
    }
}
```

In die Datenbank einfügen:

```
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 );
}
```

Daten abrufen :

```
 byte[] image = cursor.getBlob(1);
```

Hinweis:

1. Vor dem Einfügen in eine Datenbank müssen Sie Ihr Bitmap-Bild zuerst in ein Byte-Array konvertieren und es anschließend mit der Datenbankabfrage anwenden.
2. Wenn Sie aus einer Datenbank abrufen, haben Sie sicherlich ein Byte-Array von Images. Sie müssen das Byte-Array zurück in das ursprüngliche Image konvertieren. Sie müssen also BitmapFactory zum Dekodieren verwenden.

Nachfolgend finden Sie eine Utility-Klasse, von der ich hoffe, dass sie Ihnen helfen kann:

```
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);
    }
```

```
}
```

Datenbank aus Assets-Ordner erstellen

Legen Sie Ihre Datei dbname.sqlite oder dbname.db im Ordner Assets Ihres Projekts ab.

```
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) {

        }
    }
```

So können Sie auf Datenbankobjekte auf Ihre Aktivitäten zugreifen.

```
// Create Databasehelper class object in your activity.
   private Databasehelper db;
```

Dann initialisieren Sie es in der onCreate-Methode und rufen Sie wie folgt die createDatabase () - Methode auf.

```
db = new Databasehelper(MainActivity.this);
        try {
            db.createDataBase();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

Führen Sie alle Einfüge-, Aktualisierungs-, Lösch- und Auswahlvorgänge wie unten gezeigt aus.

```
String query = "select Max(Id) as Id from " + TABLE_NAME;
        db.openDataBase();
        int count = db.getId(query);
        db.close();
```

Exportieren und Importieren einer Datenbank

Sie können Ihre Datenbank beispielsweise für Bacukups importieren und exportieren. Vergessen Sie nicht die Berechtigungen.

```
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();
    }
}
```

Bulk-Einsatz

Hier ist ein Beispiel für das Einfügen großer Datenblöcke gleichzeitig. Alle Daten, die Sie einfügen möchten, werden in einem ContentValues-Array gesammelt.

```
@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;
}
```

Und hier ist ein Beispiel, wie man es benutzt:

```
ContentResolver resolver = mContext.getContentResolver();
ContentValues[] valueList = new ContentValues[object.size()];
//add whatever you like to the valueList
resolver.bulkInsert(IChatContract.MessageColumns.CONTENT_URI, valueList);
```

SQLite online lesen: https://riptutorial.com/de/android/topic/871/sqlite

Kapitel 213: Startbildschirm erstellen

Bemerkungen

Das erste Beispiel (ein grundlegender Begrüßungsbildschirm) ist nicht der effizienteste Weg, um damit umzugehen. Als solches ist es ein grundlegender Begrüßungsbildschirm.

Examples

Ein grundlegender Begrüßungsbildschirm

Ein Begrüßungsbildschirm ist wie jede andere Aktivität, kann jedoch alle Ihre Startvorgänge im Hintergrund erledigen. Beispiel:

Manifest:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.package"
    android:versionCode="1"
    android:versionName="1.0" >

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".Splash"
            android:label="@string/app_name"
             >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

    </application>

</manifest>
```

Nun wird unser Splash-Screen als erste Aktivität aufgerufen.

Hier ein Beispiel für einen Begrüßungsbildschirm, der auch einige wichtige App-Elemente behandelt:

```
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();
```

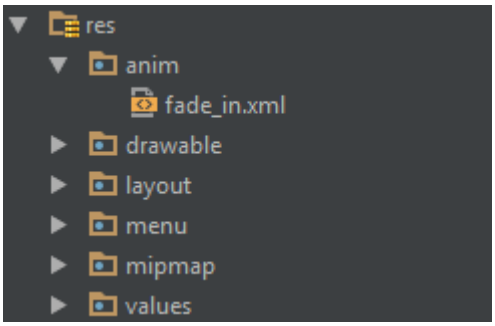
```
        finish();
    }

}
```

Splash-Bildschirm mit Animation

Dieses Beispiel zeigt einen einfachen, aber effektiven Begrüßungsbildschirm mit Animationen, die mit Android Studio erstellt werden können.

Schritt 1: Erstellen Sie eine Animation

Erstellen Sie ein neues Verzeichnis mit dem Namen *anim* im Verzeichnis *res* . Klicken Sie mit der rechten *Maustaste* darauf und erstellen Sie eine neue *Animationsressourcendatei mit dem* Namen *fade_in.xml* :



Fügen Sie dann den folgenden Code in die Datei *fade_in.xml ein* :

```
<?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>
```

Schritt 2: Erstellen Sie eine Aktivität

Erstellen Sie eine *leere Aktivität* mit Android Studio mit dem Namen `Splash` . Fügen Sie dann den folgenden Code ein:

```
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);
    }
}
```

Als nächstes fügen Sie den folgenden Code in die Layoutdatei ein:

```
<?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>
```

Schritt 3: Ersetzen Sie das Standard-Startprogramm

Verwandeln Sie Ihre `Splash` Aktivität in ein *Startprogramm,* indem Sie der *AndroidManifest*- Datei den folgenden Code *hinzufügen* :

```
<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>
```

Entfernen Sie anschließend die Standardstartaktivität, indem Sie den folgenden Code aus der *AndroidManifest*- Datei *entfernen* :

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />

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

Startbildschirm erstellen online lesen: https://riptutorial.com/de/android/topic/9316/startbildschirm-erstellen

Kapitel 214: Strict Mode Policy: Ein Werkzeug, um den Fehler in der Kompilierzeit zu erkennen.

Einführung

Strict Mode ist eine spezielle Klasse, die in Android 2.3 für das Debugging eingeführt wurde. Diese Entwicklerwerkzeuge erkennen versehentlich gemachte Dinge und machen sie darauf aufmerksam, damit wir sie beheben können. Sie wird am häufigsten verwendet, um den versehentlichen Festplatten- oder Netzwerkzugriff im Hauptthread der Anwendungen abzufangen, in dem Benutzeroberflächenoperationen empfangen werden und Animationen stattfinden. StrictMode ist im Grunde ein Werkzeug, um den Fehler im Kompiliermodus zu erfassen.

Bemerkungen

StrictMode ist im Grunde ein Werkzeug, um den Fehler im Kompiliermodus zu erfassen. Dadurch können wir Speicherlecks in unseren Anwendungen vermeiden.

Examples

Der folgende Code-Ausschnitt dient zum Einrichten des StrictMode für Thread-Richtlinien. Dieser Kodex ist an den Einstiegspunkten zu unserer Anwendung festzulegen.

```
StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
    .detectDiskWrites()
    .penaltyLog() //Logs a message to LogCat
    .build())
```

Der folgende Code befasst sich mit Speicherlecks, wie er feststellt, wenn in SQLLite finalize aufgerufen wird oder nicht.

```
StrictMode.setVmPolicy(new StrictMode.VmPolicy.Builder()
    .detectActivityLeaks()
    .detectLeakedClosableObjects()
    .penaltyLog()
    .build());
```

Strict Mode Policy: Ein Werkzeug, um den Fehler in der Kompilierzeit zu erkennen. online lesen: https://riptutorial.com/de/android/topic/8756/strict-mode-policy--ein-werkzeug--um-den-fehler-in-der-kompilierzeit-zu-erkennen-

Kapitel 215: SyncAdapter mit periodischer Synchronisierung der Daten

Einführung

Die Synchronisierungsadapterkomponente in Ihrer App kapselt den Code für die Aufgaben, die Daten zwischen dem Gerät und einem Server übertragen. Basierend auf der Planung und den Triggern, die Sie in Ihrer App bereitstellen, führt das Sync-Adapter-Framework den Code in der Sync-Adapterkomponente aus.

Kürzlich habe ich an SyncAdapter gearbeitet. Ich möchte mein Wissen mit anderen teilen, es kann anderen helfen.

Examples

Synchronisierungsadapter mit jedem minimalen Wert, der vom Server angefordert wird.

```
<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>
```

Dieser Code muss in der Manifestdatei hinzugefügt werden

Im obigen Code haben wir den Syncservice und den Conteprovider und den Authenticatorservice.

In der App müssen wir das XML-Paket erstellen, um Syncadpter- und Authenticator-XML-Dateien hinzuzufügen. **authenticator.xml**

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="@string/R.String.accountType"
    android:icon="@mipmap/ic_launcher"
    android:smallIcon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    />
```

Syncadapter

```
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"
    android:contentAuthority="@string/R.String.contentAuthority"
    android:accountType="@string/R.String.accountType"
    android:userVisible="true"
    android:allowParallelSyncs="true"
    android:isAlwaysSyncable="true"
    android:supportsUploading="false"/>
```

Authenticator

```
import android.accounts.AbstractAccountAuthenticator;
import android.accounts.Account;
import android.accounts.AccountAuthenticatorResponse;
import android.accounts.NetworkErrorException;
import android.content.Context;
import android.os.Bundle;

public class Authenticator extends AbstractAccountAuthenticator {
    private Context mContext;
    public Authenticator(Context context) {
        super(context);
         this.mContext=context;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse accountAuthenticatorResponse,
String s) {
        return null;
    }

    @Override
    public Bundle addAccount(AccountAuthenticatorResponse accountAuthenticatorResponse, String
s, String s1, String[] strings, Bundle bundle) throws NetworkErrorException {
        return null;
    }
```

```
    @Override
    public Bundle confirmCredentials(AccountAuthenticatorResponse
accountAuthenticatorResponse, Account account, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle getAuthToken(AccountAuthenticatorResponse accountAuthenticatorResponse,
Account account, String s, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public String getAuthTokenLabel(String s) {
        return null;
    }

    @Override
    public Bundle updateCredentials(AccountAuthenticatorResponse accountAuthenticatorResponse,
Account account, String s, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle hasFeatures(AccountAuthenticatorResponse accountAuthenticatorResponse,
Account account, String[] strings) throws NetworkErrorException {
        return null;
    }
}
```

AuthenticatorService

```
public class AuthenticatorService extends Service {

    private Authenticator authenticator;

    public AuthenticatorService() {
        super();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        IBinder ret = null;
        if (intent.getAction().equals(AccountManager.ACTION_AUTHENTICATOR_INTENT)) ;
        ret = getAuthenticator().getIBinder();
        return ret;
    }

    public Authenticator getAuthenticator() {
        if (authenticator == null)
            authenticator = new Authenticator(this);
        return authenticator;
    }
}
```

IpDataDBHelper

```
 public class IpDataDBHelper extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION=1;
    private static final String DATABASE_NAME="ip.db";
    public static final String TABLE_IP_DATA="ip";

    public static final String COLUMN_ID="_id";
    public static final String COLUMN_IP="ip";
    public static final String COLUMN_COUNTRY_CODE="country_code";
    public static final String COLUMN_COUNTRY_NAME="country_name";
    public static final String COLUMN_CITY="city";
    public static final String COLUMN_LATITUDE="latitude";
    public static final String COLUMN_LONGITUDE="longitude";

    public IpDataDBHelper(Context context, String name, SQLiteDatabase.CursorFactory factory,
int version) {
        super(context, DATABASE_NAME, factory, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        String CREATE_TABLE="CREATE TABLE " + TABLE_IP_DATA + "( " + COLUMN_ID + " INTEGER
PRIMARY KEY ,"
                + COLUMN_IP + " INTEGER ," + COLUMN_COUNTRY_CODE + " INTEGER ," +
COLUMN_COUNTRY_NAME +
                " TEXT ," + COLUMN_CITY + " TEXT ," + COLUMN_LATITUDE + " INTEGER ," +
COLUMN_LONGITUDE + " INTEGER)";
        sqLiteDatabase.execSQL(CREATE_TABLE);
        Log.d("SQL",CREATE_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + TABLE_IP_DATA);
        onCreate(sqLiteDatabase);
    }

    public long AddIPData(ContentValues values)
    {
        SQLiteDatabase sqLiteDatabase =getWritableDatabase();
        long insertedRow=sqLiteDatabase.insert(TABLE_IP_DATA,null,values);
        return insertedRow;
    }

    public Cursor getAllIpData()
    {
        String[]
projection={COLUMN_ID,COLUMN_IP,COLUMN_COUNTRY_CODE,COLUMN_COUNTRY_NAME,COLUMN_CITY,COLUMN_LATITUDE,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;
    }
}
```

Hauptaktivität

```
     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();
}
```

}

SyncAdapter mit periodischer Synchronisierung der Daten online lesen: https://riptutorial.com/de/android/topic/10774/syncadapter-mit-periodischer-synchronisierung-der-daten

Kapitel 216: Systemzeichensatznamen abrufen und Schriftarten verwenden

Einführung

Die folgenden Beispiele zeigen, wie Sie die Standardnamen der Systemzeichensätze abrufen, die im Verzeichnis / *system* / *fonts* / gespeichert sind, und wie Sie mit einer Systemschriftart die Schriftart eines `TextView` Elements `TextView` .

Examples

Systemzeichensatznamen abrufen

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

Anwenden einer Systemschriftart auf eine TextView

Im folgenden Code müssen Sie `fontsname` durch den Namen der Schriftart ersetzen, die Sie verwenden möchten:

```
TextView lblexample = (TextView) findViewById(R.id.lblexample);
lblexample.setTypeface(Typeface.createFromFile("/system/fonts/" + "fontsname" + ".ttf"));
```

Systemzeichensatznamen abrufen und Schriftarten verwenden online lesen: https://riptutorial.com/de/android/topic/10930/systemzeichensatznamen-abrufen-und-schriftarten-verwenden

Kapitel 217: TabLayout

Examples

TabLayout ohne ViewPager verwenden

In den meisten `TabLayout` wird ein `TabLayout` zusammen mit einem ViewPager verwendet , um die zugehörige Wischfunktion zu erhalten.

Es ist möglich, ein `TabLayout` ohne `ViewPager` mithilfe eines `TabLayout.OnTabSelectedListener` .

`TabLayout` Sie zunächst ein `TabLayout` zur XML-Datei Ihrer Aktivität hinzu:

```
<android.support.design.widget.TabLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:id="@+id/tabLayout" />
```

Füllen Sie für die Navigation innerhalb einer `Activity` die Benutzeroberfläche basierend auf der ausgewählten Registerkarte manuell aus.

```
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) {

    }
});
```

TabLayout online lesen: https://riptutorial.com/de/android/topic/7601/tablayout

Kapitel 218: Tastatur

Examples

Blenden Sie die Tastatur aus, wenn der Benutzer an eine andere Stelle auf dem Bildschirm tippt

Fügen Sie Code in Ihre **Aktivität ein** .

Dies würde auch für **Fragment** funktionieren, und Sie **müssen** diesen Code nicht in **Fragment** hinzufügen.

```
@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);
}
```

Registrieren Sie einen Rückruf für das Öffnen und Schließen der Tastatur

Die Idee ist, ein Layout vor und nach jeder Änderung zu messen, und wenn sich eine wesentliche Änderung ergibt, können Sie sicher sein, dass es das Softkeyboard ist.

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

Dann `onCreate` in unserem `onCreate` den Anfangswert für `mLastContentHeight`

```
mLastContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();
```

und fügen Sie den Listener hinzu

```
rootView.getViewTreeObserver().addOnGlobalLayoutListener(keyboardLayoutListener);
```

Vergessen Sie nicht, den Hörer bei `destroy` zu entfernen

```
rootView.getViewTreeObserver().removeOnGlobalLayoutListener(keyboardLayoutListener);
```

Tastatur online lesen: https://riptutorial.com/de/android/topic/5606/tastatur

Kapitel 219: Taste

Syntax

- <Button ... />
- android: onClick = "methodname"
- button.setOnClickListener (new OnClickListener () {...});
- public class classname implementiert View.OnLongClickListener

Examples

inline onClickListener

Angenommen, wir haben eine Schaltfläche (wir können sie programmgesteuert erstellen oder sie mithilfe von findViewbyId () usw. aus einer Ansicht binden).

```
Button btnOK = (...)
```

Erstellen Sie nun eine anonyme Klasse und setzen Sie sie inline.

```
btnOk.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Do stuff here...
    }
});
```

Verwenden des Layouts zum Definieren einer Klickaktion

Wenn wir im Layout eine Schaltfläche erstellen, können wir das Attribut `android:onClick` , um auf eine Methode im Code zu verweisen, die Klicks verarbeitet.

Taste

```
<Button
    android:width="120dp"
    android:height="wrap_content"
    android:text="Click me"
    android:onClick="handleClick" />
```

Erstellen `handleClick` anschließend in Ihrer Aktivität die `handleClick` Methode.

```
public void handleClick(View v) {
    // Do whatever.
}
```

Verwenden Sie dasselbe Click-Ereignis für eine oder mehrere Ansichten in XML

Wenn wir eine Ansicht im Layout erstellen, können wir das Attribut android: onClick verwenden, um auf eine Methode in der zugeordneten Aktivität oder im zugehörigen Fragment zu verweisen, um die Klickereignisse zu verarbeiten.

XML-Layout

```
<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" />
```

Aktivitäts- / Fragmentcode

Erstellen Sie in Ihrem **Code** die von Ihnen benannte Methode, wobei v die berührte Ansicht ist, und führen Sie für jede Ansicht, die diese Methode aufruft, etwas aus.

```
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;
    }
}
```

Wenn Sie möchten, können Sie für jede Ansicht auch eine andere Methode verwenden (in diesem Fall müssen Sie natürlich nicht nach der ID suchen).

Hören Sie sich die Long Click-Events an

Um einen langen Klick abzufangen und zu verwenden, müssen Sie den entsprechenden Listener für die Schaltfläche angeben:

```
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);
```

Wann sollte ich es benutzen?

- Wenn der Code in einem Inline-Listener zu groß ist und Ihre Methode / Klasse hässlich wird und schwer zu lesen ist
- Sie möchten dieselbe Aktion in verschiedenen Elementen (Ansichten) Ihrer App ausführen

---

Dazu müssen Sie eine Klasse erstellen, die einen der Listener in der View-API implementiert.

Geben Sie beispielsweise Hilfe, wenn Sie lange auf ein Element klicken:

```
public class HelpLongClickListener implements View.OnLongClickListener
{
    public HelpLongClickListener() {
    }

    @Override
    public void onLongClick(View v) {
        // show help toast or popup
    }
}
```

Dann brauchen Sie nur ein Attribut oder eine Variable in Ihrer `Activity` , um sie verwenden zu können:

```
HelpLongClickListener helpListener = new HelpLongClickListener(...);

button1.setOnClickListener(helpListener);
button2.setOnClickListener(helpListener);
label.setOnClickListener(helpListener);
button1.setOnClickListener(helpListener);
```

ANMERKUNG: Das Definieren von Listenern in getrennten Klassen hat einen Nachteil. Sie kann nicht direkt auf Klassenfelder zugreifen. Daher müssen Sie Daten (Kontext, Ansicht) über den Konstruktor übergeben, sofern Sie keine Attribute öffentlich machen oder Geter definieren.

Benutzerdefiniert Klicken Sie auf Listener, um mehrere schnelle Klicks zu verhindern

Um zu **verhindern** , **dass eine Schaltfläche innerhalb kurzer Zeit mehrmals *ausgelöst wird*** (beispielsweise 2 Klicks innerhalb einer Sekunde, was zu schwerwiegenden Problemen führen kann, wenn der Fluss nicht gesteuert wird), kann ein benutzerdefinierter ***SingleClickListener implementiert werden*** .

Dieser ClickListener legt ein bestimmtes Zeitintervall als Schwellenwert fest (z. B. 1000 ms). Wenn Sie auf die Schaltfläche klicken, wird geprüft, ob der Auslöser in der von Ihnen definierten Zeit ausgeführt wurde. Wenn nicht, wird er ausgelöst.

```
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);

}
```

In der Klasse wird der SingleClickListener mit dem betreffenden Button verknüpft

```
myButton = (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(new SingleClickListener() {
    @Override
    public void performClick(View view) {
        // do stuff
    }
});
```

Anpassen der Schaltflächenart

Es gibt viele Möglichkeiten, das Aussehen eines Buttons anzupassen. Dieses Beispiel bietet mehrere Optionen:

Option 0: ThemeOverlay verwenden (derzeit der einfachste / schnellste Weg)

Erstellen Sie einen neuen Stil in Ihrer Stildatei:

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>
```

Dann in dem Layout, wo Sie Ihren Button platzieren (zB MainActivity):

activity_main.xml

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

            <Button
                android:id="@+id/mybutton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello"
                android:theme="@style/mybutton"
                style="@style/Widget.AppCompat.Button.Colored"/>

</LinearLayout>
```

Option 1: Erstellen Sie Ihren eigenen Schaltflächenstil

Erstellen Sie in values / styles.xml einen neuen Stil für Ihre Schaltfläche:

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>
```

Dann in dem Layout, wo Sie Ihren Button platzieren (zB in MainActivity):

activity_main.xml

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

```
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

            <Button
                android:id="@+id/mybutton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello"
                android:theme="@style/mybuttonstyle"/>

</LinearLayout>
```

Option 2: Weisen Sie jedem Button-Status ein Zeichen zu

Erstellen Sie eine XML-Datei mit dem Namen 'mybuttondrawable.xml' in einem zeichnbaren Ordner, um die zeichnungsfähigen Ressourcen für jeden Ihrer Schaltflächenstatus zu definieren:

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>
```

Bei jedem dieser Zeichensätze kann es sich um Bilder (z. B. mybutton_disabled.png) oder XML-Dateien handeln, die von Ihnen definiert und im Zeichnungsordner gespeichert werden. Zum Beispiel:

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>
```

Dann in dem Layout, wo Sie Ihren Button platzieren (zB MainActivity):

activity_main.xml

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

            <Button
                android:id="@+id/mybutton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello"
                android:background="@drawable/mybuttondrawable"/>

</LinearLayout>
```

Option 3: Fügen Sie Ihrem App-Design Ihren Schaltflächenstil hinzu

Sie können den Standardstil für Android-Schaltflächen in der Definition Ihres App-Designs (in values / styles.xml) überschreiben.

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>
```

Option 4: Überlagern Sie eine Farbe programmgesteuert mit dem Standardschaltflächenstil

Finden Sie einfach Ihre Schaltfläche und wenden Sie einen Farbfilter an:

```
Button mybutton = (Button) findViewById(R.id.mybutton);
mybutton.getBackground().setColorFilter(anycolor, PorterDuff.Mode.MULTIPLY)
```

Sie können verschiedene Füllmethoden überprüfen hier und schöne Beispiele hier .

Taste online lesen: https://riptutorial.com/de/android/topic/5607/taste

Kapitel 220: Telefonnummern mit Muster formatieren.

Einführung

Dieses Beispiel zeigt Ihnen, wie Sie Telefonnummern mit einem Muster formatieren

Sie benötigen die folgende Bibliothek in Ihrem Schulabschluss.

compile 'com.googlecode.libphonennummer: libphonennummer: 7.2.2'

Examples

Muster + 1 (786) 1234 5678

Bei einer normalisierten Telefonnummer wie +178612345678 erhalten wir eine formatierte Nummer mit dem angegebenen Muster.

```
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;
}
```

Telefonnummern mit Muster formatieren. online lesen:
https://riptutorial.com/de/android/topic/9083/telefonnummern-mit-muster-formatieren-

Kapitel 221: TensorFlow

Einführung

TensorFlow wurde für mobile und eingebettete Plattformen konzipiert. Wir haben Beispielcode und Build-Support, den Sie jetzt für diese Plattformen ausprobieren können:

Android iOS Raspberry Pi

Bemerkungen

Anerkannte Arbeit von MindRocks

Examples

Wie benutzt man

Installieren Sie Bazel von hier aus . Bazel ist das primäre Buildsystem für TensorFlow. Bearbeiten Sie nun den WORKSPACE. Wir finden die WORKSPACE-Datei im Stammverzeichnis des TensorFlow, das wir zuvor geklont haben.

```
# 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)
```

Wie unten mit unserem sdk- und ndk-Pfad:

```
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)
```

TensorFlow online lesen: https://riptutorial.com/de/android/topic/9991/tensorflow

Kapitel 222: Testen der Benutzeroberfläche mit Espresso

Bemerkungen

Espresso

Espresso Spickzettel hilft Ihnen, Ihre Tests zu schreiben und was Sie testen möchten:

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Ein guter Bezugspunkt ist auch immer die offizielle Dokumentation:

https://google.github.io/android-testing-support-library/docs/espresso/index.html

Fortgeschrittene Espresso-Video-Vorschläge von Google:
https://www.youtube.com/watch?v=isihPOY2vS4

Fehlerbehebung

- Schließen Sie beim Scrollen unbedingt die Tastatur:

Achtung: Wenn Sie die "Espresso" -Version nicht verwenden, können Sie nichts tun, wenn Sie sie außerhalb einer ViewAction verwenden. Dies ist möglicherweise nicht offensichtlich, wenn Sie einen Import in der ViewAction-Version haben, da sie exakt denselben Methodennamen haben.

```
ViewActions.closeSoftKeyboard;
Espresso.closeSoftKeyboard();
```

- Wenn Sie Tests in einer Suite statt einzeln ausführen, beachten Sie, dass die Aktivität des vorherigen Tests möglicherweise noch ausgeführt wird. Verlassen Sie sich nicht darauf, dass onDestroy () des vorherigen Tests vor den aktuellen Tests onResume () aufgerufen wird. **Es stellt sich heraus, dass dies tatsächlich ein Fehler ist** : http://b.android.com/201513

Examples

Espresso einrichten

`build.gradle` Sie in der `build.gradle` Datei Ihres Android-App-Moduls die nächsten Abhängigkeiten hinzu:

```
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'
}
```

`AndroidJUnitRunner` den `AndroidJUnitRunner` für den Parameter `testInstrumentationRunner` in der Datei `build.gradle` .

```
android {

  defaultConfig {
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }

}
```

Fügen Sie außerdem diese Abhängigkeit hinzu, um Unterstützung für Vorspiegelungen bereitzustellen

```
androidTestCompile 'com.android.support.test.espresso:espresso-intents:2.2.2'
```

Fügen Sie diese Option für die Unterstützung von Webview-Tests hinzu

```
// Espresso-web for WebView support
androidTestCompile 'com.android.support.test.espresso:espresso-web:2.2.2'
```

Erstellen Sie eine Espresso-Testklasse

Legen Sie die nächste Java-Klasse in **src / androidTest / java ab** und führen Sie sie aus.

```
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
  }

}
```

Öffnen Sie das DrawerLayout

```
public final class DrawerLayoutTest  {
```

```
  @Test public void Open_Close_Drawer_Layout() {
    onView(withId(R.id.drawer_layout)).perform(actionOpenDrawer());
    onView(withId(R.id.drawer_layout)).perform(actionCloseDrawer());
  }

  public static ViewAction actionOpenDrawer() {
    return new ViewAction() {
      @Override public Matcher<View> getConstraints() {
        return isAssignableFrom(DrawerLayout.class);
      }

      @Override public String getDescription() {
        return "open drawer";
      }

      @Override public void perform(UiController uiController, View view) {
        ((DrawerLayout) view).openDrawer(GravityCompat.START);
      }
    };
  }

  public static ViewAction actionCloseDrawer() {
    return new ViewAction() {
      @Override public Matcher<View> getConstraints() {
        return isAssignableFrom(DrawerLayout.class);
      }

      @Override public String getDescription() {
        return "close drawer";
      }

      @Override public void perform(UiController uiController, View view) {
        ((DrawerLayout) view).closeDrawer(GravityCompat.START);
      }
    };
  }

}
```

Espresso einfacher UI-Test

UI-Testwerkzeuge

Zwei Hauptwerkzeuge, die heutzutage hauptsächlich für UI-Tests verwendet werden, sind Appium und Espresso.

Appium	Espresso
Blackbox-Test	weiße / graue Boxprüfung
Was Sie sehen, ist das, was Sie testen können	kann die inneren Abläufe der App ändern und für den Test vorbereiten, z. B. vor dem Ausführen des Tests einige Daten in einer Datenbank oder gemeinsam genutzte Präferenzen speichern
wird hauptsächlich für Integrations-	Testen der Funktionalität eines Bildschirms und / oder

Appium	Espresso
End-to-End-Tests und gesamte Benutzerflüsse verwendet	Flusses
kann abstrahiert werden, so dass der geschriebene Test auf iOS und Android ausgeführt werden kann	Nur Android
gut unterstützt	gut unterstützt
unterstützt parallele Tests an mehreren Geräten mit Selengitter	Parallele Tests sind nicht möglich, Plugins wie Spoon sind vorhanden, bis echte Unterstützung von Google herauskommt

So fügen Sie Espresso zum Projekt hinzu

```
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'
}
```

HINWEIS Wenn Sie die neuesten Support-Bibliotheken, Anmerkungen usw. verwenden, müssen Sie ältere Versionen von Espresso ausschließen, um Kollisionen zu vermeiden:

```
    // 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'
    }
```

Abgesehen von diesen Importen ist es erforderlich, Android-Testläufer zu build.gradle android.defaultConfig hinzuzufügen:

```
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
```

Gerätkonfiguration

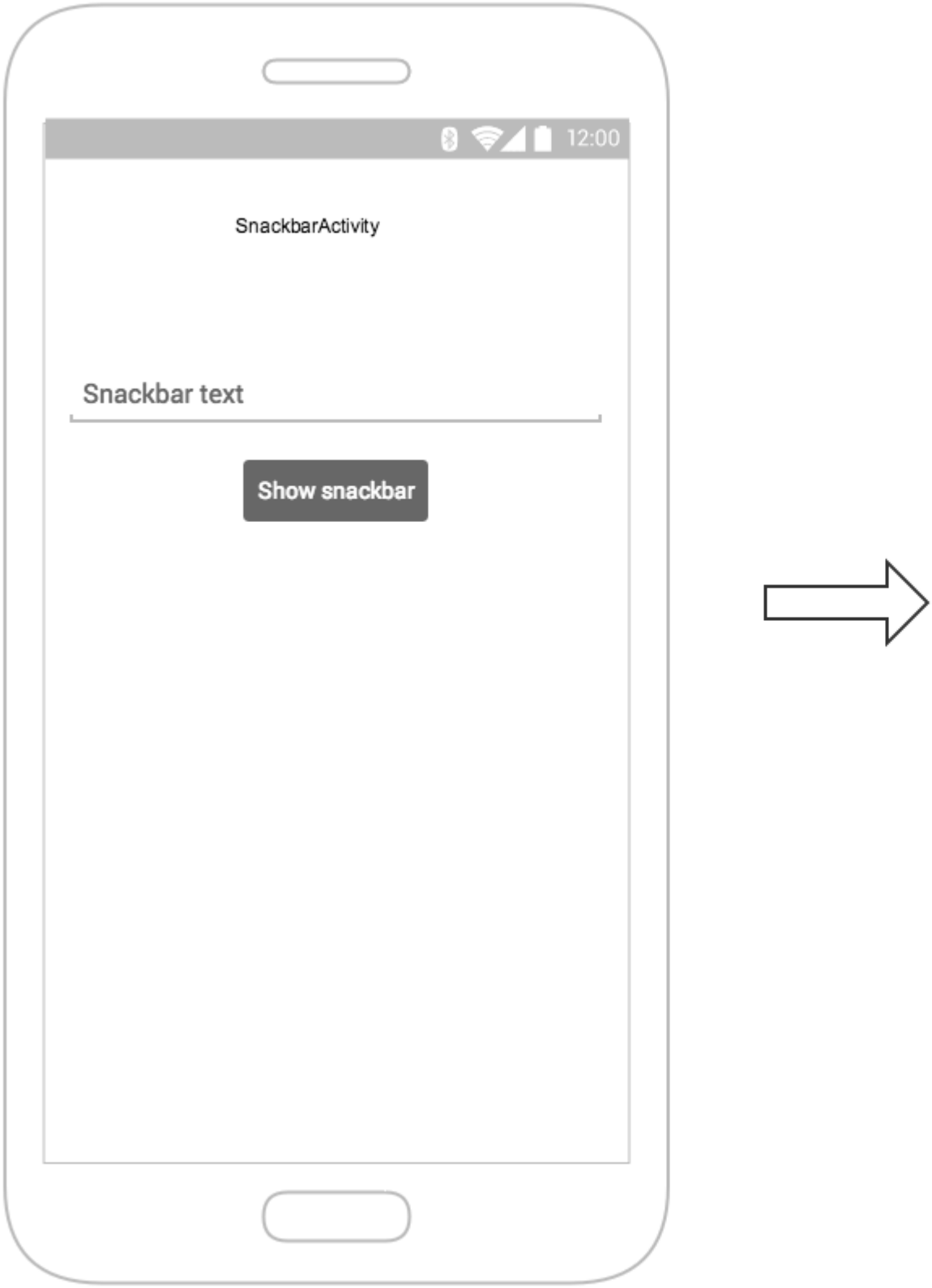
Für nicht flockige Tests wird empfohlen, die folgenden Einstellungen an Ihren Geräten vorzunehmen:

- Entwickleroptionen / Animationen deaktivieren - reduziert die Flakynität von Tests
- Entwickleroptionen / Wach bleiben - wenn Sie dedizierte Geräte für Tests haben, ist dies nützlich
- Entwickleroptionen / Logger-Puffergrößen: Legen Sie eine höhere Anzahl fest, wenn Sie sehr große Testreihen auf Ihrem Telefon ausführen
- Accessibility / Touch & Hold-Verzögerung - lang, um Probleme beim Tippen auf Espresso zu vermeiden

Ziemlich ein Setup aus der realen Welt? Nun, wenn das nicht möglich ist, werfen wir einen Blick auf die Einrichtung eines kleinen Tests

Test schreiben

Nehmen wir an, wir haben den folgenden Bildschirm:



Der Bildschirm enthält:

- Texteingabefeld - **R.id.textEntry**
- Schaltfläche, die die Snackbar mit eingetipptem Text **anzeigt** - **R.id.shownSnackbarBtn**

- Snackbar, die den vom Benutzer eingegebenen Text enthalten sollte - **android.support.design.R.id.snackbar_text**

Lassen Sie uns nun eine Klasse erstellen, die unseren Fluss testet:

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

Wie Sie bemerkt haben, gibt es 3-4 Dinge, die Sie häufig bemerken:

onView (withXYZ) <- viewMatchers Mit diesen können Sie Elemente auf dem Bildschirm finden

perform (click ()) <- viewActions, Sie können Aktionen für Elemente ausführen, die Sie zuvor

gefunden haben

check (matches (isDisplayed ())) <- viewAssertions: Überprüft, ob Sie die zuvor gefundenen Bildschirme ausführen möchten

Alle diese und viele andere finden Sie hier: https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/index.html

Jetzt können Sie den Test ausführen, indem Sie mit der rechten Maustaste auf den Klassennamen / Test klicken und Test ausführen oder mit dem Befehl auswählen:

```
./gradlew connectedFLAVORNAMEAndroidTest
```

Navigation nach oben

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

Beachten Sie, dass dies eine Problemumgehung ist und mit anderen Ansichten kollidieren wird, die dieselbe Inhaltsbeschreibung haben.

Ausführen einer Aktion für eine Ansicht

Es ist möglich, `ViewActions` für eine Ansicht mithilfe der Perform-Methode auszuführen.
Die `ViewActions` Klasse stellt `ViewActions` für die häufigsten Aktionen bereit, z.

```
ViewActions.click()
ViewActions.typeText()
ViewActions.clearText()
```

Um zum Beispiel auf die Ansicht zu klicken:

```
onView(...).perform(click());
onView(withId(R.id.button_simple)).perform(click());
```

Sie können mehrere Aktionen mit einem Perform-Aufruf ausführen:

```
onView(...).perform(typeText("Hello"), click());
```

Wenn sich die Ansicht, mit der Sie arbeiten, in einer `ScrollView` (vertikal oder horizontal) befindet, sollten Sie vorhergehende Aktionen in Betracht ziehen, die eine Anzeige der Ansicht (wie `click()` und `typeText()`) mit `scrollTo()` . Dadurch wird sichergestellt, dass die Ansicht angezeigt wird,

bevor mit der anderen Aktion fortgefahren wird:

```
onView(...).perform(scrollTo(), click());
```

Eine Ansicht mit onView finden

Mit den `ViewMatchers` können Sie Ansichten in der aktuellen Ansichtshierarchie finden.

Um eine Ansicht zu finden, verwenden Sie die Methode `onView()` mit einem View Matcher, der die richtige Ansicht auswählt. Die `onView()` -Methoden geben ein Objekt vom Typ `ViewInteraction` .

Zum Beispiel ist es so einfach, eine Ansicht anhand ihrer `R.id` :

```
onView(withId(R.id.my_view))
```

Eine Ansicht mit einem Text suchen:

```
onView(withText("Hello World"))
```

Espresso benutzerdefinierte Matchers

Espresso verfügt standardmäßig über viele Übereinstimmungen, mit deren Hilfe Sie Ansichten finden, die Sie für einige Überprüfungen oder Interaktionen mit ihnen benötigen.

Die wichtigsten finden Sie im folgenden Spickzettel:

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Einige Beispiele für Matcher sind:

- withId (R.id.ID_of_object_you_are_looking_for);
- withText ("Text, den Sie für ein Objekt erwarten");
- isDisplayed () <- Prüfen Sie, ob die Ansicht sichtbar ist
- doesNotExist () <- Überprüfen Sie, ob die Ansicht nicht vorhanden ist

All dies ist sehr nützlich für den täglichen Gebrauch. Wenn Sie jedoch komplexere Ansichten haben, können Ihre benutzerdefinierten Matcher die Tests lesbarer machen und sie an verschiedenen Stellen wiederverwenden.

Es gibt zwei gängige Typen von Matchern, die Sie erweitern können: **TypeSafeMatcher BoundedMatcher**

Um TypeSafeMatcher zu implementieren, müssen Sie die Instanz überprüfen. Wenn die Ansicht, für die Sie eine Bestätigung ausführen, der richtige Typ ist, in dem Sie einige ihrer Eigenschaften mit einem Wert vergleichen, den Sie einem Matcher bereitgestellt haben.

Geben Sie zum Beispiel sicheres Matcher ein, das eine Bildansicht überprüft.

```
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("]");
        }
    }
}
```

Die Verwendung des Streichholzspielers könnte folgendermaßen dargestellt werden:

```
  public static Matcher<View> withDrawable(final int resourceId) {
    return new DrawableMatcher(resourceId);
}

 onView(withDrawable(R.drawable.someDrawable)).check(matches(isDisplayed()));
```

Bounded Matcher sind ähnlich, Sie müssen die Typprüfung jedoch nicht durchführen, da dies jedoch automatisch für Sie erledigt wird:

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

Weitere Informationen zu Matchern finden Sie unter:

http://hamcrest.org/

https://developer.android.com/reference/android/support/test/espresso/matcher/ViewMatchers.html

Insgesamt Espresso

Setup Espresso:

```
androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
androidTestCompile 'com.android.support.test:runner:0.5'
```

ViewMatchers - Eine Sammlung von Objekten, die `Matcher<? super View>` implementieren `Matcher<? super View>` Schnittstelle. Sie können eine oder mehrere davon an die `onView` Methode übergeben, um eine Ansicht in der aktuellen Ansichtshierarchie zu `onView` .

ViewActions - Eine Auflistung von `ViewActions` , die an die `ViewInteraction.perform()` Methode

übergeben werden können (z. B. `click()`).

ViewAssertions - Eine Auflistung von `ViewAssertions` , an die die `ViewInteraction.check()` -Methode übergeben werden kann. In den meisten Fällen verwenden Sie die Übereinstimmungsassertion, bei der mithilfe eines View-Matchers der Status der aktuell ausgewählten Ansicht bestätigt wird.

Espresso Spickzettel von Google

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion);
```

View Matchers

USER PROPERTIES

```
withId(...)
withText(...)
withTagKey(...)
withTagValue(...)
hasContentDescription(...)
withContentDescription(...)
withHint(...)
withSpinnerText(...)
hasLinks()
hasEllipsizedText()
hasMultilineTest()
```

UI PROPERTIES

```
isDisplayed()
isCompletelyDisplayed()
isEnabled()
hasFocus()
isClickable()
isChecked()
isNotChecked()
withEffectiveVisibility(...)
isSelected()
```

OBJECT MATCHER

```
allOf(Matchers)
anyOf(Matchers)
is(...)
```

HIERARCHY

```
withParent(Matcher)
withChild(Matcher)
hasDescendant(Matcher)
isDescendantOfA(Matcher)
hasSibling(Matcher)
isRoot()
```

INPUT

```
supportsInputMethods(...)
hasIMEAction(...)
```

CLASS

```
isAssignableFrom(...)
withClassName(...)
```

ROOT MATCHERS

```
isFocusable()
isTouchable()
isDialog()
withDecorView()
isPlatformPopup()
```

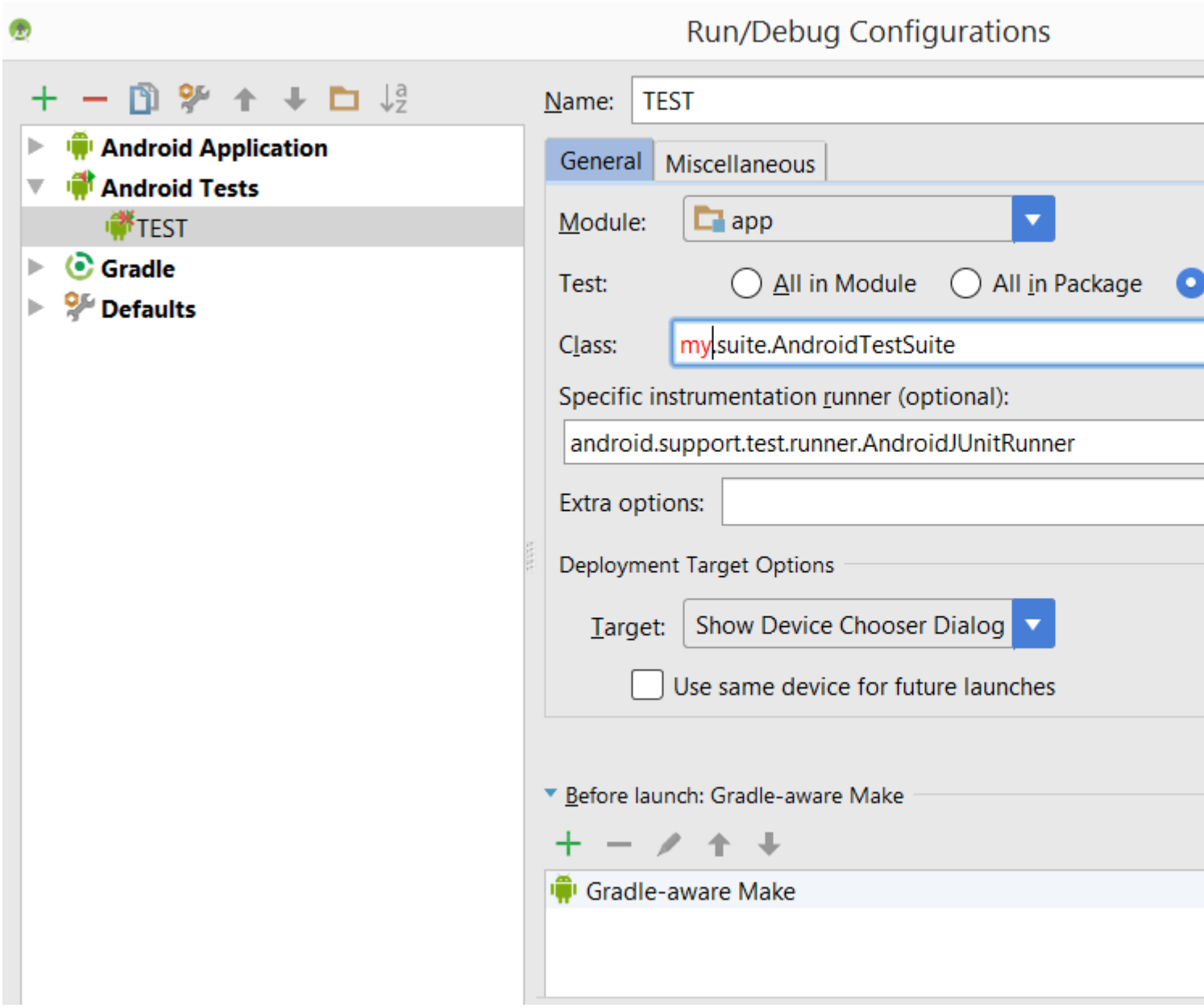
SEE ALSO

Preference matchers

Suite organisieren .

```
/**
 * Runs all unit tests.
 */
@RunWith(Suite.class)
@Suite.SuiteClasses({MyTest1.class ,
         MyTest2.class,
         MyTest3.class})
public class AndroidTestSuite {}
```

In AndroidStudio können Sie dann Gradle ausführen oder eine neue Konfiguration festlegen, z.



Testsuiten können verschachtelt sein.

Testen der Benutzeroberfläche mit Espresso online lesen:
https://riptutorial.com/de/android/topic/3485/testen-der-benutzeroberflache-mit-espresso

Kapitel 223: Text bearbeiten

Examples

Mit EditTexts arbeiten

Der EditText ist das Standard-Widget für Texteingabe in Android-Apps. Wenn der Benutzer Text in eine App eingeben muss, ist dies der primäre Weg, dies zu tun.

Text bearbeiten

Es gibt viele wichtige Eigenschaften, die festgelegt werden können, um das Verhalten eines EditText anzupassen. Einige davon sind unten aufgeführt. Weitere Informationen zum Eingabefeld finden Sie im offiziellen Textfeldleitfaden.

Verwendungszweck

Ein EditText wird einem Layout mit allen Standardverhalten mit dem folgenden XML hinzugefügt:

```
<EditText
    android:id="@+id/et_simple"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
</EditText>
```

Beachten Sie, dass ein EditText lediglich eine dünne Erweiterung der TextView ist und alle gleichen Eigenschaften erbt.

Den Wert abrufen

Den Wert des in einen EditText eingegebenen Textes lautet wie folgt:

```
EditText simpleEditText = (EditText) findViewById(R.id.et_simple);
String strValue = simpleEditText.getText().toString();
```

Weitere Anpassung des Eintrags

Möglicherweise möchten Sie den Eintrag auf eine einzeilige Textzeile beschränken (Zeilenumbrüche vermeiden):

```
<EditText
  android:singleLine="true"
  android:lines="1"
/>
```

Sie können die Zeichen, die in ein Feld eingegeben werden können, mithilfe des Ziffernattributs einschränken:

```
<EditText
  android:inputType="number"
  android:digits="01"
/>
```

Dies würde die eingegebenen Ziffern auf "0" und "1" beschränken. Vielleicht möchten Sie die Gesamtzahl der Zeichen mit:

```
<EditText
  android:maxLength="5"
/>
```

Mit diesen Eigenschaften können wir das erwartete Eingabeverhalten für Textfelder definieren.

Farben einstellen

Sie können die Hintergrundfarbe für hervorgehobenen Text innerhalb eines EditText mit der Eigenschaft `android:textColorHighlight` :

```
<EditText
    android:textColorHighlight="#7cff88"
/>
```

Anzeigen von Platzhalterhinweisen

Möglicherweise möchten Sie den Hinweis für das EditText-Steuerelement festlegen, um einen Benutzer zur Eingabe bestimmter Eingaben aufzufordern:

```
<EditText
    ...
    android:hint="@string/my_hint">
</EditText>
```

Hinweise

Ändern der Farbe der unteren Zeile

Wenn Sie die AppCompat-Bibliothek verwenden, können Sie die Stile colorControlNormal, colorControlActivated und colorControlHighlight überschreiben:

```
<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>
```

Wenn diese Stile nicht in einem DialogFragment angewendet werden, liegt ein bekannter Fehler vor, wenn der LayoutInflater an die onCreateView () - Methode übergeben wird.

Das Problem wurde bereits in der AppCompat v23-Bibliothek behoben. In dieser Anleitung

erfahren Sie, wie Sie ein Upgrade durchführen. Eine weitere temporäre Problemumgehung besteht darin, den Layout-Inflator der Aktivität anstelle desjenigen zu verwenden, der an die onCreateView () -Methode übergeben wird:

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = getActivity().getLayoutInflater().inflate(R.layout.dialog_fragment,
container);
  }
```

Auf EditText-Eingabe warten

In den Cliffnotes der grundlegenden Ereignis-Listener erfahren Sie, wie Sie Änderungen an einem EditText überwachen und eine Aktion ausführen, wenn diese Änderungen auftreten.

Anzeige des Floating Label Feedbacks

Traditionell verbirgt der EditText die Hinweisnachricht (oben erläutert), nachdem der Benutzer mit der Eingabe begonnen hat. Außerdem mussten eventuelle Überprüfungsfehlermeldungen vom Entwickler manuell verwaltet werden.

Mit dem `TextInputLayout` Sie ein Floating-Label `TextInputLayout` , um Hinweise und Fehlermeldungen anzuzeigen. Weitere Details finden Sie hier .

Anpassen des InputType

Textfelder können unterschiedliche Eingabetypen haben, z. B. Nummer, Datum, Kennwort oder E-Mail-Adresse. Der Typ bestimmt, welche Art von Zeichen innerhalb des Feldes zulässig sind, und fordert die virtuelle Tastatur möglicherweise auf, ihr Layout für häufig verwendete Zeichen zu optimieren.

Standardmäßig werden alle Textinhalte in einem `EditText` Steuerelement als `EditText` Text angezeigt. Durch das Setzen des `inputType` Attributs können wir die Eingabe verschiedener Arten von Informationen wie Telefonnummern und Kennwörtern erleichtern:

```
<EditText
    ...
    android:inputType="phone">
</EditText>
```

Zu den häufigsten Eingabetypen gehören:

Art	Beschreibung
textUri	Text, der als URI verwendet wird
textEmailAddress	Text, der als E-Mail-Adresse verwendet wird
textPersonName	Text, der der Name einer Person ist

Art	Beschreibung
textPassword	Text, der ein Kennwort ist, das verdeckt werden soll
Nummer	Nur ein numerisches Feld
Telefon	Zur Eingabe einer Telefonnummer
Datum	Zur Eingabe eines Datums
Zeit	Zur Eingabe einer Zeit
textMultiLine	Erlauben Sie mehrere Textzeilen im Feld

Mit dem `android:inputType` können Sie auch bestimmte Tastaturverhalten angeben, z. B. ob Sie alle neuen Wörter groß `android:inputType` oder Funktionen wie die automatische Vervollständigung und Vorschläge zur Rechtschreibung verwenden möchten.
Hier sind einige der häufigsten Eingabetypwerte, die das Tastaturverhalten definieren:

Art	Beschreibung
textCapSentences	Normale Texttastatur, bei der der erste Buchstabe für jeden neuen Satz großgeschrieben wird
textCapWords	Normale Texttastatur, die jedes Wort großschreibt. Gut für Titel oder Personennamen
textAutoCorrect	Normale Texttastatur, die häufig falsch geschriebene Wörter korrigiert

Sie können bei Bedarf mehrere `inputType` Attribute `inputType` (getrennt durch '|').
Beispiel:

```
<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" />
```

Eine Liste aller verfügbaren Eingabetypen finden Sie hier .

Attribut "Eingabetyp"

`inputype` Attribut in `EditText` Widget: *(getestet unter Android 4.4.3 und 2.3.3)*

```
<EditText android:id="@+id/et_test" android:inputType="?????"/>
```

textLongMessage = Tastatur: Alphabet / Standard. Eingabetaste: Senden / Weiter. Emotion: ja.

Fall: Kleinbuchstaben Vorschlag: ja. Hinzufügen. **verkohlt:, und.** und alles

textFilter = Tastatur: Alphabet / Standard. Eingabetaste: Senden / Weiter. Emotion: ja. Fall: Kleinbuchstaben **Vorschlag: nein** . Hinzufügen. **verkohlt:, und.** und alles

textCapWords = Tastatur: Alphabet / Standard. Eingabetaste: Senden / Weiter. Emotion: ja. **Fall: Kamel Fall** . Vorschlag: ja. Hinzufügen. **verkohlt:, und.** und alles

textCapSentences = Tastatur: Alphabet / Standard. Eingabetaste: Senden / Weiter. Emotion: ja. **Fall: Satzfall** . Vorschlag: ja. Hinzufügen. **verkohlt:, und.** und alles

Zeit = Tastatur: numerisch. Eingabetaste: Senden / Weiter. Emotion: nein. Fall: -. **Vorschlag: nein** . Hinzufügen. Zeichen::

textMultiLine = Tastatur: Alphabet / Standard. **Eingabetaste: nächste Zeile** . Emotion: ja. Fall: Kleinbuchstaben Vorschlag: ja. Hinzufügen. **verkohlt:, und.** und alles

number = Tastatur: numerisch . Eingabetaste: Senden / Weiter. Emotion: nein. Fall: -. Vorschlag: nein. **Hinzufügen. Zeichen: nichts**

textEmailAddress = Tastatur: Alphabet / Standard. Eingabetaste: Senden / Weiter. **Emotion: nein** . Fall: Kleinbuchstaben **Vorschlag: nein** . Hinzufügen. Zeichen: @ und **.** und alles

(Kein Typ) = Tastatur: Alphabet / Standard. **Eingabetaste: nächste Zeile** . Emotion: ja. Fall: Kleinbuchstaben Vorschlag: ja. Hinzufügen. **verkohlt:, und.** und alles

textPassword = Tastatur: Alphabet / Standard. Eingabetaste: Senden / Weiter. Emotion: nein. Fall: Kleinbuchstaben **Vorschlag: nein** . Hinzufügen. **verkohlt:, und.** und alles

Text = Tastatur: Tastatur: Alphabet / Standard. Eingabetaste: Senden / Weiter. Emotion: ja. Fall: Kleinbuchstaben Vorschlag: ja. Hinzufügen. **verkohlt:, und.** und alles

textShortMessage = Tastatur: Alphabet / Standard. **Eingabetaste: Emotion** . Emotion: ja. Fall: Kleinbuchstaben Vorschlag: ja. Hinzufügen. **verkohlt:, und.** und alles

textUri = Tastatur: Alphabet / Standard. Eingabetaste: Senden / Weiter. Emotion: nein. Fall: Kleinbuchstaben **Vorschlag: nein** . Hinzufügen. Zeichen: **/** und **.** und alles

textCapCharacters = Tastatur: Alphabet / Standard. Eingabetaste: Senden / Weiter. Emotion: ja. **Fall: GROSSBUCHSTABEN** . Vorschlag: ja. Hinzufügen. **verkohlt:, und.** und alles

Telefon = Tastatur: numerisch . Eingabetaste: Senden / Weiter. Emotion: nein. Fall: -. **Vorschlag: nein** . Hinzufügen. Zeichen: *** #. - / () WPN, + **

textPersonName = Tastatur: Alphabet / Standard. Eingabetaste: Senden / Weiter. Emotion: ja. Fall: Kleinbuchstaben Vorschlag: ja. Hinzufügen. **verkohlt:, und.** und alles

Hinweis: Die Einstellung für die `Auto-capitalization` ändert das Standardverhalten.

Hinweis 2: In der `Numeric keyboard` sind ALLE Zahlen Englisch 1234567890.

Hinweis 3: Die `Correction/Suggestion` ändert das Standardverhalten.

Softkeyboard ausblenden

Das Ausblenden des Softkeyboards ist normalerweise eine **Grundvoraussetzung,** wenn Sie mit EditText arbeiten. Das Softkeyboard kann *standardmäßig* nur durch Drücken der Zurück-Taste geschlossen werden. Daher verwenden die meisten Entwickler InputMethodManager , um Android dazu zu zwingen, die virtuelle Tastatur auszublenden, indem es hideSoftInputFromWindow aufruft und das Token des Fensters mit der fokussierten Ansicht übergibt. Der Code, um Folgendes zu tun:

```
public void hideSoftKeyboard()
{
        InputMethodManager inputMethodManager = (InputMethodManager)
getSystemService(Activity.INPUT_METHOD_SERVICE);
        inputMethodManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(), 0);
}
```

Der Code ist direkt, aber ein weiteres großes Problem ist, dass die Hide-Funktion aufgerufen werden muss, wenn ein Ereignis auftritt. Was ist zu tun, wenn das Softkeyboard beim Drücken an einer anderen Stelle als Ihrem EditText ausgeblendet werden muss? Der folgende Code enthält eine übersichtliche Funktion, die in Ihrer onCreate () - Methode nur einmal aufgerufen werden muss.

```
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);
            }
        }
}
```

Symbol oder Schaltfläche in Custom Edit Text und dessen Aktion und klicken auf Listener.

In diesem Beispiel wird der Text bearbeiten mit dem Symbol auf der rechten Seite angezeigt.

> Hinweis: In diesem Beispiel verwende ich setCompoundDrawablesWithIntrinsicBounds. Wenn Sie also die Symbolposition ändern möchten, können Sie dies mithilfe von setCompoundDrawablesWithIntrinsicBounds in setIcon erreichen.

```
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);
    }
}
```

Wenn Sie den Touch-Bereich ändern möchten, können Sie den Standardwert für EXTRA_TOUCH_AREA ändern, den ich als 50 angegeben habe.

Und für Aktivieren Sie den Button und klicken Sie auf Listener, den Sie von Ihrer Aktivität oder Ihrem Fragment aus aufrufen können.

```
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.

            }
        });
```

Text bearbeiten online lesen: https://riptutorial.com/de/android/topic/5843/text-bearbeiten

Kapitel 224: Text zu Sprache (TTS)

Examples

Text zur Sprachbasis

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);
        }
    }
}
```

Die Sprache, die gesprochen werden soll, kann durch `setLanguage()` eines `Locale` für die `setLanguage()` -Methode festgelegt werden:

```
tts.setLanguage(Locale.CHINESE); // Chinese language
```

Die Anzahl der unterstützten Sprachen variiert zwischen den Android-Stufen. Mit der Methode `isLanguageAvailable()` kann `isLanguageAvailable()` werden, ob eine bestimmte Sprache unterstützt wird:

```
tts.isLanguageAvailable(Locale.CHINESE);
```

Der `setPitch()` kann mit der Methode `setPitch()` werden. Standardmäßig ist der Tonhöhenwert 1,0. Verwenden Sie Werte unter 1,0, um die Tonhöhe zu verringern, oder Werte über 1,0, um die Tonhöhe zu erhöhen:

```
tts.setPitch(0.6);
```

Die Sprechgeschwindigkeit kann mit `setSpeechRate()` . Die voreingestellte Sprachrate beträgt 1,0. Die Sprachrate kann verdoppelt werden, indem sie auf 2,0 eingestellt wird, oder halbiert werden, indem sie auf 0,5 gesetzt wird:

```
tts.setSpeechRate(2.0);
```

TextToSpeech-Implementierung über die APIs

Kalte beobachtbare Implementierung, gibt True aus, wenn die TTS-Engine das Sprechen beendet hat, fängt an zu sprechen, wenn sie abonniert ist. Beachten Sie, dass die API-Ebene 21 eine andere Methode zum Sprechen einführt:

```
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;
}
```

}

Text zu Sprache (TTS) online lesen: https://riptutorial.com/de/android/topic/3381/text-zu-sprache--tts-

Kapitel 225: Textansicht automatisch anpassen

Einführung

Eine Textansicht, die den Text automatisch an seine Grenzen anpasst.

Mit Android O können Sie eine TextView anweisen, die Größe des Texts automatisch erweitern oder verkleinern zu lassen, um sein Layout basierend auf den Eigenschaften und Grenzen des TextView zu füllen.

Sie können die automatische Anpassung von TextView entweder in Code oder in XML einrichten.

Es gibt zwei Möglichkeiten, die automatische Größenänderung von TextView einzustellen: **Granularität** und **Vorgabegrößen**

Examples

Die Granularität

In Java:

Rufen Sie die `setAutoSizeTextTypeUniformWithConfiguration()` -Methode auf:

```
setAutoSizeTextTypeUniformWithConfiguration(int autoSizeMinTextSize, int autoSizeMaxTextSize,
int autoSizeStepGranularity, int unit)
```

In XML:

Verwenden Sie die `autoSizeMinTextSize` , `autoSizeMaxTextSize` und `autoSizeStepGranularity` , um die `autoSizeStepGranularity` für die automatische `autoSizeStepGranularity` in der XML-Layoutdatei `autoSizeStepGranularity` :

```
<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" />
```

Weitere Informationen finden Sie in der AutosizingTextViews-Demo bei GitHub.

Preset-Größen

In Java:

Rufen Sie die Methode `setAutoSizeTextTypeUniformWithPresetSizes()` :

```
setAutoSizeTextTypeUniformWithPresetSizes(int[] presetSizes, int unit)
```

In XML:

Verwenden Sie das `autoSizePresetSizes` Attribut in der XML-Layoutdatei:

```
<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" />
```

Um auf das Array als Ressource zuzugreifen, definieren Sie das Array in der Datei *res / values / arrays.xml* :

```
<array name="autosize_text_sizes">
    <item>10sp</item>
    <item>12sp</item>
    <item>20sp</item>
    <item>40sp</item>
    <item>100sp</item>
</array>
```

Weitere Informationen finden Sie in der AutosizingTextViews-Demo bei GitHub.

Textansicht automatisch anpassen online lesen:
https://riptutorial.com/de/android/topic/9652/textansicht-automatisch-anpassen

Kapitel 226: TextInputLayout

Einführung

TextInputLayout wurde eingeführt, um die Floating-Beschriftung in EditText anzuzeigen. Der EditText muss von TextInputLayout umschlossen werden, um die Floating-Bezeichnung anzuzeigen.

Bemerkungen

`TextInputLayout` ist ein Layout, das einen `EditText` (oder einen Nachkommen) `EditText` , um eine schwebende Beschriftung `EditText` wenn der Hinweis aufgrund der Eingabe von Text durch den Benutzer ausgeblendet wird. Zusätzlich ermöglicht das `TextInputLayout` die Anzeige einer Fehlermeldung unter dem `EditText` .

`build.gradle` Sie sicher, dass die folgende Abhängigkeit zur `build.gradle` Datei Ihrer App unter Abhängigkeiten hinzugefügt wird:

```
compile 'com.android.support:design:25.3.1'
```

Examples

Grundlegende Verwendung

`TextInputLayout` ist die grundlegende Verwendung des `TextInputLayout` .
`build.gradle` Sie sicher, dass Sie die Abhängigkeit in der Datei `build.gradle` , wie im Abschnitt "Anmerkungen" beschrieben.

Beispiel:

```
<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>
```

Fehler behandeln

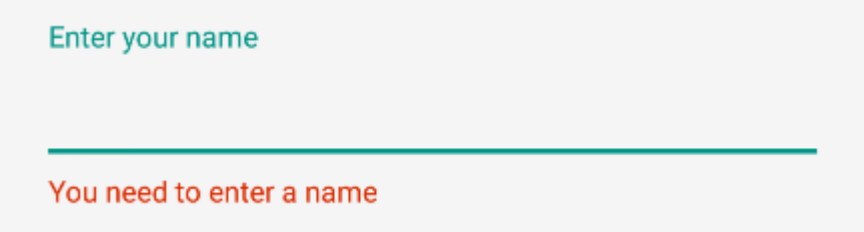
Mit dem `TextInputLayout` können Sie Fehlermeldungen gemäß den Richtlinien für das Materialdesign mit den Methoden `setError` und `setErrorEnabled` .

Um den Fehler unter dem EditText anzuzeigen, verwenden Sie:

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setErrorEnabled(true);
til.setError("You need to enter a name");
```

Um einen Fehler im `TextInputLayout` , können Sie entweder `app:errorEnabled="true"` in xml oder `til.setErrorEnabled(true);` wie oben gezeigt.

Sie erhalten:



Zeichenzählung hinzufügen

Der TextInputLayout hat einen Zeichenzähler für einen EditText darin definiert.
Der Zähler wird unter dem EditText dargestellt.

Verwenden `setCounterEnabled()` `setCounterMaxLength` Methoden `setCounterEnabled()` und `setCounterMaxLength` :

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setCounterEnabled(true);
til.setCounterMaxLength(15);
```

oder die `app:counterEnabled` und `app:counterMaxLength` Attribute in der XML- `app:counterMaxLength` .

```
<android.support.design.widget.TextInputLayout
    app:counterEnabled="true"
    app:counterMaxLength="15">

    <EditText/>

</android.support.design.widget.TextInputLayout>
```

Passwort-Sichtbarkeit wechselt

Mit einem Eingabepassworttyp können Sie auch ein Symbol aktivieren, das den gesamten Text mithilfe des Attributs `passwordToggleEnabled` anzeigen oder ausblenden kann .

Sie können denselben Standard auch mit diesen Attributen anpassen:

- `passwordToggleDrawable` : um das standardmäßige Augensymbol zu ändern
- `passwordToggleTint` : Um dem Zeichen für die Sichtbarkeit von Kennwörtern einen Farbton zuzuweisen, kann ein Zeichen gesetzt werden.

- `passwordToggleTintMode` : Zum Angeben des Mischmodus für die Hintergrundfarbe.

Beispiel:

```
<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

Der `TextInputEditText` ist ein `EditText` mit einem zusätzlichen Fix, um einen Hinweis im IME anzuzeigen, wenn sich der Modus "Extrahieren" befindet .

Der **Modus "Extrahieren"** ist der Modus, in den der Tastatureditor wechselt, wenn Sie auf einen EditText klicken, wenn der Platz zu klein ist (z. B. Querformat auf einem Smartphone). `EditText` Sie während der Bearbeitung des Textes einen `EditText` Sie in diesem Fall, dass der IME Ihnen keinen Hinweis darauf gibt, was Sie bearbeiten

Der `TextInputEditText` behebt dieses Problem, indem er einen `TextInputEditText` bereitstellt, während sich der IME des Benutzergeräts im Extraktionsmodus befindet.

Beispiel:

```
<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>
```

Anpassen der Darstellung des TextInputLayout

Sie können das Erscheinungsbild des `TextInputLayout` und des eingebetteten `EditText` indem Sie benutzerdefinierte Stile in Ihrer `styles.xml` . Die definierten Stile können entweder als Stile oder Designs zu Ihrem `TextInputLayout` .

Beispiel zum Anpassen des Hinweisaussehens:

`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>
```

Um den Style anzuwenden, aktualisieren Sie TextInputLayout und EditText wie folgt

```
<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>
```

Beispiel zum Anpassen der Akzentfarbe des `TextInputLayout` . Die Akzentfarbe beeinflusst die Farbe der Grundlinie des `EditText` und die `EditText` für den schwebenden `EditText` :

`styles.xml` :

```
<style name="TextInputLayoutWithPrimaryColor" parent="Widget.Design.TextInputLayout">
    <item name="colorAccent">@color/primary</item>
</style>
```

Layoutdatei:

```
<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>
```

TextInputLayout online lesen: https://riptutorial.com/de/android/topic/5652/textinputlayout

Kapitel 227: Textvorschau

Einführung

Alles im Zusammenhang mit der TextView-Anpassung in Android SDK

Syntax

- TextView (Kontextkontext)
- (TextView) findViewById (int id)
- void setText (int resid)
- void setText (CharSequence-Text) // Sie können String als Argument verwenden

Bemerkungen

Versuchen Sie es im XML-Design oder programmgesteuert zu verwenden.

Examples

Textansicht mit unterschiedlicher Textgröße

Sie können verschiedene Textgrößen innerhalb einer Textansicht mit einem Span archivieren

```
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-Anpassung

```
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);
        }
    }
}
```

Verwendungszweck:

```
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);
    }
}
```

Layout:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@mipmap/background">


    <pk.sohail.gallerytest.activity.CustomTextView
        android:id="@+id/pager_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:gravity="center"
        android:text="@string/txt_title_photo_gallery"
        android:textColor="@color/white"
        android:textSize="30dp"
        android:textStyle="bold"
        app:outerShadowRadius="10dp"
        app:strokeColor="@color/title_text_color"
        app:strokeJoinStyle="miter"
        app:strokeWidth="2dp" />

</RelativeLayout>
```

Attars:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <declare-styleable name="CustomTextView">

        <attr name="outerShadowRadius" format="dimension" />
        <attr name="strokeWidth" format="dimension" />
        <attr name="strokeMiter" format="dimension" />
        <attr name="strokeColor" format="color" />
        <attr name="strokeJoinStyle">
            <enum name="miter" value="0" />
            <enum name="bevel" value="1" />
            <enum name="round" value="2" />
        </attr>
    </declare-styleable>
```

```
</resources>
```

Programmgesteuerter Einsatz:

```
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");
```

Spannbare Textansicht

Eine spannbare `TextView` kann in Android verwendet werden, um einen bestimmten Textabschnitt mit einer anderen Farbe, einem anderen Stil, einer anderen Größe und / oder einem anderen Click-Ereignis in einem einzigen `TextView` Widget `TextView` .

`TextView` Sie, dass Sie eine `TextView` wie folgt definiert haben:

```
TextView textview=findViewById(R.id.textview);
```

Dann können Sie verschiedene Hervorhebungen anwenden, wie unten gezeigt:

- **Überbrückbare Farbe:** Um für einen bestimmten Textabschnitt eine andere Farbe `ForegroundColorSpan` kann ein `ForegroundColorSpan` verwendet werden, wie im folgenden Beispiel gezeigt:

  ```
  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 );
  ```

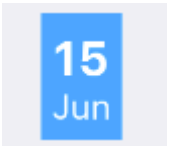
  Mit dem obigen Code erstellte Ausgabe:

- **Überbrückbare Schriftart:** Um eine andere Schriftgröße für einen bestimmten Textabschnitt **festzulegen** , kann ein `RelativeSizeSpan` verwendet werden, wie im folgenden Beispiel gezeigt:

  ```
  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 );
  ```

Mit dem obigen Code erstellte Ausgabe:



- **Überbrückbare Schrift:** Um eine andere Schriftart für einen bestimmten **Textbereich festzulegen** , kann ein benutzerdefiniertes `TypefaceSpan` verwendet werden, wie im folgenden Beispiel gezeigt:

```
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 );
```

Damit der obige Code funktioniert, muss jedoch die Klasse `CustomTypefaceSpan` von der Klasse `TypefaceSpan` . Dies kann wie folgt durchgeführt werden:

```
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);
        }
    }
```

Textansicht mit Bild

Android erlaubt Programmierern, Bilder an allen vier Ecken einer `TextView` . Wenn Sie beispielsweise ein Feld mit einer `TextView` und gleichzeitig `TextView` möchten, dass das Feld bearbeitet werden kann, platzieren Entwickler normalerweise ein Bearbeitungssymbol neben diesem Feld. Android bietet uns eine interessante Option, die für ein `TextView` als **gezeichnete** `TextView` :

```
<TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:drawablePadding="4dp"
        android:drawableRight="@drawable/edit"
        android:text="Hello world"
        android:textSize="18dp" />
```

Sie können das Zeichen auf eine beliebige Seite Ihrer `TextView` wie folgt `TextView` :

```
android:drawableLeft="@drawable/edit"
android:drawableRight="@drawable/edit"
android:drawableTop="@drawable/edit"
android:drawableBottom="@drawable/edit"
```

Das Zeichnen kann auch auf folgende Weise programmgesteuert erfolgen:

```
yourTextView.setCompoundDrawables(leftDrawable, rightDrawable, topDrawable, bottomDrawable);
```

Wenn Sie einen der an `setCompoundDrawables()` übergebenen Parameter auf `null` wird das Symbol von der entsprechenden Seite der `TextView` .

Durchgestrichene Textansicht

Durchgestrichen durch den gesamten Text

```
String sampleText = "This is a test strike";
textView.setPaintFlags(tv.getPaintFlags()| Paint.STRIKE_THRU_TEXT_FLAG);
textView.setText(sampleText);
```

Ausgabe: ~~Dies ist ein Testschlag~~

Durchgestrichen nur Teile des Textes

```
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);
```

Ausgabe: ~~Dies~~ ist ein Testschlag

Anpassung von Design und Stil

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>
```

Stellen Sie RelativeSizeSpan nach oben ausrichten

Um ein `RelativeSizeSpan` nach oben ausrichten zu lassen, kann eine benutzerdefinierte Klasse von der Klasse `SuperscriptSpan` . Im folgenden Beispiel heißt die abgeleitete Klasse `TopAlignSuperscriptSpan` :

activity_main.xml:

```
<TextView
    android:id="@+id/txtView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:textSize="26sp" />
```

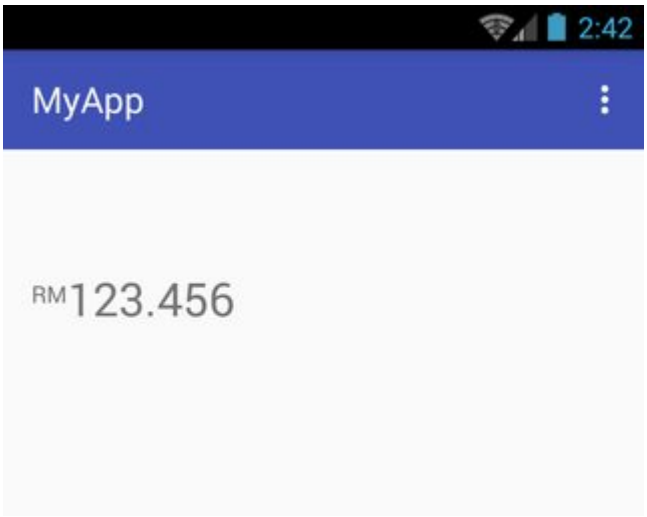
MainActivity.java:

```
TextView txtView = (TextView) findViewById(R.id.txtView);

SpannableString spannableString = new SpannableString("RM123.456");
spannableString.setSpan( new TopAlignSuperscriptSpan( (float)0.35 ), 0, 2,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE );
txtView.setText(spannableString);
```

TopAlignSuperscriptSpan.java:

```
private class TopAlignSuperscriptSpan extends SuperscriptSpan {
    //divide superscript by this number
    protected int fontScale = 2;

    //shift value, 0 to 1.0
    protected float shiftPercentage = 0;

    //doesn't shift
    TopAlignSuperscriptSpan() {}

    //sets the shift percentage
    TopAlignSuperscriptSpan( float shiftPercentage ) {
        if( shiftPercentage > 0.0 && shiftPercentage < 1.0 )
            this.shiftPercentage = shiftPercentage;
    }

    @Override
    public void updateDrawState( TextPaint tp ) {
        //original ascent
        float ascent = tp.ascent();

        //scale down the font
        tp.setTextSize( tp.getTextSize() / fontScale );

        //get the new font ascent
        float newAscent = tp.getFontMetrics().ascent;

        //move baseline to top of old font, then move down size of new font
        //adjust for errors with shift percentage
        tp.baselineShift += ( ascent - ascent * shiftPercentage )
                - (newAscent - newAscent * shiftPercentage );
    }

    @Override
    public void updateMeasureState( TextPaint tp ) {
        updateDrawState( tp );
    }
}
```

Referenz-Screenshot:



Pinchzoom auf TextView

activity_main.xml :

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

    <TextView
        android:id="@+id/mytv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="This is my sample text for pinch zoom demo, you can zoom in and out
using pinch zoom, thanks" />

</RelativeLayout>
```

MainActivity.java :

```
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.TextView;

public class MyTextViewPinchZoomClass extends Activity implements OnTouchListener {

    final static float STEP = 200;
    TextView mytv;
    float mRatio = 1.0f;
    int mBaseDist;
    float mBaseRatio;
    float fontsize = 13;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        mytv = (TextView) findViewById(R.id.mytv);
        mytv.setTextSize(mRatio + 13);
    }

    public boolean onTouchEvent(MotionEvent event) {
        if (event.getPointerCount() == 2) {
            int action = event.getAction();
            int pureaction = action & MotionEvent.ACTION_MASK;
            if (pureaction == MotionEvent.ACTION_POINTER_DOWN) {
                mBaseDist = getDistance(event);
                mBaseRatio = mRatio;
            } else {
                float delta = (getDistance(event) - mBaseDist) / STEP;
                float multi = (float) Math.pow(2, delta);
                mRatio = Math.min(1024.0f, Math.max(0.1f, mBaseRatio * multi));
                mytv.setTextSize(mRatio + 13);
            }
        }
        return true;
    }

    int getDistance(MotionEvent event) {
        int dx = (int) (event.getX(0) - event.getX(1));
        int dy = (int) (event.getY(0) - event.getY(1));
        return (int) (Math.sqrt(dx * dx + dy * dy));
    }

    public boolean onTouch(View v, MotionEvent event) {
        return false;
    }
}
```

Einzelne Textansicht mit zwei verschiedenen Farben

Farbiger Text kann erstellt werden, indem der Text und der Name der Schriftfarbe an folgende Funktion übergeben werden:

```
private String getColoredSpanned(String text, String color) {
    String input = "<font color=" + color + ">" + text + "</font>";
    return input;
}
```

Der farbige Text kann dann auf eine gesetzt werden `TextView` (oder sogar auf einen `Button` , `EditText` , etc.) durch den Beispielcode unten.

Definieren Sie zunächst eine `TextView` wie folgt:

```
TextView txtView = (TextView)findViewById(R.id.txtView);
```

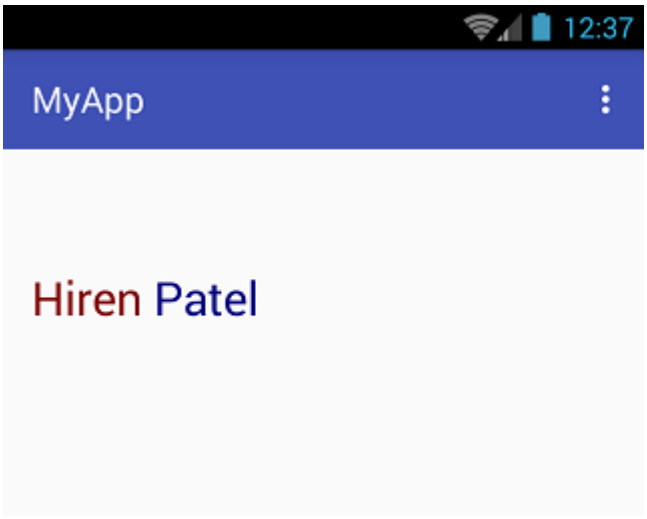
Erstellen Sie dann andersfarbigen Text und weisen Sie ihn den Zeichenfolgen zu:

```
String name = getColoredSpanned("Hiren", "#800000");
String surName = getColoredSpanned("Patel","#000080");
```

Setzen Sie zum Schluss die zwei verschiedenfarbigen Zeichenfolgen auf die `TextView` :

```
txtView.setText(Html.fromHtml(name+" "+surName));
```

Referenz-Screenshot:



Textvorschau online lesen: https://riptutorial.com/de/android/topic/4212/textvorschau

Kapitel 228: Thema, Stil, Attribut

Examples

Benutzerdefiniertes Design global verwenden

In themes.xml:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

In AndroidManifest.xml:

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">

    <!-- Activity declarations here -->

</application>
```

Definieren Sie primäre, primäre dunkle und Akzentfarben

Sie können die Farbpalette Ihres Themas anpassen.

Framework- APIs verwenden

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>
```

Verwenden der **Appcompat-Unterstützungsbibliothek** (und `AppCompatActivity`)

2.1.x

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="colorPrimary">@color/primary</item>
    <item name="colorPrimaryDark">@color/primary_dark</item>
    <item name="colorAccent">@color/accent</item>
</style>
```

Verwenden Sie ein benutzerdefiniertes Thema pro Aktivität

In themes.xml:

```
<style name="MyActivityTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

In AndroidManifest.xml:

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/Theme.AppCompat">

    <activity
        android:name=".MyActivity"
        android:theme="@style/MyActivityTheme" />

</application>
```

Overscroll-Farbe (API 21+)

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorEdgeEffect">@color/my_color</item>
</style>
```

Wellenfarbe (API 21+)

5,0

Die Welligkeitsanimation wird angezeigt, wenn der Benutzer auf anklickbare Ansichten drückt.

Sie können dieselbe Ripple-Farbe verwenden, die von Ihrer App verwendet wird, indem `?android:colorControlHighlight` in Ihren Ansichten das `?android:colorControlHighlight` . Sie können diese Farbe anpassen, indem Sie das Attribut `android:colorControlHighlight` in Ihrem `android:colorControlHighlight` ändern:

Diese Effektfarbe kann geändert werden:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorControlHighlight">@color/my_color</item>
</style>
```

Oder, wenn Sie ein Materialdesign verwenden:

```
<style name="AppTheme" parent="android:Theme.Material.Light">
    <item name="android:colorControlHighlight">@color/your_custom_color</item>
</style>
```

Lichtstatusleiste (API 23+)

Dieses Attribut kann den Hintergrund der Statussymbole (oben auf dem Bildschirm) in Weiß ändern.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowLightStatusBar">true</item>
</style>
```

Transluzente Navigations- und Statusleisten (API 19+)

Die Navigationsleiste (am unteren Rand des Bildschirms) kann transparent sein. Hier ist der Weg, um es zu erreichen.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentNavigation">true</item>
</style>
```

Die Statusleiste (oben auf dem Bildschirm) kann transparent gemacht werden, indem dieses Attribut auf den Stil angewendet wird:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentStatus">true</item>
</style>
```

Farbe der Navigationsleiste (API 21+)

5,0

Dieses Attribut wird verwendet, um die Navigationsleiste zu ändern (eine, die die Schaltfläche Zurück, Startseite zuletzt) enthält. Normalerweise ist es schwarz, die Farbe kann jedoch geändert werden.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:navigationBarColor">@color/my_color</item>
</style>
```

Thema Vererbung

Bei der Definition von Themen verwendet man normalerweise das vom System bereitgestellte Thema und ändert dann das Erscheinungsbild an seine eigene Anwendung. So wird beispielsweise das `Theme.AppCompat` vererbt:

```
<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>
```

Dieses `Theme.AppCompat` verfügt jetzt über alle Eigenschaften des Standarddesigns `Theme.AppCompat` , außer denjenigen, die wir explizit geändert haben.

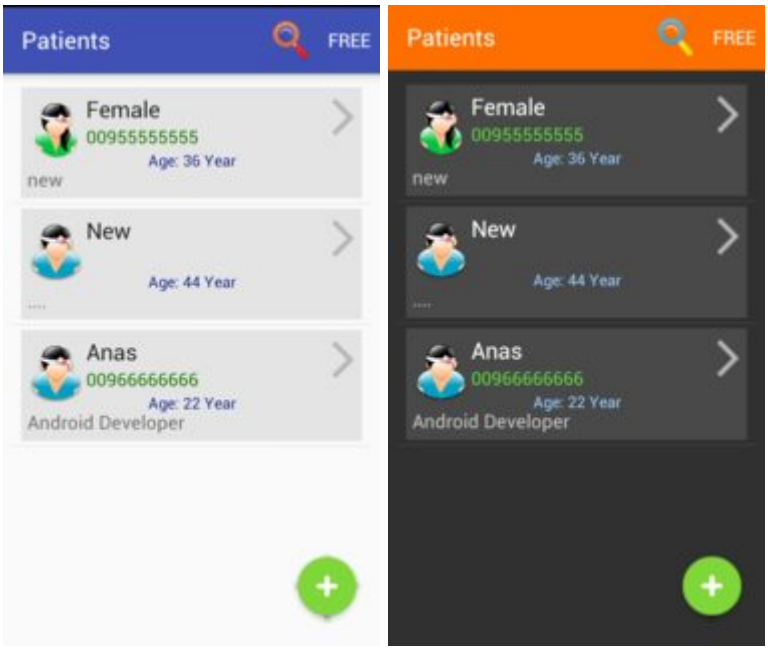
Es gibt auch eine Abkürzung beim Erben, die normalerweise verwendet wird, wenn man von seinem eigenen Thema erbt:

```
<style name="AppTheme.Red">
    <item name="colorAccent">@color/red</item>
</style>
```

Da hat es bereits `AppTheme.` Am Anfang wird der Name automatisch erben, ohne dass das `parent` Design definiert werden muss. Dies ist nützlich, wenn Sie bestimmte Stile für einen Teil (z. B. eine einzelne Aktivität) Ihrer App erstellen müssen.

Mehrere Themes in einer App

Wenn Sie mehr als ein Thema in Ihrer Android-Anwendung verwenden, können Sie jedem Thema eigene Farben hinzufügen.



Zuerst müssen wir unsere Designs wie `style.xml` zu `style.xml` hinzufügen:

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">

</style>

<!--  -->
<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >

</style>
......
```

Oben sehen Sie **OneTheme** und **TwoTheme** .

Nun gehen Sie zu Ihrem `AndroidManifest.xml` und fügen Sie diese Zeile: `android:theme="@style/OneTheme"` auf Ihre *Bewerbung* Tag, wird dies das Standard - Theme machen **OneTheme:**

```
<application
        android:theme="@style/OneTheme"
        ...>
```

Erstellen Sie eine neue XML-Datei namens `attrs.xml` und fügen Sie diesen Code hinzu:

```
<?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) -->
```

Gehen Sie zurück zu `style.xml` und fügen Sie diese Farben mit den Werten für jedes Thema hinzu:

```
<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>
```

Jetzt haben Sie benutzerdefinierte Farben für jedes Thema. Lassen Sie uns diese Farben in unseren Ansichten einfügen.

Fügen Sie der **TextView-** Farbe mit "? Attr /" die Farbe custom_blue hinzu:

Gehen Sie zu Ihrem imageView und fügen Sie diese Farbe hinzu:

```
<TextView>
    android:id="@+id/txte_view"
    android:textColor="?attr/custom_blue" />
```

Mow, wir können das Thema nur durch eine einzelne Zeile `setTheme(R.style.TwoTheme);` Diese Zeile muss vor der `setContentView()` -Methode in der `onCreate()` -Methode wie dieser `Activity.java` :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setTheme(R.style.TwoTheme);
    setContentView(R.layout.main_activity);
    ....
}
```

Ändern Sie das Thema für alle Aktivitäten gleichzeitig

Wenn Sie das Thema für alle Aktivitäten ändern möchten, müssen Sie eine neue Klasse mit dem Namen MyActivity erstellen, die die `AppCompatActivity` Klasse (oder `Activity` Klasse) erweitert und die Zeile `setTheme(R.style.TwoTheme);` hinzufügt `setTheme(R.style.TwoTheme);` zu **onCreate () -** Methode:

```
public class MyActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (new MySettings(this).isDarkTheme())
            setTheme(R.style.TwoTheme);
    }
}
```

Gehen Sie schließlich zu all Ihren Aktivitäten und fügen Sie alle dazu aus, die **MyActivity-** Basisklasse zu erweitern:

```
public class MainActivity extends MyActivity {
    ....
}
```

Um das Thema zu ändern, wechseln Sie einfach zu **MyActivity** und ändern Sie `R.style.TwoTheme` in Ihr Thema (`R.style.OneTheme` , `R.style.ThreeTheme` ....).

Thema, Stil, Attribut online lesen: https://riptutorial.com/de/android/topic/1843/thema--stil--attribut

Kapitel 229: Toast

Einführung

Ein Toast bietet ein einfaches Feedback zu einer Operation in einem kleinen Popup und verschwindet nach einem Timeout automatisch. Es füllt nur den für die Nachricht erforderlichen Platz aus und die aktuelle Aktivität bleibt sichtbar und interaktiv.

Syntax

- Toast makeText (Kontextkontext, CharSequence-Text, int duration)
- Toast makeText (Kontextkontext, int resId, int duration)
- void setGravity (int Schwerkraft, int xOffset, int yOffset)
- void show ()

Parameter

Parameter	Einzelheiten
Kontext	Der Kontext, in dem Ihr Toast angezeigt werden soll. `this` wird üblicherweise in einer Aktivität verwendet und `getActivity()` in einem Fragment
Text	Eine Zeichenfolge, die angibt, welcher Text im Toast angezeigt wird. Jedes Objekt, das CharSequence implementiert, kann verwendet werden, einschließlich einer Zeichenfolge
resId	Eine Ressourcen-ID, mit der eine Ressourcenzeichenfolge zur Anzeige im Toast bereitgestellt werden kann
Dauer	Integer-Flag, das angibt, wie lange der Toast angezeigt wird. Optionen sind `Toast.LENGTH_SHORT` und `Toast.LENGTH_LONG`
Schwere	Ganzzahl, die die Position oder "Schwerkraft" des Toasts angibt. Siehe Optionen hier
xOffset	Bestimmt den horizontalen Versatz für die Toast-Position
yOffset	Legt den vertikalen Versatz für die Toast-Position fest

Bemerkungen

Ein Toast bietet ein einfaches Feedback zu einer Operation in einem kleinen Popup. Es füllt nur den für die Nachricht erforderlichen Platz aus und die aktuelle Aktivität bleibt sichtbar und interaktiv.

Eine neuere Alternative zu Toast ist SnackBar. SnackBar bietet einen aktualisierten visuellen Stil und ermöglicht es dem Benutzer, die Nachricht zu verwerfen oder weitere Maßnahmen zu ergreifen. Einzelheiten finden Sie in der SnackBar- Dokumentation.

Offizielle Dokumentation:

https://developer.android.com/reference/android/widget/Toast.html

Examples

Position eines Toast einstellen

Eine standardmäßige Toastbenachrichtigung wird unten im Bildschirm in horizontaler Mitte angezeigt. Diese Position können Sie mit `setGravity(int, int, int)` ändern. Dies akzeptiert drei Parameter: eine Schwerkraftkonstante, einen X-Positionsversatz und einen Y-Positionsversatz.

Wenn Sie beispielsweise entscheiden, dass der Toast in der oberen linken Ecke angezeigt werden soll, können Sie die Schwerkraft wie folgt einstellen:

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

Anzeige einer Toast-Nachricht

In Android ist ein Toast ein einfaches UI-Element, das verwendet werden kann, um einem Benutzer kontextabhängiges Feedback zu geben.

Um eine einfache Toast-Nachricht anzuzeigen, können wir Folgendes tun.

```
// 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();
```

Oder, um einen Toast-Inline-Eintrag anzuzeigen, ohne sich an dem Toast-Objekt festzuhalten, können Sie:

```
Toast.makeText(context, "Ding! Your Toast is ready.", Toast.LENGTH_SHORT).show();
```

WICHTIG: Stellen Sie sicher, dass die `show()` -Methode vom UI-Thread aufgerufen wird. Wenn Sie versuchen, einen `Toast` aus einem anderen Thread `runOnUiThread` können Sie z. B. die `runOnUiThread` Methode einer `Activity` .

Andernfalls, dh der Versuch, die Benutzeroberfläche durch Erstellen eines Toast zu ändern, `RuntimeException` eine `RuntimeException` die wie folgt aussieht:

```
java.lang.RuntimeException: Can't create handler inside thread that has not called
Looper.prepare()
```

Die einfachste Möglichkeit, diese Ausnahme zu behandeln, ist die Verwendung von runOnUiThread: Die Syntax wird unten gezeigt.

```
   runOnUiThread(new Runnable() {
        @Override
        public void run() {
            // Your code here
        }
    });
```

Einen benutzerdefinierten Toast erstellen

Wenn Sie die standardmäßige Toast-Ansicht nicht verwenden möchten, können Sie mit der `setView(View)` -Methode für ein `Toast` Objekt eine eigene `setView(View)` .

Erstellen Sie zunächst das XML-Layout, das Sie in Ihrem Toast verwenden möchten.

```
<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>
```

Wenn Sie dann Ihren Toast erstellen, blasen Sie Ihre benutzerdefinierte Ansicht aus XML auf und rufen Sie `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();
```

Thread-sichere Anzeige von Toast (Application Wide)

```
public class MainApplication extends Application {

    private static Context context; //application context

    private Handler mainThreadHandler;
    private Toast toast;

    public Handler getMainThreadHandler() {
        if (mainThreadHandler == null) {
            mainThreadHandler = new Handler(Looper.getMainLooper());
        }
        return mainThreadHandler;
    }

    @Override public void onCreate() {
        super.onCreate();
        context = this;
    }

    public static MainApplication getApp(){
        return (MainApplication) context;
    }

    /**
     * Thread safe way of displaying toast.
     * @param message
     * @param duration
     */
    public void showToast(final String message, final int duration) {
        getMainThreadHandler().post(new Runnable() {
            @Override
            public void run() {
                if (!TextUtils.isEmpty(message)) {
                    if (toast != null) {
                        toast.cancel(); //dismiss current toast if visible
                        toast.setText(message);
                    } else {
                        toast = Toast.makeText(App.this, message, duration);
                    }
                    toast.show();
                }
            }
        });
    }
```

Denken Sie daran, `MainApplication` im `manifest` hinzuzufügen.

Rufen Sie es jetzt in einem beliebigen Thread auf, um eine Toastnachricht anzuzeigen.

```
MainApplication.getApp().showToast("Some message", Toast.LENGTH_LONG);
```

Toastmeldung über der Soft-Tastatur anzeigen

Standardmäßig zeigt Android Toast-Meldungen am unteren Bildschirmrand an, auch wenn die Tastatur angezeigt wird. Dies zeigt eine Toast-Nachricht direkt über der Tastatur.

```
public void showMessage(final String message, final int length) {
    View root = findViewById(android.R.id.content);
    Toast toast = Toast.makeText(this, message, length);
    int yOffset = Math.max(0, root.getHeight() - toast.getYOffset());
    toast.setGravity(Gravity.TOP | Gravity.CENTER_HORIZONTAL, 0, yOffset);
    toast.show();
}
```

Thread-sichere Methode zum Anzeigen einer Toast-Nachricht (für AsyncTask)

Wenn Sie Application nicht erweitern möchten und den Toastnachrichten-Thread sicher aufbewahren möchten, stellen Sie sicher, dass Sie sie im Abschnitt "Nach der Ausführung" Ihrer AsyncTasks anzeigen.

```
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();
    }

}
```

Toast online lesen: https://riptutorial.com/de/android/topic/1741/toast

Kapitel 230: TransitionDrawable

Examples

Fügen Sie einen Übergang oder ein Überblenden zwischen zwei Bildern hinzu.

Schritt 1: Erstellen Sie einen Übergang, der in XML gezeichnet werden kann

Speichern Sie diese Datei `transition.xml` im Ordner `res/drawable` Ihres Projekts.

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/image1"/>
    <item android:drawable="@drawable/image2"/>
</transition>
```

Image1 und image2 sind die beiden Bilder, die wir umstellen möchten, und sie sollten sich ebenfalls in Ihrem `res/drawable` Ordner befinden.

Schritt 2: Fügen Sie Code für ImageView in Ihr XML-Layout ein, um das obige Zeichenelement anzuzeigen.

```
<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>
```

Schritt 3: Greifen Sie auf die drawable XML-Transition in der Methode onCreate () Ihrer Activity zu und starten Sie die Transition im Ereignis onClick ().

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

```
    imageView = (ImageView) findViewById(R.id.image_view);
    transitionDrawable = (TransitionDrawable)
        ContextCompat.getDrawable(this, R.drawable.transition);

    birdImageView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(final View view) {
            birdImageView.setImageDrawable(transitionDrawable);
            transitionDrawable.startTransition(1000);
        }
    });
}
```

Animieren Sie die Hintergrundfarbe der Ansichten (Umschaltfarbe) mit TransitionDrawable

```
public void setCardColorTran(View view) {
   ColorDrawable[] color = {new ColorDrawable(Color.BLUE), new ColorDrawable(Color.RED)};
   TransitionDrawable trans = new TransitionDrawable(color);
    if(Build.VERSION.SDK_INT < android.os.Build.VERSION_CODES.JELLY_BEAN) {
        view.setBackgroundDrawable(trans);
        }else {
        view.setBackground(trans);
    }
    trans.startTransition(5000);
}
```

TransitionDrawable online lesen: https://riptutorial.com/de/android/topic/6088/transitiondrawable

Kapitel 231: Twitter-APIs

Examples

Login mit Twitter-Button erstellen und Rückruf anhängen

1. Fügen Sie in Ihrem Layout eine Login-Schaltfläche mit dem folgenden Code hinzu:

```
<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. In der Aktivität oder im Fragment, in der die Schaltfläche angezeigt wird, müssen Sie einen Rückruf an die Anmeldeschaltfläche erstellen und anhängen.

```
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. Übergeben Sie das Ergebnis der Authentifizierungsaktivität wieder an die Schaltfläche:

```
@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);
}
```

Hinweis: Wenn Sie den TwitterLoginButton in einem Fragment verwenden, führen Sie

stattdessen die folgenden Schritte aus:

```
 @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. Fügen Sie den **build.gradle-** Abhängigkeiten die folgenden Zeilen **hinzu** :

```
 apply plugin: 'io.fabric'

 repositories {
     maven { url 'https://maven.fabric.io/public' }
 }

 compile('com.twitter.sdk.android:twitter:1.14.1@aar') {
     transitive = true;
 }
```

Twitter-APIs online lesen: https://riptutorial.com/de/android/topic/4801/twitter-apis

Kapitel 232: Typedef-Anmerkungen: @IntDef, @StringDef

Bemerkungen

Das Annotations-Paket enthält eine Reihe nützlicher Metadaten-Annotationen, mit denen Sie Ihren eigenen Code dekorieren können, um Fehler zu finden.

`build.gradle` Sie die Abhängigkeit `build.gradle` in die Datei `build.gradle` .

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

Examples

IntDef-Anmerkungen

Diese Annotation stellt sicher, dass nur die gültigen Integer-Konstanten verwendet werden, die Sie erwarten.
Das folgende Beispiel veranschaulicht die Schritte zum Erstellen einer Anmerkung:

```
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;
    }
}
```

Sie ermöglichen auch die Codeabwicklung, um die zulässigen Konstanten automatisch anzubieten.
Wenn Sie diesen Code erstellen, wird eine Warnung generiert, wenn der Typparameter nicht auf eine der definierten Konstanten verweist.

Konstanten mit Flags kombinieren

Wenn das `IntDef#flag()` auf `true` , können mehrere Konstanten kombiniert werden.

Verwenden Sie dasselbe Beispiel in diesem Thema:

```
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)

    .....

}
```

Benutzer können die zulässigen Konstanten mit einem Flag kombinieren (z. B. `|` , `&` , `^`).

Typedef-Anmerkungen: @IntDef, @StringDef online lesen:
https://riptutorial.com/de/android/topic/4505/typedef-anmerkungen---intdef---stringdef

Kapitel 233: Überholspur

Bemerkungen

fastlane ist ein Tool für iOS-, Mac- und Android-Entwickler, mit dem Sie mühsame Aufgaben wie das **Erstellen von** Screenshots, das Bereitstellen von Profilen und das Freigeben Ihrer Anwendung automatisieren können.

Dokumente: https://docs.fastlane.tools/

Quellcode: https://github.com/fastlane/fastlane

Examples

Fastfile zum Erstellen und Hochladen mehrerer Versionen von Beta von Crashlytics

Dies ist ein Beispiel für ein **Fastfile-** Setup für eine App mit mehreren Geschmacksrichtungen. Sie haben die Möglichkeit, alle Geschmacksrichtungen oder eine einzelne Variante zu erstellen und bereitzustellen. Nach der Bereitstellung meldet es **Slack** den Status der Bereitstellung und sendet eine Benachrichtigung an die Tester in der Beta-Gruppe von Crashlytics-Testern.

Um alle Varianten zu erstellen und bereitzustellen, verwenden Sie:

```
fastlane android beta
```

Um ein einzelnes APK zu erstellen und bereitzustellen, verwenden Sie Folgendes:

```
fastlane android beta app:flavorName
```

Mit einer einzigen Fastlane-Datei können Sie iOS-, Android- und Mac-Apps verwalten. Wenn Sie diese Datei nur für eine App- `platform` ist dies nicht erforderlich.

Wie es funktioniert

1. `android` Argument sagt fastlane, dass wir verwenden werden `:android` Plattform.
2. Innerhalb `:android` Plattform können Sie mehrere Bahnen haben. Derzeit habe ich nur `:beta` Spur. Das zweite Argument des obigen Befehls gibt die Spur an, die wir verwenden möchten.
3. `options[:app]`
4. Es gibt zwei **Gradle-** Aufgaben. Erstens läuft es `gradle clean` . Wenn Sie einen `app` Schlüssel eingegeben haben, wird FastFile `gradle assembleReleaseFlavor` . Andernfalls wird `gradle assembleRelease` , um alle Build-Geschmacksrichtungen zu erstellen.
5. Wenn Sie für alle Flavours erstellen, wird ein Array von generierten APK-Dateinamen in `SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS` gespeichert. Wir verwenden dies, um die

erzeugten Dateien **durchzugehen** und sie **von Crashlytics** für **Beta bereitzustellen** . `notifications` und `groups` Felder sind optional. Sie werden verwendet, um **von Crashlytics** für **Beta** registrierte Tester zu benachrichtigen.

6. Wenn Sie mit Crashlytics vertraut sind, wissen Sie möglicherweise, dass Sie zum Aktivieren einer App im Portal sie auf einem Gerät ausführen und zuerst verwenden müssen. Andernfalls geht Crashlytics davon aus, dass die App inaktiv ist, und gibt einen Fehler aus. In diesem Szenario fange ich es auf und melde mich als fehlgeschlagen an **Slack** , sodass Sie wissen, welche App inaktiv ist.
7. Wenn die Bereitstellung erfolgreich ist, **sendet fastlane** eine Erfolgsmeldung an **Slack** .
8. `#{/([^\/]*)$/.match(apk)}` Dieser Regex wird verwendet, um den Flavour-Namen aus dem APK-Pfad zu erhalten. Sie können es entfernen, wenn es für Sie nicht funktioniert.
9. `get_version_name` und `get_version_code` sind zwei **Fastlane-** Plugins, um den Namen und Code der App-Version abzurufen. Sie müssen diese Edelsteine installieren, wenn Sie sie verwenden möchten, oder Sie können sie entfernen. Lesen Sie hier mehr über Plugins.
10. Die `else` Anweisung wird ausgeführt, wenn Sie einen einzelnen APK erstellen und bereitstellen. Wir müssen keinen `apk_path` für Crashlytics bereitstellen, da wir nur eine App haben.
11. `error do` Block am Ende wird verwendet, um benachrichtigt zu werden, wenn während der Ausführung etwas anderes schief geht.

Hinweis

Vergessen Sie nicht, `SLACK_URL` , `API_TOKEN` , `GROUP_NAME` und `BUILD_SECRET` durch Ihre eigenen Anmeldeinformationen zu ersetzen.

```
fastlane_version "1.46.1"

default_platform :android

platform :android do

    before_all do
        ENV["SLACK_URL"] = "https://hooks.slack.com/servic...."
    end

    lane :beta do |options|
        # Clean and build the Release version of the app.
        # Usage `fastlane android beta app:flavorName`

        gradle(task: "clean")

        gradle(task: "assemble",
               build_type: "Release",
               flavor: options[:app])

        # If user calls `fastlane android beta` command, it will build all projects and push
them to Crashlytics
        if options[:app].nil?
            lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].each do | apk |

                puts "Uploading APK to Crashlytics: " + apk

                begin
                    crashlytics(
```

```
                    api_token: "[API_TOKEN]",
                    build_secret: "[BUILD_SECRET]",
                    groups: "[GROUP_NAME]",
                    apk_path: apk,
                    notifications: "true"
                  )

                  slack(
                    message: "Successfully deployed new build for #{/([^\/]*)$/.match(apk)}
#{get_version_name} - #{get_version_code}",
                    success: true,
                    default_payloads: [:git_branch, :lane, :test_result]
                  )
              rescue => ex
                  # If the app is inactive in Crashlytics, deployment will fail. Handle it
here and report to slack
                  slack(
                      message: "Error uploading => #{/([^\/]*)$/.match(apk)}
#{get_version_name} - #{get_version_code}: #{ex}",
                      success: false,
                      default_payloads: [:git_branch, :lane, :test_result]
                  )
              end
          end

          after_all do |lane|
              # This block is called, only if the executed lane was successful
              slack(
                  message: "Operation completed for
#{lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].size} app(s) for #{get_version_name}
- #{get_version_code}",
                  default_payloads: [:git_branch, :lane, :test_result],
                  success: true
              )
          end
      else
          # Single APK upload to Beta by Crashlytics
          crashlytics(
              api_token: "[API_TOKEN]",
              build_secret: "[BUILD_SECRET]",
              groups: "[GROUP_NAME]",
              notifications: "true"
          )

          after_all do |lane|
              # This block is called, only if the executed lane was successful
              slack(
                  message: "Successfully deployed new build for #{options[:app]}
#{get_version_name} - #{get_version_code}",
                  default_payloads: [:git_branch, :lane, :test_result],
                  success: true
              )
          end
      end

      error do |lane, exception|
          slack(
              message: exception.message,
              success: false,
              default_payloads: [:git_branch, :lane, :test_result]
          )
```

```
        end
    end
end
```

Fastfile-Spur zum Erstellen und Installieren aller Flavors für einen bestimmten Build-Typ auf einem Gerät

Fügen Sie diese Spur Ihrem **Fastfile hinzu** und führen Sie `fastlane installAll type:{BUILD_TYPE}` in der Befehlszeile. Ersetzen Sie `BUILD_TYPE` durch den Build-Typ, den Sie erstellen möchten.

Zum Beispiel: `fastlane installAll type:Debug`

Dieser Befehl erstellt alle Varianten des angegebenen Typs und installiert sie auf Ihrem Gerät. Derzeit funktioniert es nicht, wenn Sie mehr als ein Gerät angeschlossen haben. Stellen Sie sicher, dass Sie nur einen haben. In Zukunft plane ich, eine Option zur Auswahl des Zielgeräts hinzuzufügen.

```
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
```

Überholspur online lesen: https://riptutorial.com/de/android/topic/8215/uberholspur

Kapitel 234: Überprüfen Sie die Datenverbindung

Examples

Datenverbindung überprüfen

Diese Methode dient zum Überprüfen der Datenverbindung durch Ping bestimmter IP- oder Domänennamen.

```
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;
}
```

Überprüfen Sie die Verbindung mit ConnectivityManager

```
    public static boolean isConnectedNetwork (Context context) {

        ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
        return cm.getActiveNetworkInfo () != null && cm.getActiveNetworkInfo
().isConnectedOrConnecting ();

    }
```

Verwenden Sie die Netzwerkeinstellungen, um Aufgaben auszuführen, solange Daten zulässig sind

Wenn Ihr Gerät eine Verbindung zu einem Netzwerk herstellt, wird eine Absicht gesendet. Viele Apps prüfen nicht nach diesen Absichten, aber damit Ihre Anwendung ordnungsgemäß funktioniert, können Sie die Absichten der Netzwerkänderung abhören, die Sie darüber informieren, wann Kommunikation möglich ist. Um die Netzwerkverbindung zu überprüfen, können Sie beispielsweise die folgende Klausel verwenden:

```
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
```

```
}
```

Überprüfen Sie die Datenverbindung online lesen:
https://riptutorial.com/de/android/topic/8670/uberprufen-sie-die-datenverbindung

Kapitel 235: Überprüfen Sie die Internetverbindung

Einführung

Diese Methode wird verwendet, um zu prüfen, ob WI-Fi angeschlossen ist oder nicht.

Syntax

- isNetworkAvailable (): Um zu überprüfen, ob Internet auf dem Gerät verfügbar ist

Parameter

Parameter	Detail
Kontext	Eine Referenz des Aktivitätskontexts

Bemerkungen

Wenn eine Internetverbindung besteht, wird die Methode wahr oder falsch zurückgegeben.

Examples

Überprüfen Sie, ob das Gerät über eine Internetverbindung verfügt

Fügen Sie der Anwendungsmanifestdatei die erforderlichen Netzwerkberechtigungen hinzu:

```
<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;
}
```

Wie überprüfe ich die Netzwerkstärke in Android?

```
        ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo Info = cm.getActiveNetworkInfo();
        if (Info == null || !Info.isConnectedOrConnecting()) {
            Log.i(TAG, "No connection");
        } else {
            int netType = Info.getType();
            int netSubtype = Info.getSubtype();

            if (netType == ConnectivityManager.TYPE_WIFI) {
                Log.i(TAG, "Wifi connection");
                WifiManager wifiManager = (WifiManager)
getApplication().getSystemService(Context.WIFI_SERVICE);
                List<ScanResult> scanResult = wifiManager.getScanResults();
                for (int i = 0; i < scanResult.size(); i++) {
                    Log.d("scanResult", "Speed of wifi"+scanResult.get(i).level);//The db
level of signal
                }


                // Need to get wifi strength
            } else if (netType == ConnectivityManager.TYPE_MOBILE) {
                Log.i(TAG, "GPRS/3G connection");
                // Need to get differentiate between 3G/GPRS
            }
        }
```

So überprüfen Sie die Netzwerkstärke

Um die genaue Stärke in Dezibel zu überprüfen, verwenden Sie diese-

```
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
        }
    }
```

Um den Netzwerktyp zu überprüfen, verwenden Sie diese Klassen-

```
 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 "";
        }
    }

}
```

Überprüfen Sie die Internetverbindung online lesen:
https://riptutorial.com/de/android/topic/3918/uberprufen-sie-die-internetverbindung

Kapitel 236: UI-Lebenszyklus

Examples

Speichern von Daten beim Speichern von Daten

```
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);
    }
}
```

Erläuterung

Was passiert also hier?

Das Android-System ist stets bemüht, so viel Speicher wie möglich zu löschen. Wenn Ihre Aktivität also im Hintergrund ist und eine andere Vordergrundaktivität die Freigabe verlangt, ruft das Android-System `onTrimMemory()` für Ihre Aktivität auf.

Das bedeutet jedoch nicht, dass alle Ihre Eigenschaften verschwinden sollten. Was Sie tun sollten, ist, sie in einem Bundle-Objekt zu speichern. Bundle-Objekte werden in Bezug auf den Speicher viel besser behandelt. Innerhalb eines Bündels wird jedes Objekt durch eine eindeutige Textsequenz identifiziert - im obigen Beispiel wird die Ganzzahlvariable `mArg` unter dem Referenznamen `EXAMPLE_ARG` . Wenn die Aktivität wiederhergestellt ist, extrahieren Sie Ihre alten Werte aus dem Bündelobjekt, anstatt sie von Grund auf neu zu erstellen

UI-Lebenszyklus online lesen: https://riptutorial.com/de/android/topic/3440/ui-lebenszyklus

Kapitel 237: UI-Tests schreiben - Android

Einführung

Der Schwerpunkt dieses Dokuments liegt auf der Darstellung von Zielen und Schreibweisen für Android-Benutzeroberflächen- und Integrationstests. Espresso und UIAutomator werden von Google bereitgestellt. Der Fokus sollte sich also auf diese Tools und ihre jeweiligen Verpackungen (z. B. Appium, Löffel usw.) richten.

Syntax

- **Ressource im Leerlauf**
- String getName () - Gibt den Namen der im Leerlauf befindlichen Ressource zurück (wird für die Protokollierung und die Idempotenz der Registrierung verwendet).
- boolean isIdleNow () - Gibt "true" zurück, wenn sich die Ressource aktuell im Leerlauf befindet.
- void registerIdleTransitionCallback (IdlingResource.ResourceCallback-Callback) - Registriert die angegebene IdlingResource.ResourceCallback bei der Ressource

Bemerkungen

JUnit-Regeln:

Wie Sie in MockWebServer Beispiel und ActivityTestRule sie alle fallen unter die Kategorie von JUnit Regeln sehen können , die Sie selbst erstellen können , die dann sollte für jeden Test definiert , sein Verhalten @see ausgeführt werden: https://github.com/junit-team/junit4/ Wiki / Regeln

Appium

Parameter

Da bei Parametern einige Probleme auftreten, setzen Sie sie hier, bis der Dokumentationsfehler behoben ist:

Parameter	Einzelheiten
KlassenaktivitätKlasse	welche Tätigkeit beginnt
initialTouchMode	Sollte die Aktivität beim Start in den Touch-Modus versetzt werden: https://android-developers.blogspot.de/2008/12/touch-mode.html
launchActivity	true, wenn die Aktivität einmal pro Testmethode gestartet werden

Parameter	Einzelheiten
	soll. Es wird vor der ersten Before-Methode gestartet und nach der letzten After-Methode beendet.

Examples

MockWebServer Beispiel

Falls für Ihre Aktivitäten, Fragmente und Benutzeroberflächen Hintergrundverarbeitung erforderlich ist, empfiehlt es sich, einen MockWebServer zu verwenden, der lokal auf einem Android-Gerät ausgeführt wird, das eine geschlossene und testbare Umgebung für Ihre Benutzeroberfläche bietet.

https://github.com/square/okhttp/tree/master/mockwebserver

Der erste Schritt umfasst die Abstufung der Abstufung:

```
testCompile 'com.squareup.okhttp3:mockwebserver:(insert latest version)'
```

Schritte zum Ausführen und Verwenden des Mock-Servers sind jetzt:

- Mock-Serverobjekt erstellen
- Starten Sie es an einer bestimmten Adresse und einem bestimmten Port (normalerweise localhost: Portnummer).
- Antworten für spezifische Anfragen in der Warteschlange
- Test starten

Das wird auf der Github-Seite des Mockwebservers gut erklärt, aber in unserem Fall möchten wir für alle Tests etwas Netteres und Wiederverwendbares, und JUnit-Regeln werden hier gut ins Spiel kommen:

```
/**
 *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”);
        }
    }
}
```

Nehmen wir nun an, dass wir genau die gleiche Aktivität wie im vorigen Beispiel haben. In diesem Fall wird beim Drücken der Schaltfläche die App beispielsweise etwas vom Netzwerk abrufen: https://someapi.com/name

Dies würde eine Textzeichenfolge zurückgeben, die im Snackbar-Text verkettet würde, z. B. NAME + von Ihnen eingegebener Text.

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

Ich würde vorschlagen, den Dispatcher in eine Art Builder zu packen, damit Sie auf einfache Weise neue Antworten für Ihre Bildschirme hinzufügen können. z.B

```
 return newDispatcherBuilder()
            .withSerializedJSONBody("/authenticate", Mocks.getAuthenticationResponse())
            .withSerializedJSONBody("/getUserInfo", Mocks.getUserInfo())
            .withSerializedJSONBody("/checkNotBot", Mocks.checkNotBot());
```

IdlingResource

Die Leistung von Ressourcen im Leerlauf besteht darin, dass Sie nicht auf die Verarbeitung

einiger Apps (Netzwerk, Berechnungen, Animationen usw.) warten müssen, um mit `sleep()` enden, was Flockigkeit verursacht und / oder den Testlauf verlängert. Die offiziellen Unterlagen finden Sie hier .

Implementierung

Bei der Implementierung der `IdlingResource` Schnittstelle müssen Sie drei Dinge tun:

- **`getName()`** - Gibt den Namen Ihrer im Leerlauf befindlichen Ressource zurück.
- **`isIdleNow()`** - Überprüft, ob sich das Xyz-Objekt, die Operation usw. momentan im Leerlauf befindet.
- **`registerIdleTransitionCallback`** (`IdlingResource.ResourceCallback` Rückruf) - Stellt einen Rückruf `IdlingResource.ResourceCallback` den Sie aufrufen sollten, wenn Ihr Objekt in den Leerlauf `IdlingResource.ResourceCallback` .

Jetzt sollten Sie Ihre eigene Logik erstellen und bestimmen, wann sich Ihre App im Leerlauf befindet und wann nicht, da dies von der App abhängt. Nachfolgend finden Sie ein einfaches Beispiel, um zu zeigen, wie es funktioniert. Es gibt andere Beispiele online, aber die Implementierung spezifischer Apps führt zu spezifischen Implementierungen von Ressourcen im Leerlauf.

ANMERKUNGEN

- Es gab einige Google-Beispiele, bei denen `IdlingResources` in den Code der App `IdlingResources` . **Mach das nicht.** Vermutlich haben sie es dort aufgestellt, um zu zeigen, wie sie arbeiten.
- Es bleibt Ihnen überlassen, Ihren Code sauber zu halten und ein Prinzip der Verantwortung zu wahren!

Beispiel

Nehmen wir an, Sie haben eine Aktivität, die seltsame Sachen macht und das Laden des Fragments lange Zeit in Anspruch nimmt, sodass Ihre Espresso-Tests fehlschlagen, wenn Sie keine Ressourcen in Ihrem Fragment finden können (Sie sollten ändern, wie und wann Ihre Aktivität erstellt wird) um es zu beschleunigen). Aber um es einfach zu halten, zeigt das folgende Beispiel, wie es aussehen sollte.

Unsere Beispielressource für den Leerlauf würde zwei Objekte erhalten:

- Das **Tag** des Fragments, das Sie finden müssen und darauf warten, an die Aktivität angehängt zu werden.
- Ein **FragmentManager-** Objekt, das zum Suchen des Fragments verwendet wird.

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

Nun, da Sie Ihre `IdlingResource` geschrieben haben, müssen Sie sie irgendwo richtig einsetzen?

Verwendungszweck

Lassen Sie uns die gesamte Testklasseneinrichtung überspringen und schauen Sie einfach, wie ein Testfall aussehen würde:

```
@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);
}
```

Kombination mit der JUnit-Regel

Das ist nicht zu schwer; Sie können die Leerlaufressource auch in Form einer JUnit-Testregel anwenden. Angenommen, Sie haben ein SDK mit Volley und möchten, dass Espresso darauf wartet. Anstatt jeden Testfall durchzusehen oder im Setup anzuwenden, können Sie eine JUnit-Regel erstellen und einfach Folgendes schreiben:

```
@Rule
public final SDKIdlingRule mSdkIdlingRule = new
SDKIdlingRule(SDKInstanceHolder.getInstance());
```

Jetzt, da dies ein Beispiel ist, sollte es nicht als selbstverständlich gelten. Der gesamte Code hier ist imaginär und wird nur zu Demonstrationszwecken verwendet:

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

UI-Tests schreiben - Android online lesen: https://riptutorial.com/de/android/topic/3530/ui-tests-schreiben---android

Kapitel 238: Umgang mit Berührungs- und Bewegungsereignissen

Einführung

Eine Zusammenfassung einiger grundlegender Berührungs- / Bewegungshandhabungssysteme in der Android-API.

Parameter

Hörer	Einzelheiten
onTouchListener	Behandelt einzelne Tasten für Tasten, Flächen und mehr
onTouchEvent	Ein Listener, der sich in Oberflächen befindet (zB SurfaceView). Muss nicht wie andere Listener eingestellt werden (z. B. onTouchListener)
onLongTouch	Ähnlich wie bei onTouch, hört aber auf langes Drücken auf Tasten, Flächen und mehr.

Examples

Tasten

Berührungsereignisse, die sich auf einen `Button` beziehen, können wie folgt geprüft werden:

```
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;
    }
}
```

Oberfläche

Berührungsereignishandler für Oberflächen (z. B. SurfaceView, GLSurfaceView und andere):

```
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;
    }
}
```

Oder alternativ (in der Oberfläche):

```
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;
    }
}
```

Umgang mit Multitouch in einer Oberfläche

```
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.
    }
}
```

Umgang mit Berührungs- und Bewegungsereignissen online lesen:
https://riptutorial.com/de/android/topic/9315/umgang-mit-beruhrungs--und-bewegungsereignissen

Kapitel 239: Umgang mit Deep Links

Einführung

Deep Links sind URLs, über die Benutzer direkt zu bestimmten Inhalten in Ihrer App gelangen. Sie können Deep Links einrichten, indem Sie Absichtsfilter hinzufügen und Daten aus eingehenden Absichten extrahieren, um Benutzer zum rechten Bildschirm Ihrer App zu führen.

Parameter

`<data>` Attribut	Einzelheiten
planen	Der *Schemateil* einer URI (Groß- und Kleinschreibung beachten). Beispiele: `http` , `https` , `ftp`
Wirt	Der *Host*- Teil einer URI (Groß- und Kleinschreibung wird berücksichtigt) Beispiele: `google.com` , `example.org`
Hafen	Der *Portteil* einer URI. Beispiele: `80` , `443`
Pfad	Der *Pfadteil* einer URI. Muss mit `/` beginnen Beispiele: `/` , `/about`
pathPrefix	Ein Präfix für den *Pfadteil* eines URI. Beispiele: `/item` , `/article`
Pfadmuster	Ein Muster für den *Pfadteil* eines URI. Beispiele: `/item/.*` , `/article/[0-9]*`
Mime Typ	Ein übereinstimmender Mime-Typ. Beispiele: `image/jpeg` , `audio/*`

Bemerkungen

Der `<intent-filter>`

Diese Kombination der Elemente `<action>` und `<category>` teilt dem Android-System mit, dass eine bestimmte Aktivität gestartet werden soll, wenn der Benutzer auf einen Link in einer anderen Anwendung klickt.

```
<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>
```

Mehrere `<data>` -Tags

Der Satz von Deep Links, den Ihr `<intent-filter>` unterstützt, ist das Kreuzprodukt aller `<data>` -Elemente, die Sie in diesem Absichtsfilter definieren. Die Beispiele für mehrere Domänen, mehrere Pfade und mehrere Schemas zeigen dies.

Ressourcen

- Aktivieren von Deep Links für App-Inhalte (developer.android.com)
- `<intent-filter>` (developer.android.com

Examples

Einfache tiefe Verbindung

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>
```

Dadurch wird jeder Link, der mit `http://www.example.com` beginnt, als tiefer Link zum Starten Ihrer `MainActivity` .

Mehrere Pfade in einer einzigen Domäne

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>
```

Dies startet Ihre `MainActivity` wenn der Benutzer auf einen dieser Links klickt

- `http://www.example.com/`
- `http://www.example.com/about`
- `http://www.example.com/map`

Mehrere Domänen und mehrere Pfade

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>
```

Dies startet Ihre MainActivity, wenn der Benutzer auf einen der folgenden Links klickt:

- `http://www.example.com/`
- `http://www.example2.com/`
- `http://www.example.com/map`
- `http://www.example2.com/map`

Sowohl http als auch https für dieselbe Domäne

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>
```

Dies startet Ihre MainActivity, wenn der Benutzer auf einen der folgenden Links klickt:

- `http://www.example.com/`
- `https://www.example.com/`
- `http://www.example.com/map`
- `https://www.example.com/map`

Abfrageparameter abrufen

```
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");
        }
    }

}
```

Wenn der Benutzer auf eine Verknüpfung zu `http://www.example.com/map?param1=FOO&param2=BAR` klickt, hat `param1` hier den Wert `"FOO"` und `param2` den Wert `"BAR"` .

PathPrefix verwenden

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>
```

Dies startet Ihre MainActivity, wenn der Benutzer auf einen Link klickt, der mit `http://www.example.com/item` beginnt.

- `https://www.example.com/item`
- `http://www.example.com/item/1234`
- `https://www.example.com/item/xyz/details`

Umgang mit Deep Links online lesen: https://riptutorial.com/de/android/topic/3716/umgang-mit-deep-links

Kapitel 240: Unit-Tests in Android mit JUnit

Bemerkungen

- Vogella: Unit Testing mit JUnit
- Junit-Anmerkungen: java2novice.com
- Assert Class: junit.org
- JUnit Api: tutorialspoint.com
- Anroid Testing Medium.com Beiträge

Examples

Local Unit-Tests erstellen

Platzieren Sie Ihre `/src/test/<pkg_name>/` hier: `/src/test/<pkg_name>/`

Beispiel Testklasse

```
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
    }
}
```

Nervenzusammenbruch

```
public class ExampleUnitTest {
  ...
}
```

In der Testklasse können Sie mehrere Testklassen erstellen und diese innerhalb des Testpakets platzieren.

```
@Test
public void addition_isCorrect() {
   ...
}
```

Mit der Testmethode können mehrere Testmethoden innerhalb einer Testklasse erstellt werden.

Beachten Sie die Anmerkung `@Test` .

Die Testannotation teilt JUnit mit, dass die öffentliche Void-Methode, an die sie angehängt ist, als Testfall ausgeführt werden kann.

Es gibt einige weitere nützliche Anmerkungen wie `@Before` , `@After` usw. Diese Seite wäre ein guter Anfang.

```
assertEquals(9, c); // This test passes
assertEquals(10, c); //Test fails
```

Diese Methoden sind Mitglied der `Assert` Klasse. Einige andere nützliche Methoden sind `assertFalse()` , `assertNotNull()` , `assertTrue` usw. Hier eine ausführliche Erklärung .

Anmerkungsinformationen für JUnit Test:

@Test: Die Test-Annotation teilt JUnit mit, dass die öffentliche Void-Methode, mit der sie verbunden ist, als Testfall ausgeführt werden kann. Um die Methode auszuführen, erstellt JUnit zuerst eine neue Instanz der Klasse und ruft dann die annotierte Methode auf.

@Before: Beim Schreiben von Tests wird häufig festgestellt, dass für mehrere Tests ähnliche Objekte erstellt werden müssen, bevor sie ausgeführt werden können. Wenn Sie eine öffentliche void-Methode mit `@Before` wird diese Methode vor der Test-Methode ausgeführt.

@After: Wenn Sie externe Ressourcen in einer Before-Methode zuordnen, müssen Sie sie nach dem Testlauf freigeben. Wenn Sie eine öffentliche void-Methode mit `@After` wird diese Methode nach der Test-Methode ausgeführt. Es wird garantiert, dass alle `@After` Methoden ausgeführt werden, auch wenn eine `@After` oder Test-Methode eine Ausnahme `@After`

Tipp Erstellen Sie schnell Testklassen in Android Studio

- Setzen Sie den Cursor auf den Klassennamen, für den Sie eine Testklasse erstellen möchten.
- Drücken Sie Alt + Eingabe (Windows).
- Wählen Sie Create Test aus und drücken Sie die Eingabetaste.
- Wählen Sie die Methoden aus, für die Sie Testmethoden erstellen möchten, und klicken Sie auf OK.
- Wählen Sie das Verzeichnis aus, in dem Sie die Testklasse erstellen möchten.
- Sie sind fertig, das, was Sie bekommen, ist Ihre erste Prüfung.

Tipp Führen Sie Tests einfach in Android Studio aus

- Klicken Sie mit der rechten Maustaste, um das Paket zu testen.
- Wählen Sie Ausführen 'Tests in ...
- Alle Tests im Paket werden sofort ausgeführt.

Geschäftslogik aus Android-Komponenten herausziehen

Ein großer Teil des Nutzens aus lokalen JVM-Komponententests kommt von der Art und Weise, wie Sie Ihre Anwendung entwerfen. Sie müssen es so gestalten, dass Sie Ihre Geschäftslogik von Ihren Android-Komponenten trennen können. Hier ist ein Beispiel für eine solche Verwendung des Model-View-Presenter-Musters . Lassen Sie uns dies üben, indem Sie einen einfachen Anmeldebildschirm implementieren, der nur einen Benutzernamen und ein Kennwort enthält. Unsere Android-App ist dafür verantwortlich, dass der Benutzername, den der Benutzer bereitstellt, nicht leer ist und dass das Kennwort mindestens acht Zeichen lang ist und mindestens eine Ziffer enthält. Wenn der Benutzername / das Passwort gültig ist, führen wir unseren Anmeldungsaufruf aus, andernfalls wird eine Fehlermeldung angezeigt.

Beispiel, bei dem die Geschäftslogik stark mit der Android-Komponente gekoppelt ist.

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

Beispiel, bei dem die Geschäftslogik von der Android-Komponente entkoppelt ist.

Hier definieren wir in einer einzigen Klasse, LoginContract, die die verschiedenen Interaktionen zwischen unseren verschiedenen Klassen beherbergen wird.

```
public interface LoginContract {
    public interface View {
        performSignUpApiCall(String username, String password);
        displayInvalidCredentialsErrorMessage();
    }
    public interface Presenter {
        void validateUserCredentials(String username, String password);
    }
}
```

Unsere LoginActivity ist größtenteils gleich, mit der Ausnahme, dass wir die Verantwortung für das Validieren des Anmeldeformulars eines Benutzers (unsere Geschäftslogik) aufgehoben haben. Die LoginActivity-Funktion verlässt sich jetzt auf den neuen LoginPresenter, um die Validierung durchzuführen.

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

Jetzt befindet sich Ihre Geschäftslogik in Ihrer neuen LoginPresenter-Klasse.

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

Jetzt können wir lokale JVM-Komponententests für Ihre neue LoginPresenter-Klasse erstellen.

```
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);
    }
}
```

Wie Sie sehen, haben wir unsere Geschäftslogik aus der LoginActivity extrahiert und in den LoginPresenter POJO gestellt . Wir können jetzt lokale JVM-Komponententests gemäß unserer Geschäftslogik erstellen.

Es sollte beachtet werden, dass unsere Architekturveränderung verschiedene andere Auswirkungen hat, wie zum Beispiel, dass wir uns fast an jeder Klasse mit einer einzigen Verantwortung, an zusätzlichen Klassen usw. halten Entkopplung über den MVP-Style. MVP ist nur eine Möglichkeit, dies zu tun, aber es gibt andere Alternativen, die Sie sich anschauen möchten, wie MVVM . Sie müssen nur das beste System auswählen, das für Sie funktioniert.

Erste Schritte mit JUnit

Konfiguration

Um den Gerätetest Ihres Android-Projekts mit JUnit zu starten, müssen Sie die JUnit-Abhängigkeit zu Ihrem Projekt hinzufügen und einen Testquellensatz erstellen, der den Quellcode für die Komponententests enthält. Projekte, die mit Android Studio erstellt wurden, enthalten häufig bereits die JUnit-Abhängigkeit und den Testquellensatz

Fügen Sie der Modul- `build.gradle` Datei die folgende Zeile innerhalb der Abhängigkeiten `Closure` :

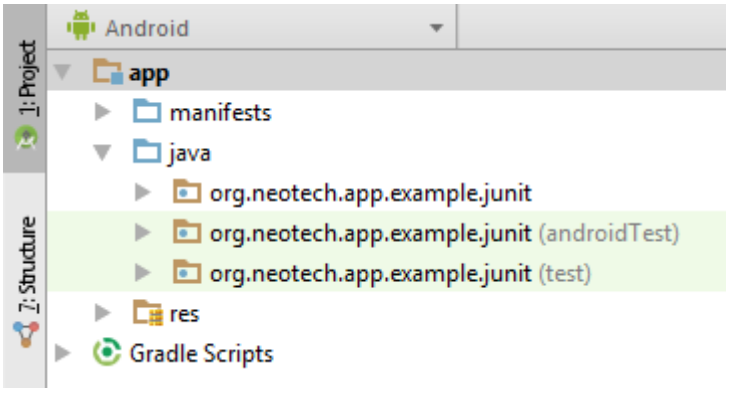
```
testCompile 'junit:junit:4.12'
```

JUnit-Testklassen befinden sich in einem speziellen Quellsatz mit dem Namen `test` . Wenn diese Quellgruppe nicht vorhanden ist, müssen Sie selbst einen neuen Ordner erstellen. Die Ordnerstruktur eines Standardprojekts von Android Studio (Gradle-basiert) sieht folgendermaßen aus:

```
<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)
```

Wenn Ihr Projekt nicht über den Ordner `/app/src/test` , müssen Sie es selbst erstellen. Im `test` müssen Sie auch einen `java` - Ordner (erstellen Sie es , wenn es nicht existiert). Die Java - Ordner in der `test` Satz sollte enthalten die gleiche Paketstruktur als `main`

Wenn Sie Ihre Projektstruktur (in der Android-Ansicht in Android Studio) richtig eingerichtet haben, sollte dies folgendermaßen aussehen:



Hinweis: Sie müssen nicht unbedingt das `androidTest` Quellenset haben. Dieses Quellenset wird häufig in Projekten gefunden, die von Android Studio erstellt wurden, und ist hier als Referenz enthalten.

Test schreiben

1. **Erstellen Sie eine neue Klasse innerhalb der `test` .**
   Klicken Sie in der Projektansicht mit der rechten Maustaste auf den Testquellensatz, und wählen Sie `New` > `Java class` .
   Das am häufigsten verwendete Benennungsmuster ist die Verwendung des Namens der

Klasse, die Sie testen möchten, wobei `Test` hinzugefügt wird. So wird `StringUtilities` zu `StringUtilitiesTest` .

2. **Fügen Sie die Anmerkung `@RunWith`**
   Die Annotation `@RunWith` ist erforderlich, damit JUnit die Tests `@RunWith` , die in unserer `@RunWith` definiert werden. Der Standard-JUnit-Runner (für JUnit 4) ist `BlockJUnit4ClassRunner` aber anstatt direkt zu laufen, ist es bequemer, den Alias `JUnit4` der eine Abkürzung für den Standard-JUnit-Runner ist.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

}
```

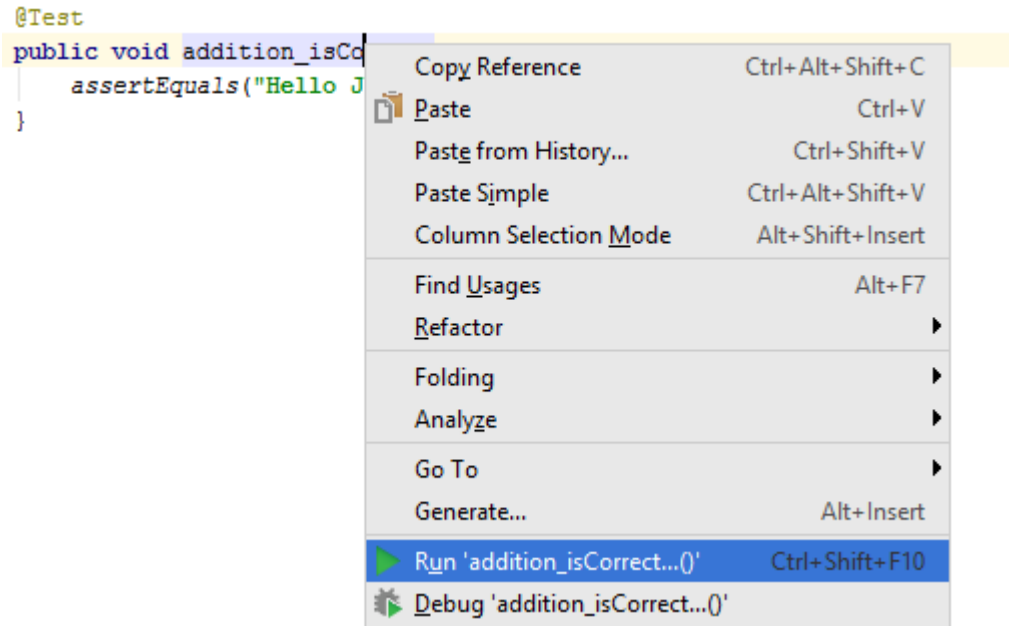
3. **Erstellen Sie einen Test**
   Ein Komponententest ist im Wesentlichen nur eine Methode, die in den meisten Fällen nicht fehlschlagen sollte, wenn sie ausgeführt wird. Mit anderen Worten, es sollte keine Ausnahme auslösen. In einer Testmethode finden Sie fast immer Assertionen, die prüfen, ob bestimmte Bedingungen erfüllt sind. Wenn eine Assertion fehlschlägt, wird eine Ausnahme ausgelöst, die dazu führt, dass die Methode / der Test fehlschlägt. Eine Testmethode wird immer mit der Annotation `@Test` . Ohne diese Anmerkung führt JUnit den Test nicht automatisch aus.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

    @Test
    public void addition_isCorrect() throws Exception {
        assertEquals("Hello JUnit", "Hello" + " " + "JUnit");
    }
}
```

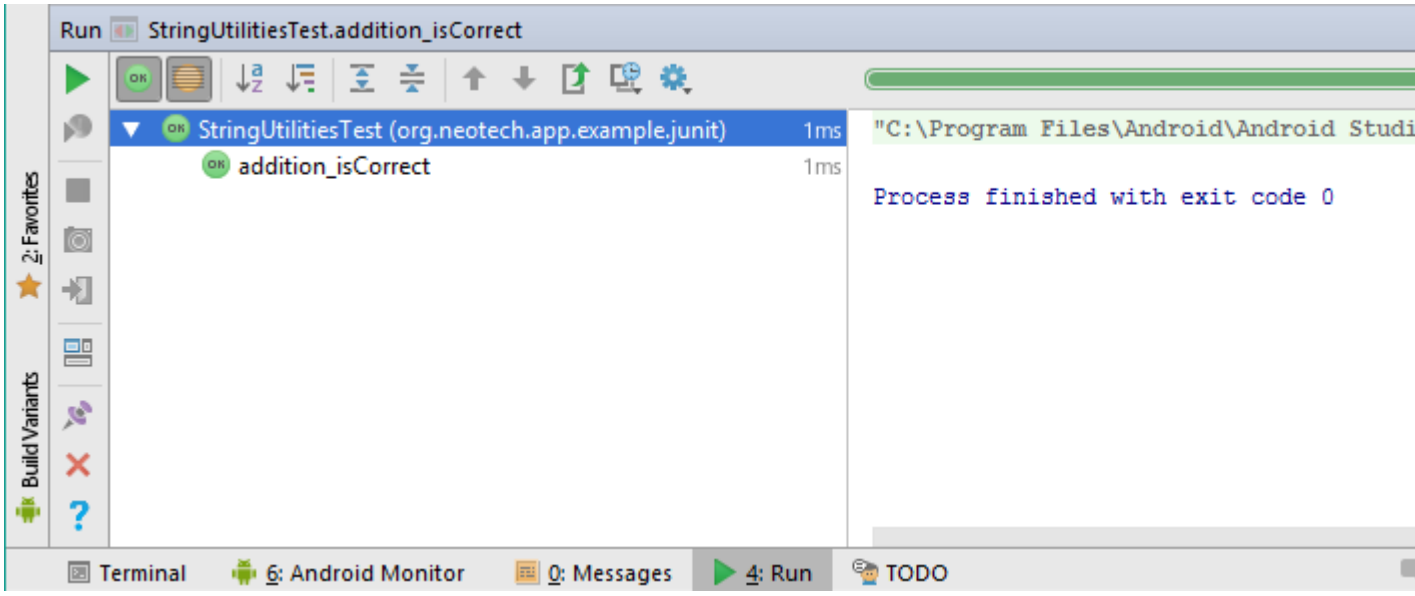
Hinweis: Im Gegensatz zu den Standardmethoden für die Benennung von Java-Methoden enthalten die Namen von Testmethoden oft Unterstriche.

Test durchführen

1. **Methode**
   Um eine einzelne Testmethode auszuführen, klicken Sie mit der rechten Maustaste auf die Methode und klicken Sie auf `Run 'addition_isCorrect()'` oder verwenden Sie die Tastenkombination `ctrl+shift+f10` .



Wenn alles korrekt eingerichtet ist, startet JUnit die Methode und Sie sollten die folgende Schnittstelle in Android Studio sehen:



2. **Klasse**
   Sie können auch alle in einer einzelnen Klasse definierten Tests ausführen, indem Sie in der Projektansicht mit der rechten Maustaste auf die Klasse klicken und auf `Run 'StringUtilitiesTest '` oder die Tastenkombination `ctrl+shift+f10` wenn Sie die Klasse in der Projektansicht ausgewählt haben.

3. **Paket (alles)**
   Wenn Sie nicht alle im Projekt oder in einem Paket definierten Tests ausführen möchten, können Sie einfach mit der rechten Maustaste auf das Paket klicken und auf `Run ...` klicken, genauso wie Sie alle in einer einzelnen Klasse definierten Tests ausführen würden.

Ausnahmen

Mit JUnit kann auch getestet werden, ob eine Methode für eine bestimmte Eingabe eine

bestimmte Ausnahme auslöst.

In diesem Beispiel testen wir, ob die folgende Methode wirklich eine Ausnahme auslöst, wenn das boolesche Format (Eingabe) nicht erkannt / unbekannt ist:

```
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);
    }
}
```

Durch Hinzufügen des `expected` Parameters zur `@Test` Annotation kann definiert werden, welche Ausnahme erwartet wird. Der Komponententest schlägt fehl, wenn diese Ausnahme nicht auftritt, und ist erfolgreich, wenn die Ausnahme tatsächlich ausgelöst wird:

```
@Test(expected = IllegalArgumentException.class)
public void parseBoolean_parsesInvalidFormat_throwsException(){
    StringUtilities.parseBoolean("Hello JUnit");
}
```

Dies funktioniert gut, beschränkt sich jedoch auf einen einzelnen Testfall innerhalb der Methode. Manchmal möchten Sie möglicherweise mehrere Fälle in einer einzigen Methode testen. `Assert.fail()` diese Einschränkung zu überwinden, werden häufig `try-catch` Blöcke und die `Assert.fail()` -Methode verwendet:

```
@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){
    }
}
```

Hinweis: Einige Leute halten es für eine schlechte Praxis, mehr als einen einzelnen Fall innerhalb eines Komponententests zu testen.

Statischer Import

JUnit definiert einige `assertEquals` Methoden, von denen mindestens eine für jeden `assertEquals` und eine für Objekte verfügbar ist. Diese Methoden sind standardmäßig nicht direkt für den Aufruf

verfügbar und sollten wie `Assert.assertEquals` aufgerufen werden: `Assert.assertEquals` . Da diese Methoden jedoch so häufig verwendet werden, wird fast immer ein statischer Import verwendet, sodass die Methode direkt verwendet werden kann, als wäre sie Teil der Klasse selbst.

Verwenden Sie die folgende Importanweisung, um einen statischen Import für die `assertEquals` Methode hinzuzufügen:

```
import static org.junit.Assert.assertEquals;
```

Sie können auch alle `assertArrayEquals` Methoden einschließlich der `assertArrayEquals` , `assertNotNull` und `assertFalse` usw. mithilfe des folgenden statischen Imports statisch importieren:

```
import static org.junit.Assert.*;
```

Ohne statischen Import:

```
@Test
public void addition_isCorrect(){
    Assert.assertEquals(4 , 2 + 2);
}
```

Bei statischem Import:

```
@Test
public void addition_isCorrect(){
    assertEquals(4 , 2 + 2);
}
```

Unit-Tests in Android mit JUnit online lesen: https://riptutorial.com/de/android/topic/3205/unit-tests-in-android-mit-junit

Kapitel 241: Universal Image Loader

Bemerkungen

Akzeptable URI-Beispiele:

```
"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

Initialisieren Sie den Universal Image Loader

1. Fügen Sie der Datei *build.gradle* die folgende Abhängigkeit *hinzu* :

   ```
   compile 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
   ```

2. Fügen Sie der Datei *AndroidManifest.xml* die folgenden Berechtigungen *hinzu* :

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

3. Initialisieren Sie den Universal Image Loader. Dies muss vor der ersten Verwendung erfolgen:

   ```
   ImageLoaderConfiguration config = new ImageLoaderConfiguration.Builder(this)
       // ...
       .build();
   ImageLoader.getInstance().init(config);
   ```

   Die vollständigen Konfigurationsmöglichkeiten finden Sie hier .

Grundlegende Verwendung

1. Laden Sie ein Bild, dekodieren Sie es in eine Bitmap und zeigen Sie die Bitmap in einer `ImageView` (oder einer anderen Ansicht, die die `ImageAware` Schnittstelle implementiert) an:

   ```
   ImageLoader.getInstance().displayImage(imageUri, imageView);
   ```

2. Laden Sie ein Bild, dekodieren Sie es in eine Bitmap und geben Sie die Bitmap an einen Rückruf zurück:

```
ImageLoader.getInstance().loadImage(imageUri, new SimpleImageLoadingListener() {
    @Override
    public void onLoadingComplete(String imageUri, View view, Bitmap loadedImage) {
        // Do whatever you want with the bitmap.
    }
});
```

3. Laden Sie ein Bild, dekodieren Sie es in eine Bitmap und geben Sie die Bitmap synchron zurück:

```
Bitmap bmp = ImageLoader.getInstance().loadImageSync(imageUri);
```

Universal Image Loader online lesen: https://riptutorial.com/de/android/topic/2760/universal-image-loader

Kapitel 242: Untere Blätter

Einführung

Ein unteres Blatt ist ein Blatt, das vom unteren Rand des Bildschirms nach oben verschoben wird.

Bemerkungen

Die unteren Blätter werden vom unteren Rand des Bildschirms nach oben verschoben, um mehr Inhalt anzuzeigen.
Sie wurden in der Version 23.2.0 der Android Support Library hinzugefügt.

Examples

BottomSheetBehavior wie Google maps

2.1.x

Dieses Beispiel hängt von der Support Library 23.4.0. + Ab.

BottomSheetBehavior zeichnet sich aus durch:

1. Zwei Symbolleisten mit Animationen, die auf die unteren Blattbewegungen reagieren.
2. Ein FAB, das ausgeblendet wird, wenn es sich in der Nähe der "modalen Symbolleiste" befindet (die beim Hochrutschen angezeigt wird).
3. Ein Hintergrundbild hinter dem unteren Blatt mit einem Parallaxeffekt.
4. Ein Titel (TextView) in der Symbolleiste, der angezeigt wird, wenn das unterste Blatt es erreicht.
5. Die Benachrichtigungs-Satus-Leiste kann ihren Hintergrund transparent oder vollfarbig gestalten.
6. Ein benutzerdefiniertes unteres Blattverhalten mit dem Status "Anker".

Jetzt wollen wir sie einzeln prüfen:

Werkzeugleisten
Wenn Sie diese Ansicht in Google Maps öffnen, sehen Sie eine Symbolleiste, in der Sie suchen können. Dies ist die einzige, die ich nicht genau wie Google Maps mache, weil ich es generischer machen wollte. Die `ToolBar` befindet sich jedoch in einem `AppBarLayout` und wurde ausgeblendet, wenn Sie das BottomSheet ziehen, und es wird wieder angezeigt, wenn das `COLLAPSED` den `COLLAPSED` .
Um dies zu erreichen, müssen Sie:

- Erstellen Sie ein `Behavior` und erweitern Sie es von `AppBarLayout.ScrollingViewBehavior`

- Überschreiben `onDependentViewChanged` Methoden `layoutDependsOn` und `onDependentViewChanged` . Wenn Sie es tun, werden Sie auf die Bewegungen des unteren Blattes achten.
- Erstellen Sie einige Methoden, um AppBarLayout / ToolBar mit Animationen ein- und auszublenden.

So habe ich es für die erste Symbolleiste oder ActionBar gemacht:

```
@Override
public boolean layoutDependsOn(CoordinatorLayout parent, View child, View dependency) {
    return dependency instanceof NestedScrollView;
}

@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, View child,
                                      View dependency) {

    if (mChild == null) {
        initValues(child, dependency);
        return false;
    }

    float dVerticalScroll = dependency.getY() - mPreviousY;
    mPreviousY = dependency.getY();

    //going up
    if (dVerticalScroll <= 0 && !hidden) {
        dismissAppBar(child);
        return true;
    }

    return false;
}

private void initValues(final View child, View dependency) {

    mChild = child;
    mInitialY = child.getY();

    BottomSheetBehaviorGoogleMapsLike bottomSheetBehavior =
BottomSheetBehaviorGoogleMapsLike.from(dependency);
    bottomSheetBehavior.addBottomSheetCallback(new
BottomSheetBehaviorGoogleMapsLike.BottomSheetCallback() {
        @Override
        public void onStateChanged(@NonNull View bottomSheet,
@BottomSheetBehaviorGoogleMapsLike.State int newState) {
            if (newState == BottomSheetBehaviorGoogleMapsLike.STATE_COLLAPSED ||
                    newState == BottomSheetBehaviorGoogleMapsLike.STATE_HIDDEN)
                showAppBar(child);
        }

        @Override
        public void onSlide(@NonNull View bottomSheet, float slideOffset) {

        }
    });
}

private void dismissAppBar(View child){
    hidden = true;
```

```
    AppBarLayout appBarLayout = (AppBarLayout)child;
    mToolbarAnimation =
appBarLayout.animate().setDuration(mContext.getResources().getInteger(android.R.integer.config_shortAni

    mToolbarAnimation.y(-(mChild.getHeight()+25)).start();
}

private void showAppBar(View child) {
    hidden = false;
    AppBarLayout appBarLayout = (AppBarLayout)child;
    mToolbarAnimation =
appBarLayout.animate().setDuration(mContext.getResources().getInteger(android.R.integer.config_mediumAn

    mToolbarAnimation.y(mInitialY).start();
}
```

Hier ist die komplette Datei, wenn Sie es brauchen

Die zweite Symbolleiste oder "Modal" -Symbolleiste:
Sie müssen die gleichen Methoden überschreiben, aber in diesem Fall müssen Sie sich um mehr Verhalten kümmern:

- Werkzeugleiste mit Animationen anzeigen / ausblenden
- Farbe / Hintergrund der Statusleiste ändern
- Ein- / Ausblenden des BottomSheet-Titels in der Symbolleiste
- Schließen Sie das bottomSheet oder senden Sie es in den minimierten Zustand

Der Code für dieses ist ein wenig umfangreich, also werde ich den Link zulassen

Die FAB

Dies ist auch ein benutzerdefiniertes Verhalten, erstreckt sich jedoch von `FloatingActionButton.Behavior` . In `onDependentViewChanged` Sie nachsehen, wenn es das "offSet" erreicht, oder an die Stelle zeigen, an der Sie es ausblenden möchten. In meinem Fall möchte ich es ausblenden, wenn es sich in der Nähe der zweiten Symbolleiste befindet. Ich greife in FAB (ein CoordinatorLayout) nach dem AppBarLayout, das die Toolbar enthält. Dann verwende ich die Toolbar-Position wie `OffSet` :

```
@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, FloatingActionButton child,
View dependency) {

    if (offset == 0)
        setOffsetValue(parent);

    if (dependency.getY() <=0)
        return false;

    if (child.getY() <= (offset + child.getHeight()) && child.getVisibility() == View.VISIBLE)
        child.hide();
    else if (child.getY() > offset && child.getVisibility() != View.VISIBLE)
        child.show();
```

```
    return false;
}
```

Vollständige Verknüpfung zum benutzerdefinierten FAB-Verhalten

Das Bild hinter dem BottomSheet mit Parallaxeffekt :
Wie bei den anderen handelt es sich um ein benutzerdefiniertes Verhalten. Das einzige "Komplizierte" in diesem Fall ist der kleine Algorithmus, der das Bild im BottomSheet verankert und den Bildeinbruch wie einen standardmäßigen Parallaxeffekt verhindert:

```
@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;
}
```

Die komplette Datei für das Hintergrundbild mit Parallax-Effekt

Nun zum Schluss: **Das Verhalten von Custom BottomSheet**
Um die 3 Schritte zu erreichen, müssen Sie zunächst verstehen, dass das standardmäßige BottomSheetBehavior 5 `STATE_DRAGGING, STATE_SETTLING, STATE_EXPANDED, STATE_COLLAPSED, STATE_HIDDEN` hat: `STATE_DRAGGING, STATE_SETTLING, STATE_EXPANDED, STATE_COLLAPSED, STATE_HIDDEN` und für das Verhalten von Google Maps einen mittleren Status zwischen collapsed und `STATE_ANCHOR_POINT` : `STATE_ANCHOR_POINT` .
Ich habe versucht, das Standard-bottomSheetBehavior ohne Erfolg zu erweitern. Daher kopiere ich den gesamten Code und modifizierte, was ich brauche.
Um zu erreichen, worüber ich spreche, folgen Sie den nächsten Schritten:

1. Erstellen Sie eine Java-Klasse, und erweitern Sie sie aus `CoordinatorLayout.Behavior<V>`
2. Kopieren Sie den Einfügecode aus der Standarddatei `BottomSheetBehavior` in Ihre neue.
3. Ändern Sie die Methode `clampViewPositionVertical` mit folgendem Code:

```
@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. Fügen Sie einen neuen Status hinzu

   public static final int STATE_ANCHOR_POINT = X;

5. Ändern Sie die nächsten Methoden: `onLayoutChild` , `onStopNestedScroll` , `BottomSheetBehavior<V> from(V view)` und `setState` (optional).

```
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;
}
```

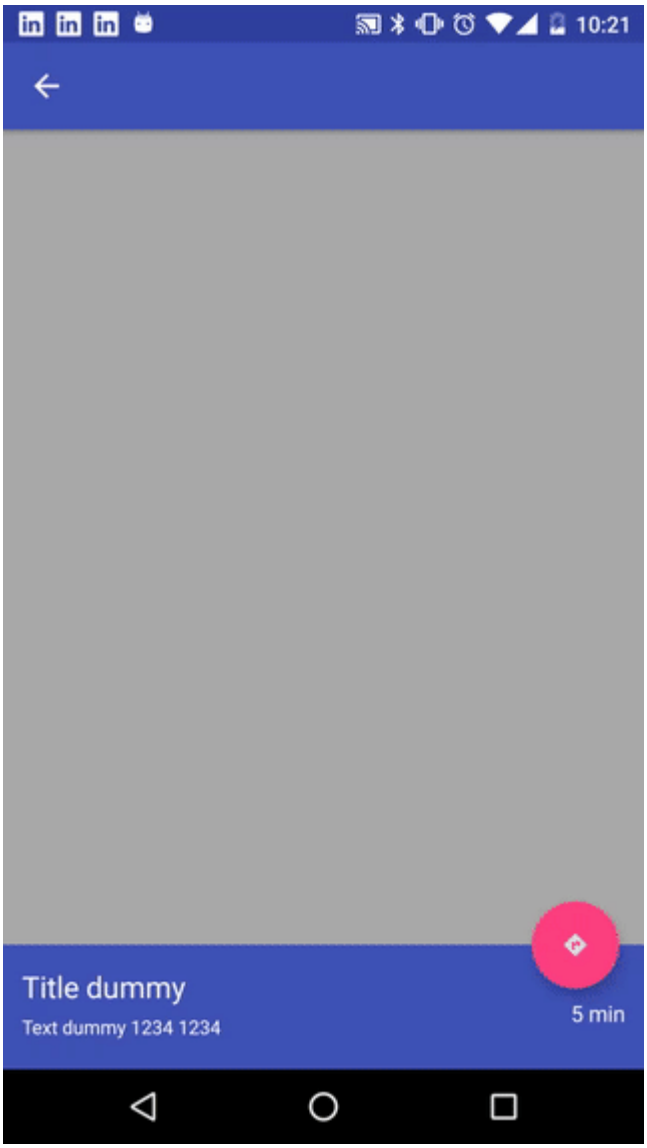
```
public void onStopNestedScroll(CoordinatorLayout coordinatorLayout, V child, View target) {
    if (child.getTop() == mMinOffset) {
        setStateInternal(STATE_EXPANDED);
        return;
    }
    if (target != mNestedScrollingChildRef.get() || !mNestedScrolled) {
        return;
    }
    int 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;
}
```

Verknüpfen Sie mit dem gesamten Projekt, in dem Sie alle benutzerdefinierten Verhaltensweisen sehen können

Und so sieht es aus:

[



]

Schnelle Einrichtung

Stellen Sie sicher, dass die folgende Abhängigkeit zur build.gradle-Datei Ihrer App unter Abhängigkeiten hinzugefügt wird:

```
compile 'com.android.support:design:25.3.1'
```

Dann können Sie das untere Blatt mit diesen Optionen verwenden:

- `BottomSheetBehavior` , das mit `CoordinatorLayout`
- `BottomSheetDialog` ist ein Dialog mit einem Verhalten der unteren Tabelle
- `BottomSheetDialogFragment` ist eine Erweiterung von `DialogFragment` , die einen `BottomSheetDialog` anstelle eines Standarddialogfelds erstellt.

Persistente untere Blätter

Sie können ein beständiges Bottom-Sheet erreichen, indem Sie ein `BottomSheetBehavior` an eine `BottomSheetBehavior` Ansicht eines `CoordinatorLayout` anhängen:

```
<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>
```

Dann können Sie in Ihrem Code eine Referenz erstellen mit:

```
 // The View with the BottomSheetBehavior
 View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
 BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

Sie können den Status Ihres BottomSheetBehavior mithilfe der setState () -Methode festlegen :

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

Sie können einen dieser Zustände verwenden:

- `STATE_COLLAPSED` : Dieser `STATE_COLLAPSED` Status ist die Standardeinstellung und zeigt nur einen Teil des Layouts am unteren Rand. Die Höhe kann mit dem `app:behavior_peekHeight` Attribut (Standardeinstellung 0) gesteuert werden.
- `STATE_EXPANDED` : Der vollständig erweiterte Zustand des untersten Blattes, wobei entweder das gesamte unterste Blatt sichtbar ist (wenn seine Höhe geringer als das enthaltene `CoordinatorLayout`) oder das gesamte `CoordinatorLayout` gefüllt ist
- `STATE_HIDDEN` : Standardmäßig deaktiviert (und mit dem Attribut `app:behavior_hideable` aktiviert). `STATE_HIDDEN` können Benutzer das untere Blatt nach unten streichen, um das untere Blatt vollständig auszublenden

Wenn Sie Rückrufe von `BottomSheetCallback` erhalten möchten, können Sie `BottomSheetCallback` hinzufügen:

```
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
   }
```

```
});
```

Modale untere Blätter mit BottomSheetDialogFragment

Sie können modale untere Arbeitsblätter mit einem `BottomSheetDialogFragment` .

Das `BottomSheetDialogFragment` ist ein modales unteres Blatt.
Dies ist eine Version von `DialogFragment` , die ein unteres Blatt mit `BottomSheetDialog` anstelle eines schwebenden Dialogfelds zeigt.

Definieren Sie einfach das Fragment:

```
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);
    }
}
```

Verwenden Sie dann diesen Code, um das Fragment anzuzeigen:

```
MyBottomSheetDialogFragment mySheetDialog = new MyBottomSheetDialogFragment();
FragmentManager fm = getSupportFragmentManager();
mySheetDialog.show(fm, "modalSheetDialog");
```

Dieses Fragment erstellt einen `BottomSheetDialog` .

Modale untere Blätter mit BottomSheetDialog

Der `BottomSheetDialog` ist ein Dialog, der als unteres Blatt gestaltet ist

Benutz einfach:

```
//Create a new BottomSheetDialog
BottomSheetDialog dialog = new BottomSheetDialog(context);
//Inflate the layout R.layout.my_dialog_layout
dialog.setContentView(R.layout.my_dialog_layout);
//Show the dialog
dialog.show();
```

In diesem Fall müssen Sie kein BottomSheet-Verhalten hinzufügen.

Öffnen Sie BottomSheet DialogFragment standardmäßig im erweiterten Modus.

BottomSheet `STATE_COLLAPSED` wird standardmäßig in `STATE_COLLAPSED` . Welche kann erzwungen werden, um `STATE_EXPANDED` zu `STATE_EXPANDED` und den gesamten `STATE_EXPANDED` mit Hilfe der folgenden `STATE_EXPANDED` .

@NonNull @Override public Dialog onCreateDialog (Bundle savedInstanceState) {

```
    BottomSheetDialog dialog = (BottomSheetDialog) super.onCreateDialog(savedInstanceState);

    dialog.setOnShowListener(new DialogInterface.OnShowListener() {
        @Override
        public void onShow(DialogInterface dialog) {
            BottomSheetDialog d = (BottomSheetDialog) dialog;

            FrameLayout bottomSheet = (FrameLayout)
d.findViewById(android.support.design.R.id.design_bottom_sheet);

BottomSheetBehavior.from(bottomSheet).setState(BottomSheetBehavior.STATE_EXPANDED);
        }
    });

    // Do something with your dialog like setContentView() or whatever
    return dialog;
}
```

Die Dialoganimation ist zwar etwas auffällig, macht aber das Öffnen des DialogFragment im Vollbildmodus sehr gut.

Untere Blätter online lesen: https://riptutorial.com/de/android/topic/5702/untere-blatter

Kapitel 243: Unterschreiben Sie Ihre Android App zur Veröffentlichung

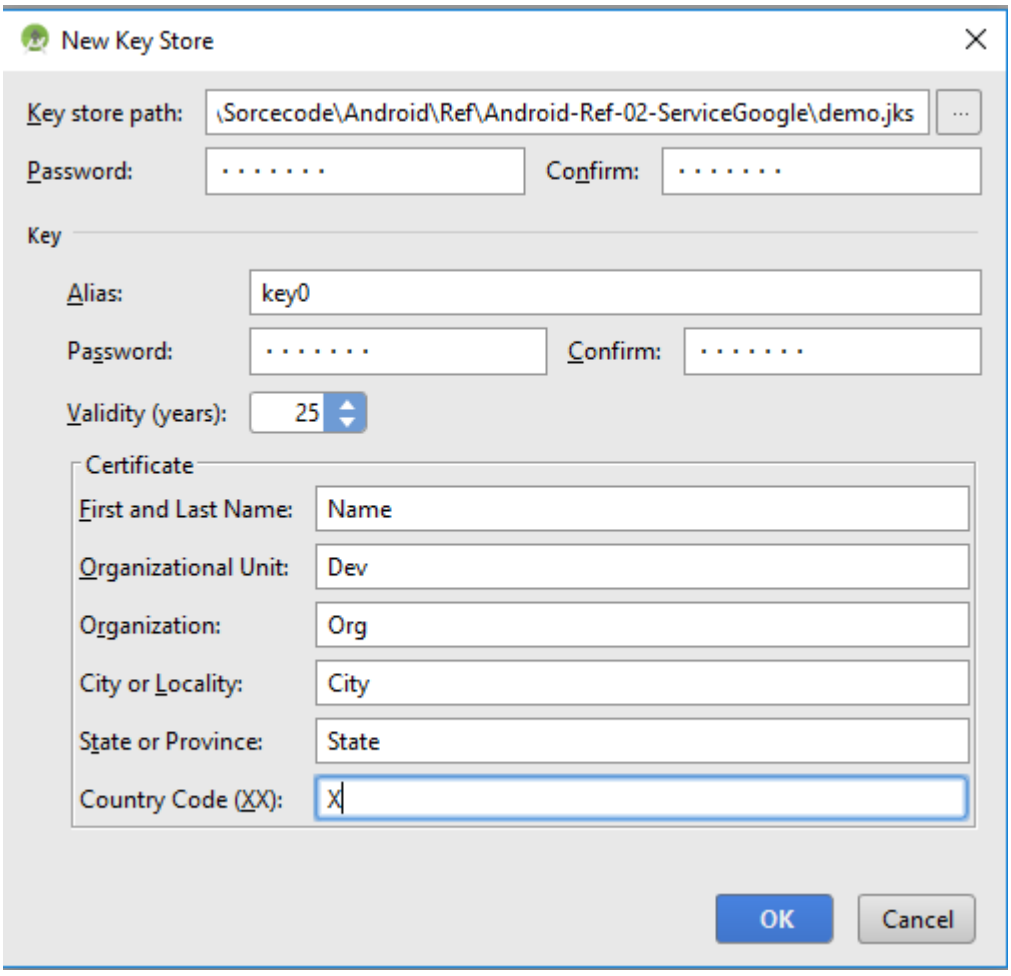
Einführung

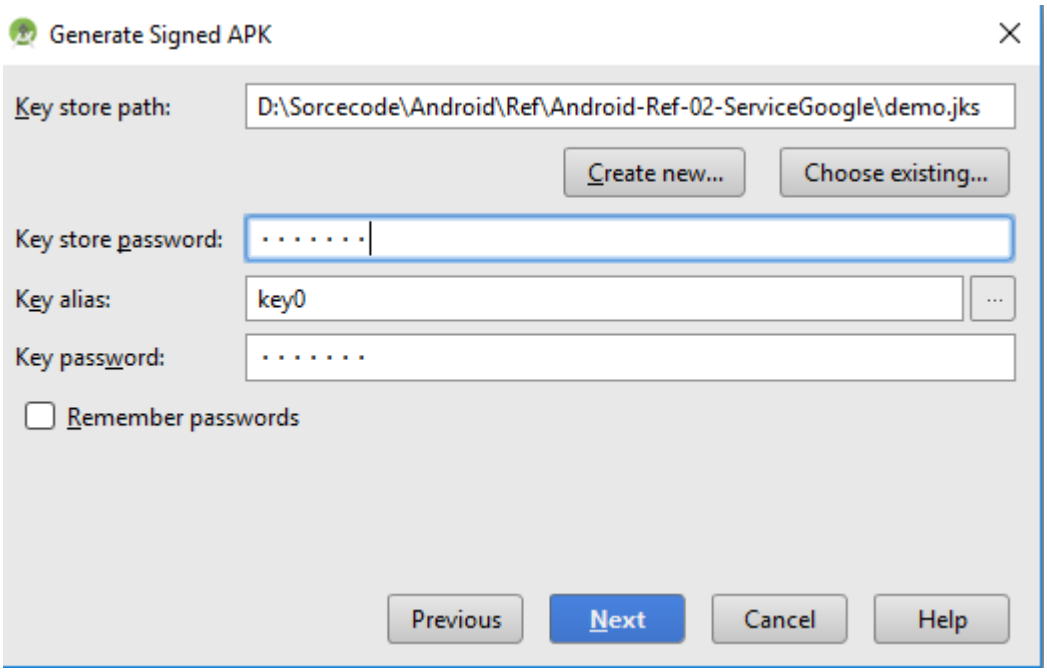
Android erfordert, dass alle APKs zur Veröffentlichung unterschrieben sind.

Examples

Unterschreiben Sie Ihre App

1. Klicken Sie in der Menüleiste auf Erstellen> Signierte APK generieren.
2. Wählen Sie das Modul aus, das Sie freigeben möchten, und klicken Sie auf Weiter.
3. Um einen neuen Keystore zu erstellen, klicken Sie auf Neu erstellen. Geben Sie nun die erforderlichen Informationen ein und klicken Sie in New Key Store auf OK.





4. Im Assistenten zum Generieren von signierten APK-Dateien sind bereits Felder eingetragen, wenn Sie gerade einen neuen Schlüsselspeicher erstellt haben.

5. Wählen Sie im nächsten Fenster ein Ziel für die signierte APK aus, wählen Sie den Build-Typ aus und klicken Sie auf Fertig stellen.

Konfigurieren Sie das build.gradle mit der Signierkonfiguration

Sie können die Signierungskonfiguration definieren, um die apk in der Datei `build.gradle` zu signieren.

Sie können definieren:

- `storeFile` : die Keystore-Datei
- `storePassword` : das Schlüsselspeicherkennwort
- `keyAlias` : ein Aliasname
- `keyPassword` : Ein Schlüsselalias-Kennwort

Sie müssen den `signingConfigs` Block **definieren** , um eine `signingConfigs` zu erstellen:

```
android {
    signingConfigs {

        myConfig {
            storeFile file("myFile.keystore")
            storePassword "xxxx"
            keyAlias "xxxx"
            keyPassword "xxxx"
        }
    }
    //....
}
```

Dann können Sie es einem oder mehreren Build-Typen **zuordnen** .

```
android {

    buildTypes {
        release {
            signingConfig signingConfigs.myConfig
        }
    }
}
```

Unterschreiben Sie Ihre Android App zur Veröffentlichung online lesen: https://riptutorial.com/de/android/topic/9721/unterschreiben-sie-ihre-android-app-zur-veroffentlichung

Kapitel 244: Unterstützende Bildschirme mit unterschiedlichen Auflösungen, Größen

Bemerkungen

Begriffe und Konzepte

Bildschirmgröße

Tatsächliche physische Größe, gemessen als Diagonale des Bildschirms. Zur Vereinfachung gruppiert Android alle tatsächlichen Bildschirmgrößen in vier allgemeinen Größen: klein, normal, groß und extra groß.

Bildschirmdichte

Die Anzahl der Pixel in einem physischen Bereich des Bildschirms. normalerweise als dpi (Dots per Inch) bezeichnet. Beispielsweise hat ein Bildschirm mit "niedriger Dichte" weniger Pixel innerhalb eines gegebenen physikalischen Bereichs im Vergleich zu einem Bildschirm mit "normaler" oder "hoher Dichte". Zur Vereinfachung gruppiert Android alle tatsächlichen Bildschirmdichten in sechs allgemeine Dichten: niedrig, mittel, hoch, extra hoch, extra extra hoch und extra extra extra hoch.

Orientierung

Die Ausrichtung des Bildschirms aus Sicht des Benutzers. Dies ist entweder Querformat oder Hochformat, dh das Seitenverhältnis des Bildschirms ist entweder groß oder groß. Beachten Sie, dass verschiedene Geräte nicht nur standardmäßig in unterschiedlichen Ausrichtungen arbeiten. Die Ausrichtung kann sich zur Laufzeit ändern, wenn der Benutzer das Gerät dreht. Auflösung Die Gesamtzahl der physischen Pixel auf einem Bildschirm. Wenn Sie Unterstützung für mehrere Bildschirme hinzufügen, arbeiten Anwendungen nicht direkt mit der Auflösung. Die Anwendungen sollten sich nur mit der Bildschirmgröße und -dichte befassen, wie in den allgemeinen Größen- und Dichtegruppen angegeben. Dichteunabhängiges Pixel (dp) Eine virtuelle Pixeleinheit, die Sie beim Definieren des UI-Layouts verwenden sollten, um Layoutdimensionen oder -position auf dichtenunabhängige Weise anzugeben. Das dichteunabhängige Pixel entspricht einem physischen Pixel auf einem Bildschirm mit 160 dpi. Hierbei handelt es sich um die Basisliniendichte, die das System für einen Bildschirm mit "mittlerer" Dichte annimmt. Zur Laufzeit behandelt das System die Skalierung der dp-Einheiten je nach Bedarf auf der Grundlage der tatsächlichen Dichte des verwendeten Bildschirms transparent. Die Umwandlung von dp-Einheiten in Bildschirmpixel ist einfach: px = dp * (dpi / 160). Auf einem Bildschirm mit 240 dpi entspricht 1 dp beispielsweise 1,5 physikalischen Pixeln. Sie sollten immer

dp units verwenden, wenn Sie die Benutzeroberfläche Ihrer Anwendung definieren, um sicherzustellen, dass Ihre Benutzeroberfläche auf Bildschirmen mit unterschiedlichen Dichten angezeigt wird.

Einheiten

- **px**

  Pixel - entspricht den tatsächlichen Pixeln auf dem Bildschirm.

- **im**

  Zoll - basierend auf der physischen Größe des Bildschirms. 1 Zoll = 2,54 Zentimeter

- **mm**

  Millimeter - basierend auf der physischen Größe des Bildschirms.

- **pt**

  Punkte - 1/72 Zoll je nach Größe des Bildschirms.

- **dp oder dip**

  Dichteunabhängige Pixel - eine abstrakte Einheit, die auf der physischen Dichte des Bildschirms basiert. Diese Einheiten sind relativ zu einem Bildschirm mit 160 dpi, also ist ein Pixel ein Pixel auf einem Bildschirm mit 160 dpi. Das Verhältnis von dp zu Pixel ändert sich mit der Bildschirmdichte, jedoch nicht unbedingt in direktem Verhältnis. Hinweis: Der Compiler akzeptiert sowohl "dip" als auch "dp", obwohl "dp" eher mit "sp" übereinstimmt.

- **sp**

  Skalierungsunabhängige Pixel - Dies ist wie bei der dp-Einheit, wird jedoch auch nach der Einstellung der Schriftgröße des Benutzers skaliert. Es wird empfohlen, dieses Gerät bei der Angabe von Schriftgrößen zu verwenden, damit diese sowohl der Bildschirmdichte als auch den Vorlieben des Benutzers angepasst werden. Vom Verständnis der Dichteunabhängigkeit in Android:

Einheit	Beschreibung	Einheiten pro Zoll	Dichte unabhängig	Gleiche physische Größe auf jedem Bildschirm
px	Pixel	Variiert	Nein	Nein
im	Zoll	1	Ja	Ja
mm	Millimeter	25.4	Ja	Ja
pt	Punkte	72	Ja	Ja
dp	Dichte unabhängige Pixel	~ 160	Ja	Nein
sp	Unabhängige Pixel skalieren	~ 160	Ja	Nein

Verweise:

- https://developer.android.com/guide/practices/screens_support.html
- http://developer.android.com/guide/topics/resources/more-resources.html

Examples

Konfigurationsqualifizierer verwenden

Android unterstützt mehrere Konfigurationsqualifizierer, mit denen Sie steuern können, wie das System Ihre alternativen Ressourcen basierend auf den Eigenschaften des aktuellen Gerätebildschirms auswählt. Ein Konfigurationsqualifikationsmerkmal ist eine Zeichenfolge, die Sie an ein Ressourcenverzeichnis in Ihrem Android-Projekt anhängen können und die Konfiguration angibt, für die die darin enthaltenen Ressourcen entworfen wurden.

So verwenden Sie ein Konfigurationsqualifikationsmerkmal:

1. Erstellen Sie ein neues Verzeichnis im res / -Verzeichnis Ihres Projekts und benennen Sie es mit folgendem Format: `<resources_name>-<qualifier>` . `<resources_name>` ist der Standard-Ressourcenname (z. B. Zeichenbar oder Layout).
2. `<qualifier>` ist ein Konfigurationsqualifier, der die Bildschirmkonfiguration angibt, für die diese Ressourcen verwendet werden sollen (z. B. hdpi oder xlarge).

Beispielsweise bieten die folgenden Anwendungsressourcenverzeichnisse unterschiedliche Layoutdesigns für unterschiedliche Bildschirmgrößen und verschiedene Zeichenobjekte. Verwenden Sie die `mipmap/` Ordner für Launcher-Symbole.

```
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
```

Konvertieren von dp und sp in Pixel

Wenn Sie einen Pixelwert für etwas wie " `Paint.setTextSize` und dennoch auf der Grundlage des Geräts skaliert werden sollen, können Sie dp- und sp-Werte konvertieren.

```
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);
```

Alternativ können Sie eine Dimensionsressource in Pixel konvertieren, wenn Sie über einen Kontext zum Laden der Ressource verfügen.

```
<?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);
```

Textgröße und verschiedene Android-Bildschirmgrößen

Manchmal ist es besser, nur drei Optionen zu haben

```
 style="@android:style/TextAppearance.Small"
 style="@android:style/TextAppearance.Medium"
 style="@android:style/TextAppearance.Large"
```

Verwenden Sie klein und groß, um sich von der normalen Bildschirmgröße zu unterscheiden.

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="@android:style/TextAppearance.Small"/>
```

Normalerweise müssen Sie nichts angeben.

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

Auf diese Weise können Sie das Testen und Festlegen von Abmessungen für verschiedene Bildschirmgrößen vermeiden.

Unterstützende Bildschirme mit unterschiedlichen Auflösungen, Größen online lesen: https://riptutorial.com/de/android/topic/1086/unterstutzende-bildschirme-mit-unterschiedlichen-auflosungen--gro-en

Kapitel 245: VectorDrawable und AnimatedVectorDrawable

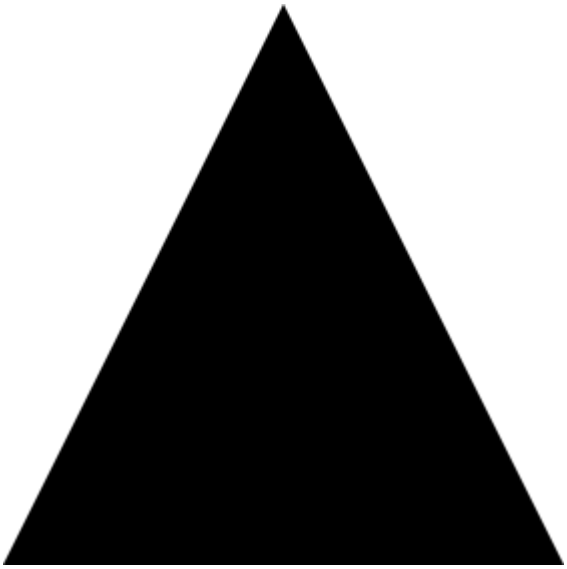
Examples

Grundlegende VectorDrawable

Ein `VectorDrawable` sollte aus mindestens einem `<path>` -Tag bestehen, das eine Form definiert

```
<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>
```

Dies würde ein schwarzes Dreieck erzeugen:



Verwenden

Ein `<clip-path>` definiert eine Form, die als Fenster fungiert, und lässt nur Teile eines `<path>` anzeigen, wenn sie sich innerhalb der `<clip-path>` -Form befinden und den Rest abschneiden.

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">
    <clip-path
```

```
        android:name="square clip path"
        android:pathData="M6,6 h12 v12 h-12 z"/>
    <path
        android:name="triangle"
        android:fillColor="#FF000000"
        android:pathData="M0,24 l12,-24 l12,24 z"/>

</vector>
```

In diesem Fall erzeugt der `<path>` ein schwarzes Dreieck, der `<clip-path>` definiert jedoch eine kleinere quadratische Form und lässt nur einen Teil des Dreiecks durchscheinen:

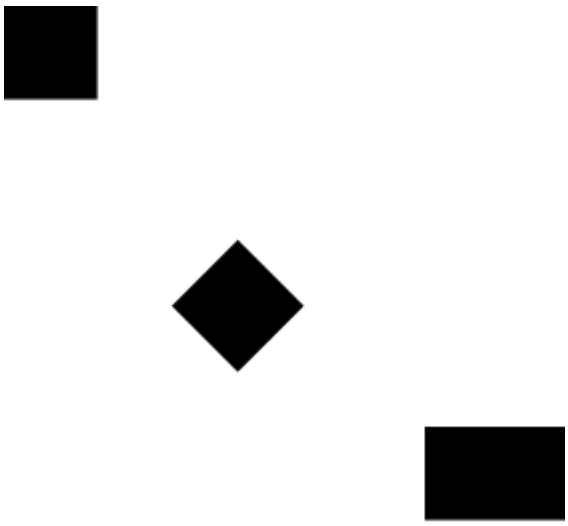
Stichworte

Mit einem `<group>` -Tag können Sie die Skalierung, Drehung und Position eines oder mehrerer Elemente eines `VectorDrawable` anpassen:

```
<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>
```

Der obige Beispielcode enthält drei identische `<path>` -Tags, die alle schwarze Quadrate beschreiben. Das erste Quadrat ist nicht angepasst. Das zweite Quadrat wird in ein `<group>` -Tag gehüllt, das es bewegt und um 45 ° dreht. Das dritte Quadrat wird in ein `<group>` -Tag eingeschlossen, das es bewegt und horizontal um 50% streckt. Das Ergebnis ist wie folgt:



Ein `<group>` -Tag kann mehrere `<path>` und `<clip-path>` -Tags enthalten. Es kann sogar eine andere `<group>` .

Grundlegende AnimatedVectorDrawable

Ein `AnimatedVectorDrawable` erfordert mindestens 3 Komponenten:

- Eine `VectorDrawable` die `VectorDrawable` wird
- Ein `objectAnimator` der definiert, welche Eigenschaft geändert werden soll und wie
- Das `AnimatedVectorDrawable` selbst, das den `objectAnimator` mit dem `VectorDrawable` , um die Animation zu erstellen

Im Folgenden wird ein Dreieck erstellt, das seine Farbe von Schwarz nach Rot übergibt.

Die `VectorDrawable` , Dateiname: `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>
```

Der `objectAnimator` , Dateiname: `color_change_animator.xml`

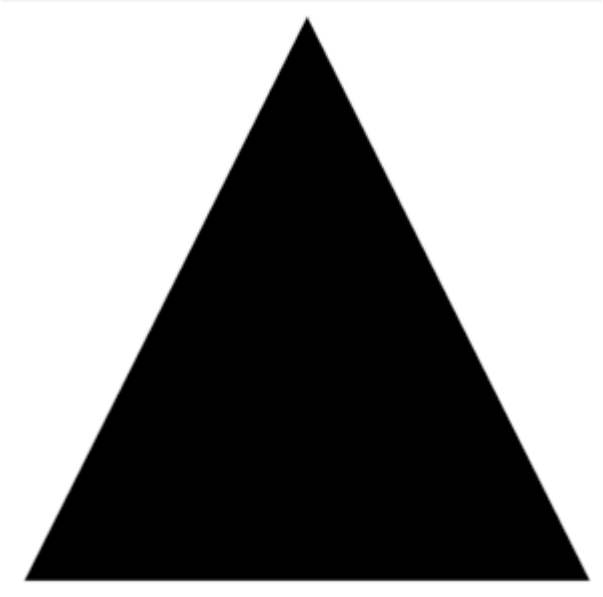
```
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:propertyName="fillColor"
    android:duration="2000"
    android:repeatCount="infinite"
    android:valueFrom="@android:color/black"
    android:valueTo="@android:color/holo_red_light"/>
```

Die `AnimatedVectorDrawable` , Dateiname: `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>
```

Beachten Sie, dass das `<target>` `android:name="triangle"` angibt, das dem `<path>` in der `VectorDrawable` . Eine `VectorDrawable` kann mehrere Elemente enthalten. Mit der Eigenschaft `android:name` wird festgelegt, auf welches Element abgezielt werden soll.

Ergebnis:



Striche verwenden

Die Verwendung des SVG-Strichs erleichtert das Erstellen eines Vektors, der mit einer einheitlichen Hublänge gemäß den Material Design-Richtlinien gezeichnet werden kann :

> Konsistente Strichgewichte sind der Schlüssel zur Vereinheitlichung der gesamten Systemsymbolfamilie. Behalten Sie eine Breite von 2 dp für alle Striche, einschließlich Kurven, Winkel sowie innere und äußere Striche bei.

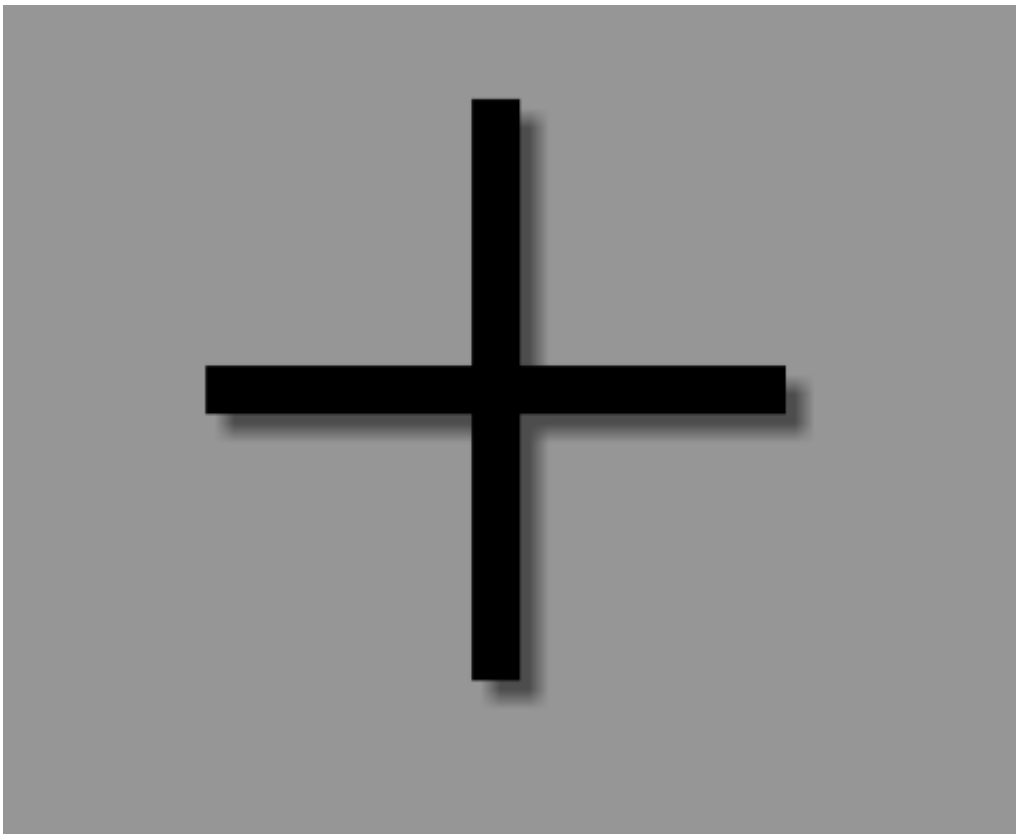
So erstellen Sie beispielsweise mit Strichen ein Pluszeichen:

```
<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` definiert die Farbe des Strichs.
- `strokeWidth` definiert die Breite (in dp) des Strichs (in diesem Fall 2dp, wie von den Richtlinien vorgeschlagen).
- `pathData` beschreiben wir unser SVG-Image:
- `M12,0` bewegt den "Cursor" an die Position 12,0
- `V24` erzeugt eine vertikale Linie zur Position 12, 24

usw. finden Sie in der SVG-Dokumentation und in diesem nützlichen Tutorial "SVG-Pfad" von w3schools , um mehr über die spezifischen Pfadbefehle zu erfahren.

Als Ergebnis erhielten wir dieses No-Frills-Plus-Zeichen:



Dies ist **besonders nützlich** , wenn Sie ein `AnimatedVectorDrawable` erstellen `AnimatedVectorDrawable` , da Sie jetzt mit einem einzelnen Strich mit einer einheitlichen Länge anstelle eines sonst komplizierten Pfads arbeiten.

Vektorkompatibilität durch AppCompat

Einige Voraussetzungen im `build.gradle` für Vektoren, die bis API 7 für VectorDrawables und API 13 für AnimatedVectorDrawables funktionieren (mit einigen Einschränkungen):

```
//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'
}
```

In Ihrer `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" />
```

VectorDrawable und AnimatedVectorDrawable online lesen:
https://riptutorial.com/de/android/topic/1627/vectordrawable-und-animatedvectordrawable

Kapitel 246: Vektor Drawables

Einführung

Wie der Name schon sagt, basieren Vektorzeichnungen auf Vektorgrafiken. Vektorgrafiken sind eine Möglichkeit, grafische Elemente mit geometrischen Formen zu beschreiben. Auf diese Weise können Sie eine Zeichnung erstellen, die auf einer XML-Vektorgrafik basiert. Jetzt müssen Sie kein Bild mit unterschiedlicher Größe für mdpi, hdpi, xhdpi usw. erstellen. Mit Vector Drawable müssen Sie das Bild nur einmal als xml-Datei erstellen und es für alle dpi und verschiedene Geräte skalieren. Dies spart nicht nur Platz, sondern vereinfacht auch die Wartung.

Parameter

Parameter	Einzelheiten
`<vector>`	Wird verwendet, um einen Vektor zu definieren, der gezeichnet werden kann
`<group>`	Definiert eine Gruppe von Pfaden oder Untergruppen sowie Transformationsinformationen. Die Transformationen werden in denselben Koordinaten wie das Ansichtsfenster definiert. Die Transformationen werden in der Reihenfolge der Skalierung angewendet, drehen und verschieben.
`<path>`	Definiert die zu zeichnenden Pfade.
`<clip-path>`	Definiert den Pfad als aktuellen Clip. Beachten Sie, dass der Clippfad nur für die aktuelle Gruppe und ihre untergeordneten Objekte gilt.

Bemerkungen

Aktualisieren **Sie die** Datei **build.gradle** .

```
dependencies {
    ...
   compile 'com.android.support:appcompat-v7:23.2.1'
}
```

Wenn Sie **Version 2.0 oder höher** des **Gradle-Plugins verwenden** , fügen Sie den folgenden Code hinzu.

```
// Gradle Plugin 2.0+
 android {
   defaultConfig {
     vectorDrawables.useSupportLibrary = true
    }
 }
```

Wenn Sie **Version 1.5 oder niedriger** des **Gradle-Plugins verwenden** , fügen Sie den folgenden Code hinzu.

```
// Gradle Plugin 1.5
 android {
   defaultConfig {
     generatedDensities = []
  }

  // This is handled for you by the 2.0+ Gradle Plugin
  aaptOptions {
    additionalParameters "--no-version-vectors"
  }
 }
```

Weitere Informationen finden Sie in der Android Support Library 23.2 Versionshinweise.

HINWEIS: Auch mit *AppCompat funktioniert* Vector Drawables in älteren Android-Versionen nicht außerhalb Ihrer App. Sie können beispielsweise keine Vektorzeichnungen als Benachrichtigungssymbole übergeben, da diese vom System und nicht von der App verarbeitet werden. Siehe diese Antwort für eine Problemumgehung.

Examples

VectorDrawable-Verwendungsbeispiel

Hier ist ein Beispiel für ein Vektor-Asset, das wir in AppCompat verwenden:

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>
```

Bei Verwendung dieser Zeichenweise wäre eine `ImageView` Deklaration beispielsweise:

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@drawable/ic_search"/>
```

Sie können es auch zur Laufzeit einstellen:

```
ImageView iv = (ImageView) findViewById(...);
iv.setImageResource(R.drawable.ic_search);
```

Dasselbe Attribut und Aufrufe funktionieren auch für `ImageButton` .

VectorDrawable-XML-Beispiel

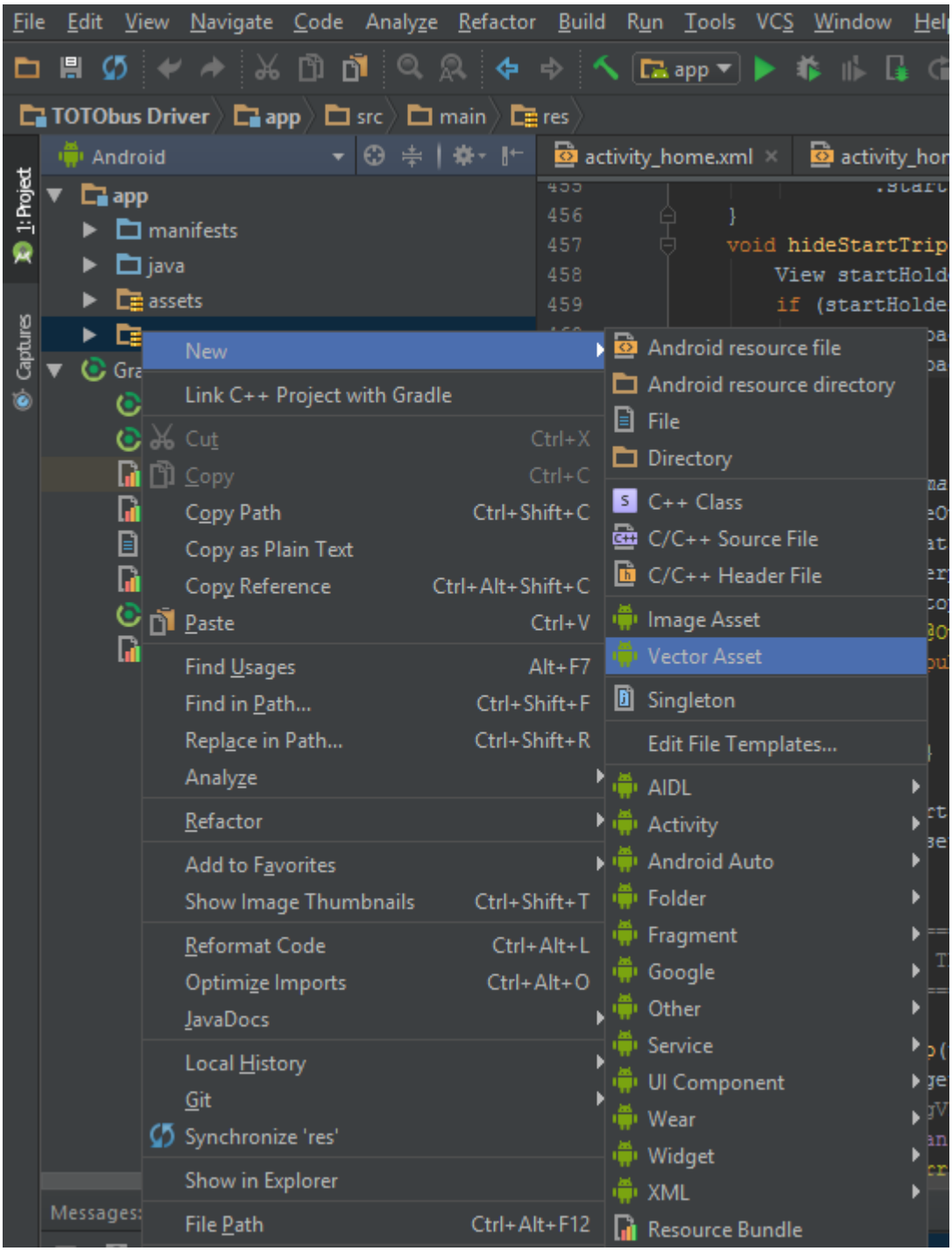
Hier ist eine einfache `VectorDrawable` in dieser Datei **vectordrawable.xml** .

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="64dp"
    android:width="64dp"
    android:viewportHeight="600"
    android:viewportWidth="600" >
    <group
        android:name="rotationGroup"
        android:pivotX="300.0"
        android:pivotY="300.0"
        android:rotation="45.0" >
        <path
            android:name="v"
            android:fillColor="#000000"
            android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
    </group>
</vector>
```

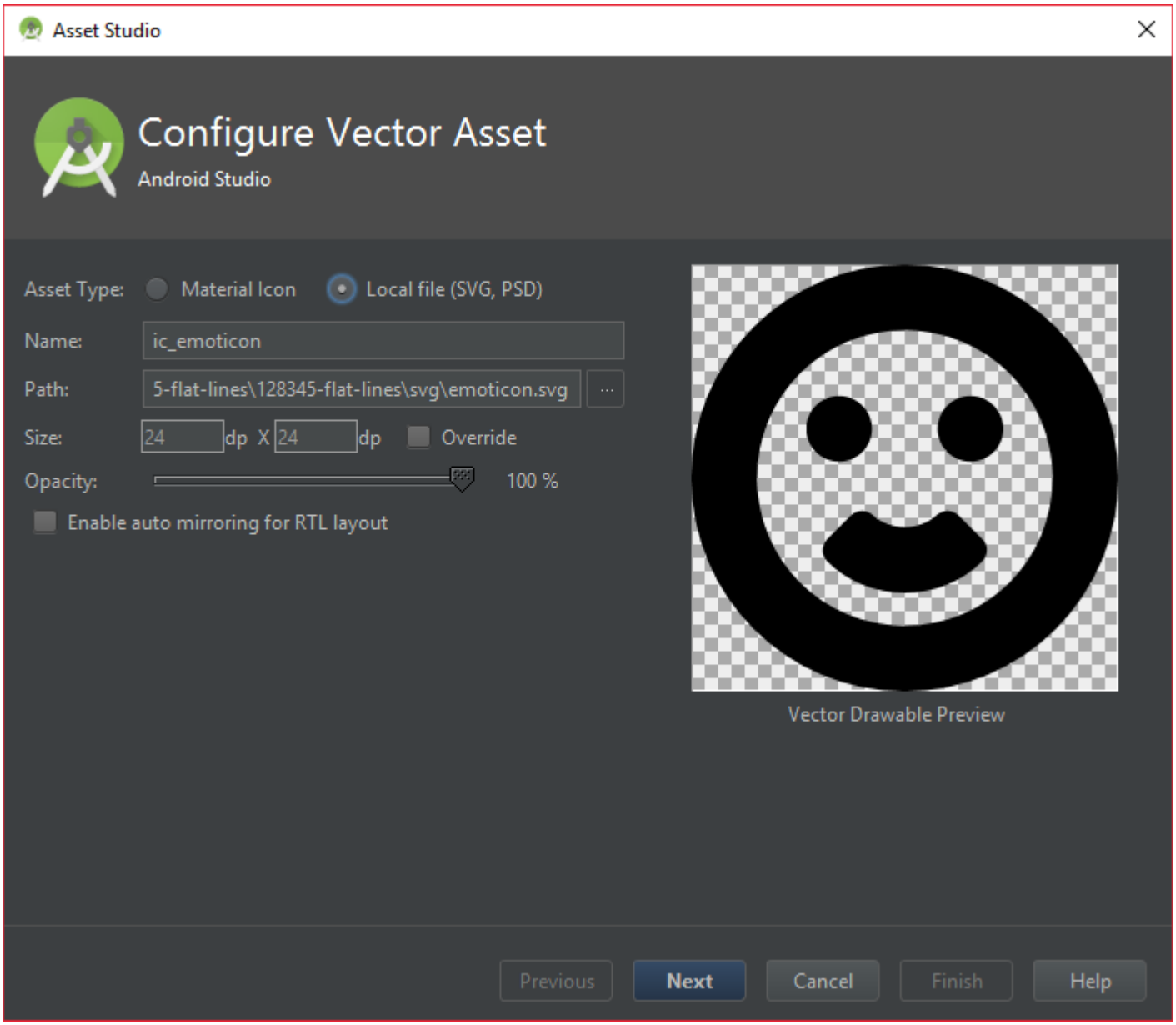
SVG-Datei als VectorDrawable importieren

Sie können eine *SVG*- Datei als `VectorDrawable` in Android Studio importieren. `VectorDrawable` folgendermaßen vor:

Klicken Sie mit der rechten Maustaste auf den Ordner " `res` " und wählen Sie " **Neu"** > " **Vektor-Asset"** .



Wählen Sie die Option **Lokale Datei** und navigieren Sie zu Ihrer SVG-Datei. Ändern Sie die Optionen nach Ihren Wünschen und klicken Sie auf Weiter. Erledigt.



Vektor Drawables online lesen: https://riptutorial.com/de/android/topic/8194/vektor-drawables

Kapitel 247: Verbesserung der Android-Leistung mithilfe von Symbol-Schriftarten

Bemerkungen

Symbolschriftarten sind wie normale Schriftarten, die Symbole anstelle von Buchstaben haben. Es kann in Ihrer Anwendung mit Leichtigkeit verwendet werden.

Sie sind:

- Flexibel
- Skalierbar
- Vektoren
- Schnell verarbeitbar
- Leicht
- Zugänglich

Auswirkungen auf die Größe

Das Exportieren eines Bildes in verschiedenen Größen für Android-Geräte würde Ihre App zusätzlich kosten, etwa 30 KB pro Bild. Das Hinzufügen einer Schriftartdatei (.ttf) mit etwa 36 Symbolen würde nur 9 KB kosten. Stellen Sie sich nur den Fall vor, wenn Sie 36 einzelne Dateien mit verschiedenen Konfigurationen hinzufügen, sie wären etwa 1000 KB groß. Es ist ein angemessener Speicherplatz, den Sie durch die Verwendung von Symbolschriftarten sparen.

Einschränkungen von Icon-Schriftarten.

- Symbolzeichensätze können in der Navigationsleiste verwendet werden. Die Verwendung in Navigationsansichten als Symbol für Menüelemente ist nicht möglich, da die Menüdatei nicht ohne Angabe des Titels erstellt werden kann. Daher ist es ratsam, SVG-Dateien als Ressourcen für diese Symbole zu verwenden.
- Symbolfonts können nicht in Floating-Aktionsschaltflächen verwendet werden. da sie kein `setText()` -Attribut haben.
- Externe Schriftarten können nicht aus XML übernommen werden. Sie müssen mit der Java-Datei angegeben werden. Oder Sie müssen die Basisansicht erweitern und eine Ansicht erstellen, wie in diesem Beitrag angegeben

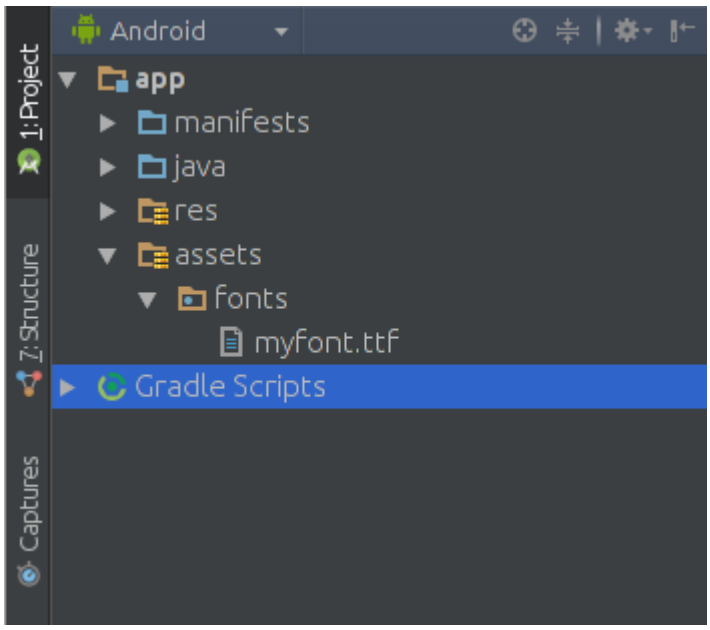
Examples

So integrieren Sie Icon-Schriftarten

Um die Symbolschriftarten zu verwenden, folgen Sie einfach den folgenden Schritten:

- **Fügen Sie die Schriftartdatei Ihrem Projekt hinzu**

  Sie können Ihre Schriftart- Symboldatei von Online-Websites wie z. B. icomoon erstellen , auf der Sie SVG-Dateien mit den erforderlichen Symbolen hochladen und dann die erstellte Symbolschriftart herunterladen können. *Legen Sie* dann die *.ttf-* Schriftartdatei in einen Ordner mit dem Namen *fonts* (benennen Sie ihn nach Belieben) im Assets-Ordner ab:



- **Erstellen Sie eine Helferklasse**

  Erstellen Sie nun die folgende Hilfsklasse, damit Sie den Initialisierungscode für die Schriftart nicht wiederholen müssen:

```
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);
  }
}
```

  Sie können die `Typeface` Klasse verwenden, um die Schriftart aus den Assets auszuwählen. Auf diese Weise können Sie die Schriftart auf verschiedene Ansichten einstellen, beispielsweise auf eine Schaltfläche:

```
Button button=(Button) findViewById(R.id.button);
Typeface iconFont=FontManager.getTypeface(getApplicationContext());
button.setTypeface(iconFont);
```

  Die Schaltflächenschriftart wurde nun in die neu erstellte Symbolschriftart geändert.

- **Heben Sie die gewünschten Symbole auf**

  Öffnen Sie die Datei *styles.css* , die der *Symbolschriftart* angehängt ist. Dort finden Sie die Stile mit Unicode-Zeichen Ihrer Symbole:

```
.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";
}
```

Diese Ressourcendatei dient als Wörterbuch, in dem das mit einem bestimmten Symbol verknüpfte Unicode-Zeichen einem vom Menschen lesbaren Namen zugeordnet wird. Erstellen Sie nun die Zeichenfolgenressourcen wie folgt:

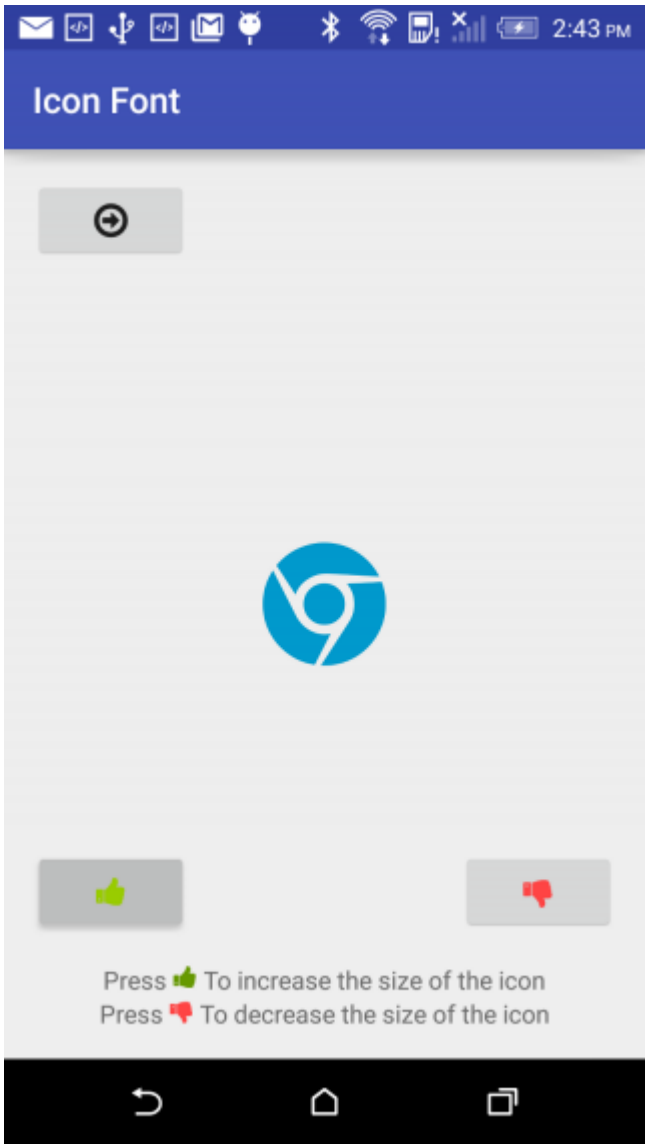
```
<resources>
  <! — Icon Fonts -->
  <string name="icon_arrow_circle_down">&#xe001; </string>
  <string name="icon_arrow_circle_left">&#xe002; </string>
  <string name="icon_arrow_circle-o_down">&#xe003; </string>
  <string name="icon_arrow_circle_o_left">&#xe004; </string>
</resources>
```

- **Verwenden Sie die Symbole in Ihrem Code**

  Nun können Sie Ihre Schrift beispielsweise in folgenden Ansichten verwenden:

```
button.setText(getString(R.string.icon_arrow_circle_left))
```

  Sie können Schaltflächentextansichten auch mit Symbolschriftarten erstellen:



TabLayout mit Symbolschriftarten

```
public class TabAdapter extends FragmentPagerAdapter {

CustomTypefaceSpan fonte;
List<Fragment> fragments = new ArrayList<>(4);
private String[] icons = {"\ue001","\uE002","\uE003","\uE004"};

public TabAdapter(FragmentManager fm, CustomTypefaceSpan fonte) {
    super(fm);
    this.fonte = fonte
    for (int i = 0; i < 4; i++){
        fragments.add(MyFragment.newInstance());
    }
}

public List<Fragment> getFragments() {
    return fragments;
}

@Override
public Fragment getItem(int position) {
    return fragments.get(position);
}
```

```
@Override
public CharSequence getPageTitle(int position) {
    SpannableStringBuilder ss = new SpannableStringBuilder(icons[position]);
    ss.setSpan(fonte,0,ss.length(), Spanned.SPAN_INCLUSIVE_INCLUSIVE);
    ss.setSpan(new RelativeSizeSpan(1.5f),0,ss.length(), Spanned.SPAN_INCLUSIVE_INCLUSIVE );
    return ss;
}



@Override
public int getCount() {
    return 4;
}

}
```

- In diesem Beispiel befindet sich myfont.ttf im Ordner Assets. Ordner "Assets" erstellen
- In deiner Aktivitätsklasse

```
//..
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);
    //..
```

Verbesserung der Android-Leistung mithilfe von Symbol-Schriftarten online lesen: https://riptutorial.com/de/android/topic/3642/verbesserung-der-android-leistung-mithilfe-von-symbol-schriftarten

Kapitel 248: Veröffentlichen Sie die .aar-Datei mit Gradle in Apache Archiva

Examples

Einfaches Implementierungsbeispiel

```
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'
            }
        }
    }

}
```

Veröffentlichen Sie die .aar-Datei mit Gradle in Apache Archiva online lesen: https://riptutorial.com/de/android/topic/6453/veroffentlichen-sie-die--aar-datei-mit-gradle-in-apache-archiva

Kapitel 249: Veröffentlichen Sie eine Bibliothek in Maven Repositories

Examples

Veröffentlichen Sie die .aar-Datei in Maven

Um in einem Repository im Maven-Format veröffentlichen zu können, kann das Plugin „maven-publish“ für Gradle verwendet werden.

Das Plugin sollte zur `build.gradle` Datei im Bibliotheksmodul hinzugefügt werden.

```
apply plugin: 'maven-publish'
```

Sie sollten die Publikation und ihre Identitätsattribute auch in der Datei `build.gradle` definieren. Diese Identitätsattribute werden in der generierten POM-Datei angezeigt. Zum Importieren dieser Publikation werden Sie sie in Zukunft verwenden. Sie müssen auch definieren, welche Artefakte Sie veröffentlichen möchten. Zum Beispiel möchte ich die generierte .aar-Datei nach dem Erstellen der Bibliothek veröffentlichen .

```
publishing {
    publications {
        myPulication(MavenPublication) {
            groupId 'com.example.project'
            version '1.0.2'
            artifactId 'myProject'
            artifact("$buildDir/outputs/aar/myProject.aar")
        }
    }
}
```

Sie müssen auch Ihre Repository-URL definieren

```
publishing{
    repositories {
        maven {
            url "http://www.myrepository.com"
        }
    }
}
```

Hier ist die vollständige Bibliothek `build.gradle` Datei

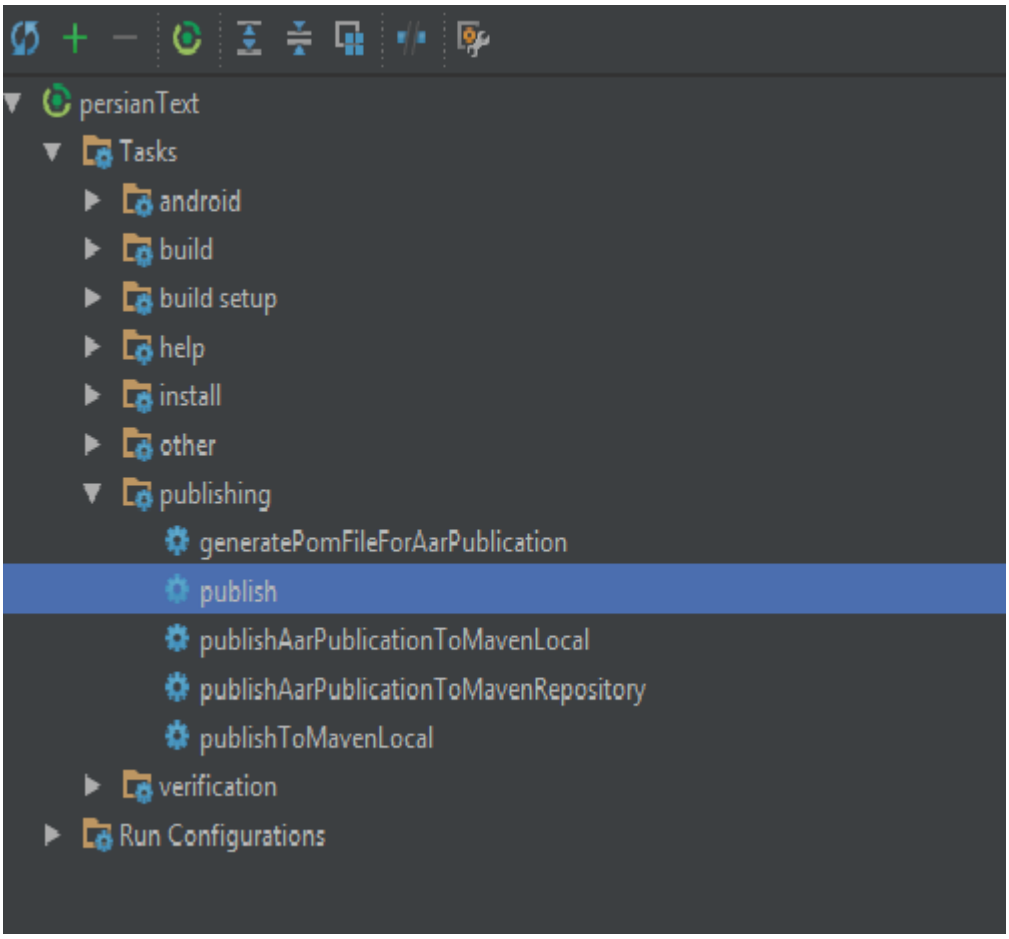
```
apply plugin: 'com.android.library'
apply plugin: 'maven-publish'

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"
        }
    }
}
```

Zum Veröffentlichen können Sie den Befehl gradle console ausführen

```
gradle publizieren
```

oder Sie können von der Gradle-Taskleiste aus starten



Veröffentlichen Sie eine Bibliothek in Maven Repositories online lesen: https://riptutorial.com/de/android/topic/9359/veroffentlichen-sie-eine-bibliothek-in-maven-repositories

Kapitel 250: Vibration

Examples

Erste Schritte mit Vibration

Vibrationsgenehmigung gewähren

Bevor Sie mit dem Implementieren von Code beginnen, müssen Sie die Berechtigung für das Android-Manifest hinzufügen:

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

Vibration Library importieren

```
import android.os.Vibrator;
```

Rufen Sie die Instanz von Vibrator aus dem Kontext ab

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

Überprüfen Sie, ob das Gerät einen Vibrator hat

```
void boolean isHaveVibrate(){
    if (vibrator.hasVibrator()) {
        return true;
    }
    return false;
}
```

Unbegrenzt vibrieren

mit dem *vibrieren (langes [] Muster, int Wiederholung)*

```
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);
```

Vibrationsmuster

Sie können Vibrationsmuster erstellen, indem Sie ein Array von Longs übergeben, von denen jedes eine Dauer in Millisekunden darstellt. Die erste Zahl ist die Startzeitverzögerung. Jeder

Feldeintrag wechselt dann zwischen Vibration, Schlaf, Vibration, Schlaf usw.

Das folgende Beispiel veranschaulicht dieses Muster:

- Vibrieren Sie 100 Millisekunden und schlafen Sie 1000 Millisekunden
- 200 Millisekunden vibrieren und 2000 Millisekunden schlafen

```
long[] pattern = {0, 100, 1000, 200, 2000};
```

Um das Muster zu wiederholen, übergeben Sie den Index in das Musterfeld, an dem die Wiederholung beginnen soll, oder `-1` , um die Wiederholung zu deaktivieren.

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(pattern, -1); // does not repeat
vibrator.vibrate(pattern,  0); // repeats forever
```

Stoppen Sie zu vibrieren

Wenn Sie aufhören zu vibrieren, rufen Sie bitte an:

```
vibrator.cancel();
```

Einmal vibrieren

mit dem *vibrieren (lange Millisekunden)*

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(500);
```

Vibration online lesen: https://riptutorial.com/de/android/topic/3359/vibration

Kapitel 251: VideoView

Examples

VideoView Create

Suchen Sie VideoView in Activity und fügen Sie Video hinzu.

```
VideoView videoView = (VideoView) .findViewById(R.id.videoView);
videoView.setVideoPath(pathToVideo);
```

Video abspielen

```
videoView.start();
```

Definieren Sie VideoView in der XML-Layoutdatei.

```
<VideoView
    android:id="@+id/videoView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center" />
```

Geben Sie das Video über die URL mit VideoView wieder

```
 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;
        }
});
```

VideoView online lesen: https://riptutorial.com/de/android/topic/8962/videoview

Kapitel 252: ViewFlipper

Einführung

Ein `ViewFlipper` ist ein `ViewAnimator` , der zwischen zwei oder mehr hinzugefügten Ansichten wechselt. Es wird jeweils nur ein Kind angezeigt. Auf Wunsch kann der `ViewFlipper` in regelmäßigen Abständen automatisch zwischen den einzelnen Kindern `ViewFlipper` .

Examples

ViewFlipper mit Bildschieben

XML-Datei:

```
<ViewFlipper
    android:id="@+id/viewflip"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:layout_weight="1"
    />
```

JAVA-Code:

```
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);
    }
}
```

ViewFlipper online lesen: https://riptutorial.com/de/android/topic/9032/viewflipper

Kapitel 253: ViewPager

Einführung

ViewPager ist ein Layout-Manager, mit dem der Benutzer nach links und rechts durch Datenseiten blättern kann. Sie wird meistens in Verbindung mit Fragment verwendet. Dies ist eine bequeme Möglichkeit, den Lebenszyklus jeder Seite bereitzustellen und zu verwalten.

Bemerkungen

Wichtig bei der Verwendung von ViewPager ist, dass es zwei verschiedene Versionen von `FragmentPagerAdapter` und `FragmentStatePagerAdapter` .

Wenn Sie `android.app.Fragment` native Fragmente mit einem FragmentPagerAdapter oder FragmentStatePagerAdapter verwenden, müssen Sie die v13-unterstützten Bibliotheksversionen des Adapters verwenden, `android.support.v13.app.FragmentStatePagerAdapter` . B. `android.support.v13.app.FragmentStatePagerAdapter` .

Wenn Sie die `android.support.v4.app.Fragment` Unterstützungsbibliothek Fragmente mit einem FragmentPagerAdapter oder FragmentStatePagerAdapter verwenden, müssen Sie die v4-Unterstützungsbibliothek-Versionen des Adapters verwenden, `android.support.v4.app.FragmentStatePagerAdapter` . B. `android.support.v4.app.FragmentStatePagerAdapter` .

Examples

Grundlegende ViewPager-Verwendung mit Fragmenten

Ein `ViewPager` ermöglicht das `ViewPager` mehrerer Fragmente in einer Aktivität, die durch Umblättern nach links oder rechts navigiert werden kann. Ein `ViewPager` muss mit einem `PagerAdapter` entweder Ansichten oder Fragmente `PagerAdapter` .

Es gibt jedoch zwei spezifischere Implementierungen, die für den Fall der Verwendung von Fragmenten (`FragmentPagerAdapter` und `FragmentStatePagerAdapter` am nützlichsten sind. Wenn ein Fragment zum ersten Mal instanziiert werden muss, wird `getItem(position)` für jede Position aufgerufen, die instantiiert werden muss. Die Methode `getCount()` gibt die Gesamtzahl der Seiten zurück, damit der `ViewPager` weiß, wie viele Fragmente angezeigt werden müssen.

Sowohl `FragmentPagerAdapter` als auch `FragmentStatePagerAdapter` bewahren einen Cache der Fragmente auf, die der `ViewPager` muss. Standardmäßig versucht der `ViewPager` , maximal 3 Fragmente zu speichern, die dem aktuell sichtbaren Fragment entsprechen, und den Fragmenten rechts und links davon. `FragmentStatePagerAdapter` behält auch den Status aller Ihrer Fragmente bei.

Beachten Sie, dass bei beiden Implementierungen davon ausgegangen wird, dass Ihre Fragmente

ihre Positionen `getItem()` Wenn Sie also eine Liste der Fragmente `getItem()` und keine statische Anzahl davon haben, wie Sie in der `getItem()` -Methode sehen können, müssen Sie eine Unterklasse von `PagerAdapter` und überschreiben Sie mindestens die Methoden `instantiateItem()` , `destroyItem()` und `getItemPosition()` .

Fügen Sie einfach einen ViewPager in Ihr Layout ein, wie im grundlegenden Beispiel beschrieben :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>
    <android.support.v4.view.ViewPager
        android:id="@+id/vpPager">
    </android.support.v4.view.ViewPager>
</LinearLayout>
```

Definieren Sie dann den Adapter, der bestimmt, wie viele Seiten vorhanden sind und welches Fragment für jede Seite des Adapters angezeigt werden soll.

```
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

Wenn Sie `android.app.Fragment` , müssen Sie diese Abhängigkeit hinzufügen:

```
compile 'com.android.support:support-v13:25.3.1'
```

Wenn Sie `android.support.v4.app.Fragment` , müssen Sie diese Abhängigkeit hinzufügen:

```
compile 'com.android.support:support-fragment:25.3.1'
```

ViewPager mit TabLayout

Ein `TabLayout` kann zur einfacheren Navigation verwendet werden.
Sie können die Registerkarten für jedes Fragment in Ihrem Adapter mithilfe der `TabLayout.newTab()` Methode `TabLayout.newTab()` gibt jedoch eine andere bequemere und einfachere Methode für diese Aufgabe, `TabLayout.setupWithViewPager()` .

Diese Methode wird durch das Erstellen und Entfernen von Registerkarten entsprechend dem Inhalt des mit Ihrem `ViewPager` Adapters bei jedem Aufruf `ViewPager` .
Außerdem wird ein Rückruf festgelegt, sodass jedes Mal, wenn der Benutzer die Seite umblättert, die entsprechende Registerkarte ausgewählt wird.

Definieren Sie einfach ein Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>

    <android.support.design.widget.TabLayout
        android:id="@+id/tabs"
        app:tabMode="scrollable" />

    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager"
        android:layout_width="match_parent"
        android:layout_height="0px"
        android:layout_weight="1" />

</LinearLayout>
```

Implementieren Sie dann den `FragmentPagerAdapter` und wenden Sie ihn auf den `ViewPager` :

```
public class MyViewPagerActivity extends AppCompatActivity {
    private static final String TAG = MyViewPagerActivity.class.getName();

    private MyPagerAdapter mFragmentAdapter;
    private ViewPager mViewPager;
    private TabLayout mTabLayout;

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

        // Get the ViewPager and apply the PagerAdapter
        mFragmentAdapter = new MyPagerAdapter(getSupportFragmentManager());
        mViewPager = (ViewPager) findViewById(R.id.view_pager);
        mViewPager.setAdapter(mFragmentAdapter);

        // link the tabLayout and the viewpager together
        mTabLayout = (TabLayout) findViewById(R.id.tab_layout);
        mTabLayout.setupWithViewPager(mViewPager);
    }

    private class MyPagerAdapter extends FragmentPagerAdapter{

        public MyPagerAdapter(FragmentManager supportFragmentManager) {
            super(supportFragmentManager);
        }

         // Returns the fragment to display for that page
        @Override
        public Fragment getItem(int position) {
            switch(position) {
                case 0:
                    return new Fragment1();

                case 1:
                    return new Fragment2();

                case 2:
                    return new Fragment3();

                default:
                    return null;
            }
        }

        // Will be displayed as the tab's label
        @Override
        public CharSequence getPageTitle(int position) {
            switch(position) {
                case 0:
                    return "Fragment 1 title";

                case 1:
                    return "Fragment 2 title";

                case 2:
                    return "Fragment 3 title";

                default:
                    return null;
```

```
            }
        }

        // Returns total number of pages
        @Override
        public int getCount() {
            return 3;
        }

    }
}
```

ViewPager mit PreferenceFragment

Bis vor kurzem verhinderte die Verwendung von `android.support.v4.app.FragmentPagerAdapter` die Verwendung eines `PreferenceFragment` als eines der im FragmentPagerAdapter verwendeten Fragmente.

Die neuesten Versionen der Support-v7-Bibliothek enthalten jetzt die `PreferenceFragmentCompat` Klasse, die mit einem ViewPager und der v4-Version von FragmentPagerAdapter funktioniert.

Beispielfragment, das `PreferenceFragmentCompat` :

```
import android.os.Bundle;
import android.support.v7.preference.PreferenceFragmentCompat;
import android.view.View;

public class MySettingsPrefFragment extends PreferenceFragmentCompat {

    public MySettingsPrefFragment() {
        // Required empty public constructor
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.fragment_settings_pref);
    }

    @Override
    public void onCreatePreferences(Bundle bundle, String s) {

    }
}
```

Sie können dieses Fragment jetzt in einer Unterklasse `android.support.v4.app.FragmentPagerAdapter` :

```
private class PagerAdapterWithSettings extends FragmentPagerAdapter {

    public PagerAdapterWithSettings(FragmentManager supportFragmentManager) {
        super(supportFragmentManager);
    }

    @Override
    public Fragment getItem(int position) {
```

```
        switch(position) {
            case 0:
                return new FragmentOne();

            case 1:
                return new FragmentTwo();

            case 2:
                return new MySettingsPrefFragment();

            default:
                return null;
        }
    }

    // .......

}
```

ViewPager hinzufügen

`build.gradle` Sie sicher, dass die folgende Abhängigkeit zur `build.gradle` Datei Ihrer App unter Abhängigkeiten hinzugefügt wird:

```
compile 'com.android.support:support-core-ui:25.3.0'
```

`ViewPager` Sie dann den `ViewPager` zu Ihrem Aktivitätslayout hinzu:

```
<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
```

Dann definieren Sie Ihren `PagerAdapter` :

```
public class MyPagerAdapter extends PagerAdapter {

    private Context mContext;

    public CustomPagerAdapter(Context context) {
        mContext = context;
    }

    @Override
    public Object instantiateItem(ViewGroup collection, int position) {

        // Create the page for the given position. For example:
        LayoutInflater inflater = LayoutInflater.from(mContext);
        ViewGroup layout = (ViewGroup) inflater.inflate(R.layout.xxxx, collection, false);
        collection.addView(layout);
        return layout;
    }

    @Override
    public void destroyItem(ViewGroup collection, int position, Object view) {
```

```
        // Remove a page for the given position. For example:
        collection.removeView((View) view);
    }

    @Override
    public int getCount() {
        //Return the number of views available.
        return numberOfPages;
    }

    @Override
    public boolean isViewFromObject(View view, Object object) {
        // Determines whether a page View is associated with a specific key object
        // as returned by instantiateItem(ViewGroup, int). For example:
        return view == object;
    }
}
```

`ViewPager` Sie schließlich den `ViewPager` in Ihrer Aktivität ein:

```
public class MainActivity extends AppCompatActivity {

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

        ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);
        viewPager.setAdapter(new MyPagerAdapter(this));
    }
}
```

ViewPager mit Punktanzeige

Alles was wir brauchen sind: ViewPager , TabLayout und 2 Drawables für ausgewählte und

Standardpunkte.

Zuerst müssen wir `TabLayout` zu unserem Bildschirmlayout hinzufügen und es mit `ViewPager` . Wir können dies auf zwei Arten tun:

Verschachteltes TabLayout in ViewPager

```
<android.support.v4.view.ViewPager
    android:id="@+id/photos_viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.design.widget.TabLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</android.support.v4.view.ViewPager>
```

In diesem Fall wird `TabLayout` automatisch mit `ViewPager` , aber `TabLayout` befindet sich neben `ViewPager` und nicht über ihm.

Separates 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"/>
```

In diesem Fall können wir setzen `TabLayout` überall, aber wir haben eine Verbindung `TabLayout` mit `ViewPager` programmatisch

```
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);
```

Nachdem wir unser Layout erstellt haben, müssen wir unsere Punkte vorbereiten. Wir erstellen also drei Dateien: `selected_dot.xml` , `default_dot.xml` und `tab_selector.xml` .

---

selected_dot.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape
            android:innerRadius="0dp"
            android:shape="ring"
            android:thickness="8dp"
            android:useLevel="false">
            <solid android:color="@color/colorAccent"/>
        </shape>
    </item>
</layer-list>
```

---

default_dot.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape
            android:innerRadius="0dp"
            android:shape="ring"
            android:thickness="8dp"
            android:useLevel="false">
            <solid android:color="@android:color/darker_gray"/>
        </shape>
    </item>
</layer-list>
```

---

tab_selector.xml

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

    <item android:drawable="@drawable/selected_dot"
          android:state_selected="true"/>

    <item android:drawable="@drawable/default_dot"/>
</selector>
```

Jetzt müssen wir nur noch 3 Zeilen Code zu `TabLayout` in unserem XML-Layout hinzufügen und fertig.

```
app:tabBackground="@drawable/tab_selector"
app:tabGravity="center"
app:tabIndicatorHeight="0dp"
```

Richten Sie OnPageChangeListener ein

Wenn Sie Änderungen an der ausgewählten Seite `ViewPager.OnPageChangeListener` möchten, können Sie den `ViewPager.OnPageChangeListener` Listener im ViewPager implementieren:

```
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
    }
});
```

ViewPager online lesen: https://riptutorial.com/de/android/topic/692/viewpager

Kapitel 254: Volley

Einführung

Volley ist eine Android-HTTP-Bibliothek, die von Google eingeführt wurde, um Netzwerkanrufe zu vereinfachen. Standardmäßig werden alle Volley-Netzwerkaufrufe asynchron ausgeführt. Dabei wird alles in einem Hintergrundthread verarbeitet und die Ergebnisse mithilfe von Rückrufen in den Vordergrund zurückgegeben. Da das Abrufen von Daten über ein Netzwerk eine der häufigsten Aufgaben ist, die in einer App ausgeführt werden, wurde die Volley-Bibliothek zur Entwicklung der Android-App entwickelt.

Syntax

- RequestQueue-Warteschlange = Volley.newRequestQueue (Kontext); // die Warteschlange einrichten
- Request request = new SomeKindOfRequestClass (Request.Method, String-URL, Response.Listener, Response.ErrorListener); // eine Art Anforderung einrichten, der genaue Typ und die Argumente ändern sich für jeden Anforderungstyp
- queue.add (Anfrage); // füge die Anfrage zur Warteschlange hinzu; Der entsprechende Antwortlistener wird aufgerufen, sobald die Anforderung abgeschlossen ist (oder aus irgendeinem Grund beendet wird).

Bemerkungen

Installation

Sie können Volley aus dem offiziellen Google-Quellcode erstellen . Für eine Weile war das die einzige Option. Oder verwenden Sie eine der vorgefertigten Versionen eines Drittanbieters. Google hat jedoch endlich ein offizielles Maven-Paket bei Jcenter veröffentlicht.

`build.gradle` Sie dies in Ihrer `build.gradle` Datei auf Anwendungsebene Ihrer Abhängigkeitsliste hinzu:

```
dependencies {
    ...
    compile 'com.android.volley:volley:1.0.0'
}
```

Stellen Sie sicher, dass die `INTERNET` Berechtigung im Manifest Ihrer App festgelegt ist:

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

Offizielle Dokumentation

Google hat diese Bibliothek nicht sehr umfangreich dokumentiert und sie haben sie seit Jahren nicht mehr angerührt. Was verfügbar ist, finden Sie unter:

https://developer.android.com/training/volley/index.html

Es gibt inoffizielle Dokumentation, die auf GitHub gehostet wird, obwohl es einen besseren Ort geben sollte, um dies in Zukunft zu hosten:

https://pablobaxter.github.io/volley-docs/

Examples

Basic StringRequest mit der GET-Methode

```
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);
```

Anfrage abbrechen

```
// 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);
```

Hinzufügen von benutzerdefinierten Entwurfszeitattributen zu NetworkImageView

Es gibt einige zusätzliche Attribute, die Volley `NetworkImageView` zum Standard- `ImageView` . Diese Attribute können jedoch nur im Code festgelegt werden. Das folgende Beispiel zeigt, wie Sie eine Erweiterungsklasse erstellen, die die Attribute aus Ihrer XML-Layoutdatei aufnimmt und auf die `NetworkImageView` Instanz für Sie anwendet.

`attrx.xml` Verzeichnis `~/res/xml` eine Datei mit dem Namen `attrx.xml` :

```
<resources>
    <declare-styleable name="MoreNetworkImageView">
        <attr name="defaultImageResId" format="reference"/>
        <attr name="errorImageResId" format="reference"/>
    </declare-styleable>
</resources>
```

Fügen Sie Ihrem Projekt eine neue Klassendatei hinzu:

```
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);
        }
    }
}
```

Eine Beispiel-Layoutdatei, die die Verwendung der benutzerdefinierten Attribute zeigt:

```
<?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 anfordern

```
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);
```

Hinzufügen von benutzerdefinierten Kopfzeilen zu Ihren Anforderungen [z. B. für grundlegende Auth]

Wenn Sie Ihren Volley-Requests benutzerdefinierte Header hinzufügen müssen, ist dies nach der Initialisierung nicht möglich, da die Header in einer privaten Variablen gespeichert werden.

Stattdessen müssen Sie die `getHeaders()` Methode von `Request.class` wie `getHeaders()` überschreiben:

```
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;
    }
};
```

Erklärung der Parameter:

- `REQUEST_METHOD` - Eine der `Request.Method.*` Konstanten.
- `REQUEST_URL` - Die vollständige URL, an die Ihre Anfrage `REQUEST_URL` .
- `REQUEST_BODY` - Ein `JSONObject` das den zu `JSONObject` POST-Body (oder `JSONObject` enthält.
- `RESP_LISTENER` - Ein `Response.Listener<?>` Objekt, dessen `onResponse(T data)` -Methode nach erfolgreichem Abschluss aufgerufen wird.
- `ERR_LISTENER` - Ein `Response.ErrorListener` Objekt, dessen Methode `onErrorResponse(VolleyError e)` bei einer nicht erfolgreichen Anforderung aufgerufen wird.

Wenn Sie eine benutzerdefinierte Anfrage erstellen möchten, können Sie auch die Header hinzufügen:

```
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;
    }
    ...
}
```

Hilfsklasse für die Behandlung von Volley-Fehlern

```
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);
        }
```

Remote-Serverauthentifizierung mit StringRequest über die POST-Methode

Nehmen wir für dieses Beispiel an, dass wir einen Server haben, um die POST-Anfragen zu verarbeiten, die wir von unserer Android-App aus machen werden:

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

Verwendung von Volley für HTTP-Anforderungen

Fügen Sie die Gradle-Abhängigkeit in build.gradle auf App-Ebene hinzu

```
compile 'com.android.volley:volley:1.0.0'
```

Fügen Sie dem Manifest Ihrer App auch die Berechtigung android.permission.INTERNET hinzu .

** Volley RequestQueue-Instanz-Singleton in Ihrer Anwendung erstellen **

```
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;
}
}
```

Jetzt können Sie die Volleyinstanz mit der Methode getInstance () verwenden und mit `InitApplication.getInstance().addToQueue(request);` eine neue Anforderung in die Warteschlange `InitApplication.getInstance().addToQueue(request);`

Ein einfaches Beispiel zum Anfordern von JsonObject vom Server ist

```
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));
```

Um mit Volley-Timeouts umgehen zu können, müssen Sie eine `RetryPolicy` . Eine Wiederholungsrichtlinie wird verwendet, wenn eine Anforderung aufgrund eines Netzwerkfehlers oder in einigen anderen Fällen nicht abgeschlossen werden kann.

Volley bietet eine einfache Möglichkeit, Ihre `RetryPolicy` für Ihre Anforderungen zu implementieren. Standardmäßig setzt Volley für alle Anforderungen alle Socket- und Verbindungszeitlimits auf 5 Sekunden. `RetryPolicy` ist eine Schnittstelle, in der Sie Ihre Logik implementieren müssen, wie Sie eine bestimmte Anforderung wiederholen möchten, wenn ein Timeout auftritt.

Der Konstruktor verwendet die folgenden drei Parameter:

- `initialTimeoutMs` - Gibt das Socket-Timeout in Millisekunden für jeden Wiederholungsversuch an.
- `maxNumRetries` - Die Anzahl der Wiederholversuche.
- `backoffMultiplier` - Ein Multiplikator, mit dem die exponentielle Zeit für jeden Wiederholungsversuch auf Socket festgelegt wird.

Boolesche variable Antwort vom Server mit Json-Anfrage in Volley

Sie können benutzerdefinierte Klasse unter einem

```
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;
        }
    }
}
```

Verwenden Sie dies bei Ihrer Aktivität

```
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();
    }
```

Verwenden Sie JSONArray als Anforderungstext

Die in Volley integrierten Standardanforderungen erlauben es nicht, ein `JSONArray` als Anforderungstext in einer `POST` Anforderung zu übergeben. Stattdessen können Sie ein `JSON` Objekt nur als Parameter übergeben.

Anstatt jedoch ein `JSON` Objekt als Parameter an den Anforderungskonstruktor zu übergeben, müssen Sie die `getBody()` Methode der `Request.class` überschreiben. Sie sollten auch `null` als dritten Parameter übergeben:

```
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;
        }
    }
};
```

Erklärung der Parameter:

- `REQUEST_URL` - Die vollständige URL, an die Ihre Anfrage `REQUEST_URL` .
- `RESP_LISTENER` - Ein `Response.Listener<?>` Objekt, dessen `onResponse(T data)` -Methode nach erfolgreichem Abschluss aufgerufen wird.
- `ERR_LISTENER` - Ein `Response.ErrorListener` Objekt, dessen Methode `onErrorResponse(VolleyError e)` bei einer nicht erfolgreichen Anforderung aufgerufen wird.

Volley online lesen: https://riptutorial.com/de/android/topic/2800/volley

Kapitel 255: Was ist ProGuard? Was ist die Verwendung in Android?

Einführung

Proguard ist ein freier Java-Klassendatei-Verkleinerer, Optimierer, Verschleierer und Vorversteller. Es erkennt und entfernt nicht verwendete Klassen, Felder, Methoden und Attribute. Es optimiert den Bytecode und entfernt nicht verwendete Anweisungen. Die verbleibenden Klassen, Felder und Methoden werden mit kurzen, sinnlosen Namen umbenannt.

Examples

Verkleinern Sie Ihren Code und Ihre Ressourcen mit proguard

Um Ihre APK-Datei so klein wie möglich zu machen, sollten Sie das Verkleinern aktivieren, um nicht verwendeten Code und Ressourcen in Ihrem Release-Build zu entfernen. Diese Seite beschreibt, wie Sie dies tun und wie Sie festlegen, welcher Code und welche Ressourcen während des Builds aufbewahrt oder verworfen werden sollen.

Code-Verkleinerung ist in ProGuard verfügbar, das nicht verwendete Klassen, Felder, Methoden und Attribute aus Ihrer gepackten App erkennt und entfernt, einschließlich derjenigen aus enthaltenen Codebibliotheken (damit ein wertvolles Werkzeug für die Verwendung des Referenzgrenzwerts von 64 KB). ProGuard optimiert außerdem den Bytecode, entfernt nicht verwendete Codeanweisungen und verschleiert die verbleibenden Klassen, Felder und Methoden mit Kurznamen. Aufgrund des verschleierten Codes ist es schwierig, Ihr APK Reverse Engineering durchzuführen. Dies ist besonders nützlich, wenn Ihre App sicherheitsrelevante Funktionen verwendet, z. B. die Lizenzprüfung.

Das Android-Plugin für Gradle bietet Ressourcenreduzierung, mit der ungenutzte Ressourcen, einschließlich ungenutzter Ressourcen in Codebibliotheken, aus der gepackten App entfernt werden. Es funktioniert in Verbindung mit dem Schrumpfen von Code, sodass nach dem Entfernen von nicht verwendetem Code alle Ressourcen, auf die nicht mehr verwiesen wird, auch sicher entfernt werden können.

Verkleinern Sie Ihren Code

`minifyEnabled` Sie `minifyEnabled` true zum entsprechenden Build-Typ in Ihrer `build.gradle` Datei hinzu, um die Code-Verkleinerung mit `ProGuard` `build.gradle` .

Beachten Sie, dass das Verkürzen von Code die Erstellungszeit verlangsamt. Sie sollten es daher möglichst nicht in Ihrem Debug-Build verwenden. Es ist jedoch wichtig, dass Sie den Code für die endgültige APK-Verkleinerung aktivieren, die zum Testen verwendet wird. Andernfalls kann es zu Fehlern kommen, wenn Sie den Code nicht ausreichend anpassen.

Mit dem folgenden Snippet aus einer `build.gradle` Datei können Sie beispielsweise den Code für

den Release-Build verkleinern:

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

Neben der `minifyEnabled` Eigenschaft, die `proguardFiles` definiert Eigenschaft der `ProGuard rules` :

Die Methode getDefaultProguardFile ('proguard-android.txt') ruft die ProGuard-Standardeinstellungen aus dem Android SDK- `tools/proguard/ folder` . Tipp: Um noch mehr Code zu verkleinern, probieren Sie die Datei `proguard-android-optimize.txt` , die sich am selben Ort befindet. Es enthält die gleichen ProGuard-Regeln, jedoch mit anderen Optimierungen, die eine Analyse auf Bytecode-Ebene (innerhalb und zwischen Methoden) durchführen, um die APK-Größe weiter zu reduzieren und zu beschleunigen. In der Datei `proguard-rules.pro` können Sie benutzerdefinierte ProGuard-Regeln hinzufügen. Diese Datei befindet sich standardmäßig im Stammverzeichnis des Moduls (neben der Datei build.gradle). Um weitere ProGuard-Regeln hinzuzufügen, die für jede Buildvariante spezifisch sind, fügen Sie dem entsprechenden `productFlavor` Block eine weitere proguardFiles `productFlavor` . Die folgende Gradle-Datei fügt beispielsweise Flavor2-rules.pro zum Flavour2-Produktgeschmack hinzu. Nun verwendet flavor2 alle drei ProGuard-Regeln, da auch die Regeln aus dem Release-Block angewendet werden.

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

Was ist ProGuard? Was ist die Verwendung in Android? online lesen:
https://riptutorial.com/de/android/topic/9205/was-ist-proguard--was-ist-die-verwendung-in-android-

Kapitel 256: WebView

Einführung

WebView ist eine Ansicht, die Webseiten in Ihrer Anwendung anzeigt. Hiermit können Sie Ihre eigene URL hinzufügen.

Bemerkungen

Vergessen Sie nicht, die Erlaubnis in Ihre Android-Manifestdatei aufzunehmen

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

Examples

JavaScript-Warnungsdialogfelder in WebView - So funktionieren sie

Standardmäßig implementiert WebView keine JavaScript-Warnungsdialogfelder, z. `alert()` tut nichts. Um dies zu ermöglichen, müssen Sie zunächst JavaScript aktivieren (offensichtlich ..) und dann einen `WebChromeClient` , um Anforderungen für `WebChromeClient` auf der Seite zu bearbeiten:

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

Hier überschreiben wir `onJsAlert` und rufen dann die Super-Implementierung auf, die uns einen Standard-Android-Dialog gibt. Sie können die Nachricht und die URL auch selbst verwenden, z. B. wenn Sie ein benutzerdefiniertes Dialogfenster erstellen möchten oder wenn Sie sie protokollieren möchten.

Kommunikation von Javascript nach Java (Android)

Android-Aktivität

```
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
    }
}
```

Webseite, Javascript aufrufen

```
<script>
...
Android.jsCallback();
...
Android.jsCallback('hello test');
...
</script>
```

Extra Tipp

Eine mögliche Lösung, die eine komplexe Datenstruktur übergibt, ist JSON.

```
Android.jsCallback('{ "fake-var" : "fake-value", "fake-array" : [0,1,2] }');
```

Verwenden Sie auf der Android-Seite Ihren bevorzugten JSON-Parser, z

Kommunikation von Java nach Javascript

Basisbeispiel

```
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!')");
   }
}
```

Dialer-Beispiel öffnen

Wenn die Webseite a eine Telefonnummer enthält, können Sie mit dem Telefon Ihres Telefons einen Anruf tätigen. Dieser Code prüft die URL, die mit tel: beginnt: Warten Sie dann auf Dialer, und Sie können die geklickte Telefonnummer anrufen:

```
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;
}
```

Fehlerbehebung für WebView durch Drucken von Konsolennachrichten oder durch Remote-Debugging

Meldungen der Webview-Konsole in logcat drucken

Um `console` von Webseiten zu verarbeiten, können Sie `onConsoleMessage` in `WebChromeClient` :

```
final class ChromeClient extends WebChromeClient {
    @Override
    public boolean onConsoleMessage(ConsoleMessage msg) {
        Log.d(
            "WebView",
            String.format("%s %s:%d", msg.message(), msg.lineNumber(), msg.sourceId())
        );
        return true;
    }
}
```

Und setze es in deine Aktivität oder dein Fragment:

```
webView.setWebChromeClient(new ChromeClient());
```

Also diese Beispielseite:

```
<html>
<head>
    <script type="text/javascript">
        console.log('test message');
    </script>
</head>
<body>
</body>
</html>
```

schreibt Protokoll 'Testnachricht' in Logcat:

> WebView: Testnachricht sample.html: 4

`console.info()` , `console.warn()` und `console.error()` werden ebenfalls von Chrome-Client unterstützt.

Remote-Debugging von Android-Geräten mit Chrome

Ihre Webview-basierte Remote-Anwendung kann von Ihrem Desktop-Chrome aus ferngesteuert werden.

Aktivieren Sie das USB-Debugging auf Ihrem Android-Gerät

Öffnen Sie auf Ihrem Android-Gerät Einstellungen, finden Sie den Abschnitt Entwickleroptionen und aktivieren Sie das USB-Debugging.

Verbinden Sie sich und entdecken Sie Ihr Android-Gerät

Öffnen Sie die folgende Seite in Chrome: Chrome: // inspect / # devices

Wählen Sie im Dialogfeld "Geräte **prüfen"** Ihr Gerät aus und drücken Sie auf "Prüfen". Auf Ihrem Entwicklungscomputer wird eine neue Instanz von Chrome DevTools geöffnet.

Detailliertere Richtlinie und Beschreibung von DevTools finden Sie unter developers.google.com

Lokale Datei öffnen / Dynamischen Inhalt in Webview erstellen

Layout.xml

```
            <WebView
            android:id="@+id/WebViewToDisplay"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:fadeScrollbars="false" />
```

Laden Sie Daten in WebViewToDisplay

```
    WebView webViewDisplay;
    StringBuffer LoadWEb1;

    webViewDisplay = (WebView) findViewById(R.id.WebViewToDisplay);
    LoadWEb1 = new StringBuffer();
    LoadWEb1.append("<html><body><h1>My First Heading</h1><p>My first paragraph.</p>");
    //Sample code to read parameters at run time
    String strName = "Test Paragraph";
    LoadWEb1.append("<br/><p>"+strName+"</p>");
    String result = LoadWEb1.append("</body></html>").toString();
            WebSettings webSettings = webViewDisplay.getSettings();
            webSettings.setJavaScriptEnabled(true);
            webViewDisplay.getSettings().setBuiltInZoomControls(true);
            if (android.os.Build.VERSION.SDK_INT >= 11){
                webViewDisplay.setLayerType(View.LAYER_TYPE_SOFTWARE, null);
                webViewDisplay.getSettings().setDisplayZoomControls(false);
            }

            webViewDisplay.loadDataWithBaseURL(null, result, "text/html", "utf-8",
                    null);
            //To load local file directly from assets folder use below code
            //webViewDisplay.loadUrl("file:///android_asset/aboutapp.html");
```

WebView online lesen: https://riptutorial.com/de/android/topic/153/webview

Kapitel 257: Werkzeugeigenschaften

Bemerkungen

Android verfügt über einen speziellen XML-Namespace, der dazu bestimmt ist, dass Tools Informationen in XML-Dateien aufzeichnen können.

Der Namespace-URI lautet:

`http://schemas.android.com/tools` und ist normalerweise an die `tools:` Präfix gebunden.

Examples

Design-Layout-Attribute

Diese Attribute werden verwendet, wenn das Layout in Android Studio gerendert wird, haben jedoch keine Auswirkungen auf die Laufzeit.

Im Allgemeinen können Sie jedes Android-Framework-Attribut verwenden, indem Sie einfach die folgenden `tools:` Namespace statt `android:` Namespace für die Layoutvorschau. Sie können sowohl das `android:` namespace-Attribut (das zur Laufzeit verwendet wird) als auch das entsprechende `tools:` -Attribut (das das Laufzeitattribut nur in der Layoutvorschau überschreibt) hinzufügen.

Definieren Sie einfach den Namespace der Tools wie im Abschnitt "Anmerkungen" beschrieben.

Zum Beispiel des `text`

```
<EditText
      tools:text="My Text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
```

Oder das `visibility` , um eine Ansicht für die Vorschau zu deaktivieren:

```
<LinearLayout
        android:id="@+id/ll1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:visibility="gone" />
```

Oder das `context` , um das Layout einer Aktivität oder einem Fragment zuzuordnen

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity" >
```

Oder das `showIn` Attribut, um die Layoutvorschau in einem anderen Layout anzuzeigen

```
<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" />
```

Werkzeugeigenschaften online lesen:
https://riptutorial.com/de/android/topic/1676/werkzeugeigenschaften

Kapitel 258: Widgets

Bemerkungen

SDv

Examples

Manifesterklärung -

Deklarieren Sie die `AppWidgetProvider` Klasse in Ihrer Anwendung `AndroidManifest.xml` Datei. Zum Beispiel:

```
<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>
```

Metadaten

Fügen Sie die AppWidgetProviderInfo-Metadaten in `res/xml` :

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="40dp"
    android:minHeight="40dp"
    android:updatePeriodMillis="86400000"
    android:previewImage="@drawable/preview"
    android:initialLayout="@layout/example_appwidget"
    android:configure="com.example.android.ExampleAppWidgetConfigure"
    android:resizeMode="horizontal|vertical"
    android:widgetCategory="home_screen">
</appwidget-provider>
```

AppWidgetProvider-Klasse

Der wichtigste `AppWidgetProvider` Callback ist **`onUpdate()`** . Es wird jedes Mal aufgerufen, wenn ein Appwidget hinzugefügt wird.

```
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()` wird aufgerufen, wenn das Widget platziert oder seine Größe `onAppWidgetOptionsChanged()` wird.

`onDeleted(Context, int[])` wird aufgerufen, wenn das Widget gelöscht wird.

Zwei Widgets mit unterschiedlichen Layouts

1. Deklarieren Sie zwei Empfänger in einer Manifestdatei:

```
<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. Erstellen Sie zwei Layouts
    - `@xml/widget_1x1`
    - `@xml/widget_2x2`
3. Deklarieren Sie die Unterklasse `UVMateWidget2x2` aus der `UVMateWidget` Klasse mit erweitertem Verhalten:

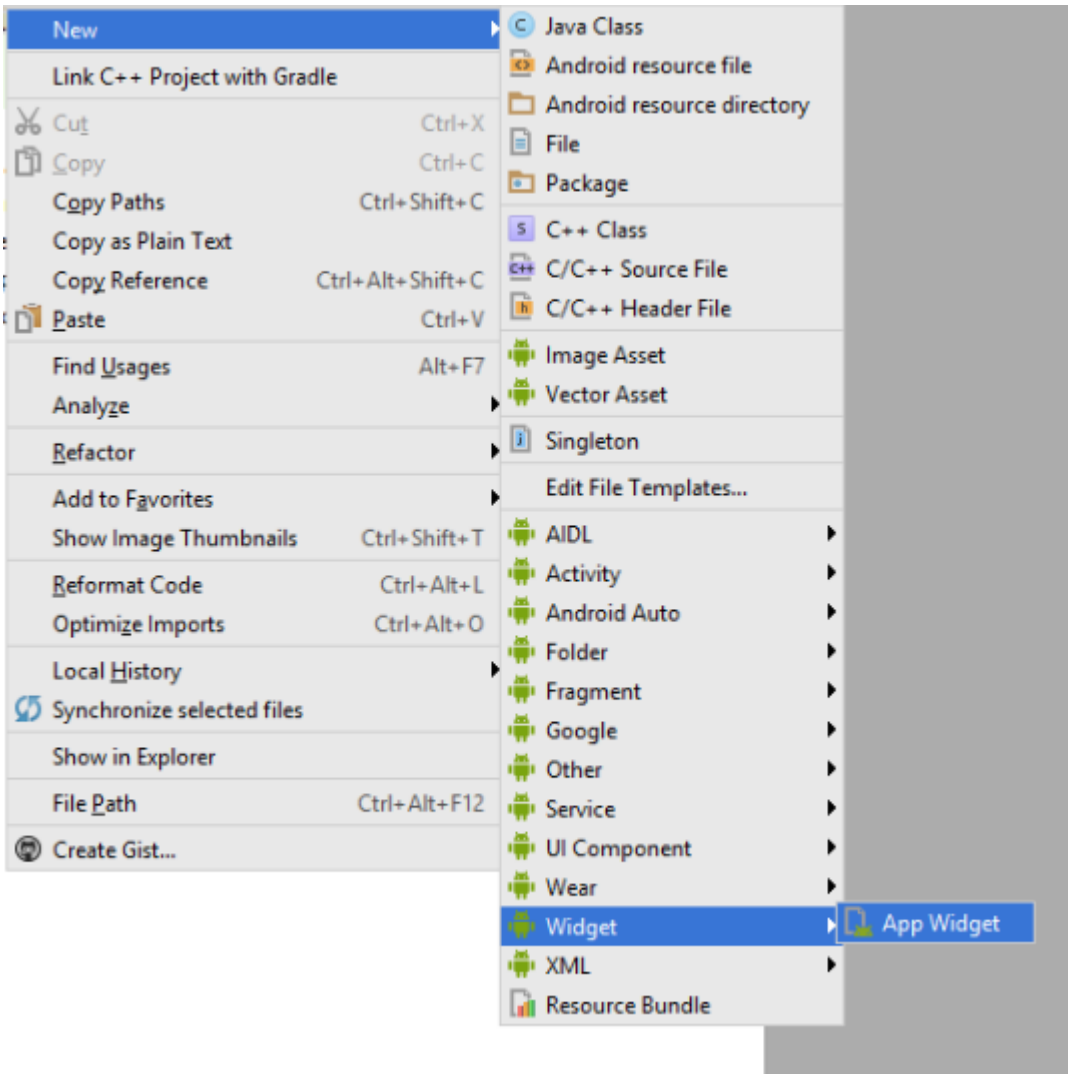
```
package au.com.aershov.uvmate;

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);
    }
}
```

Erstellen / Integrieren Sie Basic Widget mit Android Studio

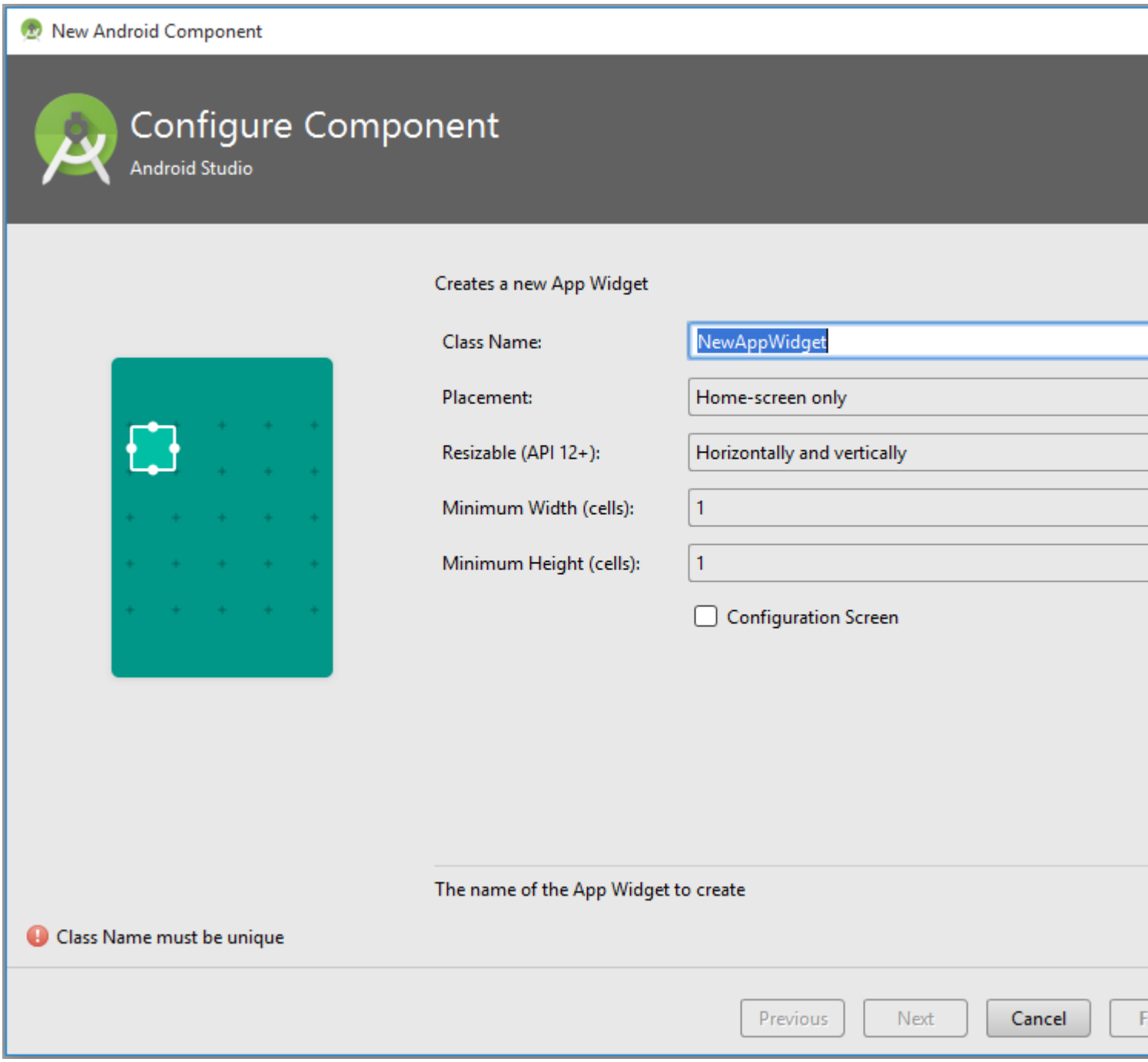
Das neueste Android Studio erstellt und integriert ein Basic Widget in Ihre Anwendung in 2 Schritten.

Rechts in Ihrer Anwendung ==> Neu ==> Widget ==> App Widget

.



Es wird ein Bildschirm wie unten angezeigt und die Felder werden ausgefüllt



Es ist fertig.

Es **erstellt und integriert ein einfaches HelloWorld-Widget** (einschließlich Layoutdatei, Metadatendatei, Deklaration in Manifestdatei usw.) in Ihre Anwendung.

Widgets online lesen: https://riptutorial.com/de/android/topic/2812/widgets

Kapitel 259: Wie verwende ich SparseArray?

Einführung

Ein `SparseArray` ist eine Alternative für eine `Map` . Eine `Map` erfordert, dass ihre Schlüssel Objekte sind. Das Phänomen des Autoboxings tritt auf, wenn wir einen primitiven `int` Wert als Schlüssel verwenden wollen. Der Compiler konvertiert primitive Werte automatisch in ihre geschachtelten Typen (z. B. `int` in `Integer`). Der Unterschied beim Speicherbedarf ist spürbar: `int` verwendet 4 Byte, `Integer` 16 Byte. Ein `SparseArray` verwendet `int` als Schlüsselwert.

Bemerkungen

Vorteil :

- Weniger Speicherbedarf (aufgrund der primitiven Schlüssel).
- Kein Auto-Boxen.

Nachteil:

- SparseArray verwendet die binäre Suche nach dem Suchwert (O (log n)). Daher ist es möglicherweise nicht die beste Lösung, wenn mit einer großen Anzahl von Elementen gearbeitet werden muss (verwenden Sie HashMap).

Es gibt verschiedene Varianten der Familie: -SparseArray <Integer, Object> -SparseBooleanArray <Integer, Boolean> -SparseIntArray <Integer, Integer> -SparseLongArray <Integer, Long> -LongSparseArray <Long, Object> -LongSparseLongArray <Long, Long >

SparseArray-Operationen

- Element hinzufügen - put (int, x): Fügt dem angegebenen Wert eine Zuordnung vom angegebenen Schlüssel hinzu, wobei die vorherige Zuordnung vom angegebenen Schlüssel ersetzt wird, falls vorhanden. - append (int, x): Fügt ein Schlüssel / Wert-Paar in das Array ein und optimiert für den Fall, dass der Schlüssel größer ist als alle vorhandenen Schlüssel im Array. Bei sequentiellen Schlüsseln sollten Sie append () verwenden, um die Leistung zu optimieren. Ansonsten ist put () in Ordnung.

- Element entfernen - delete (int): Entfernt die Zuordnung aus dem angegebenen Schlüssel, falls vorhanden. - removeAt (int): Entfernt die Zuordnung am angegebenen Index. - removeAtRange (int, int): Entfernt einen Zuordnungsbereich als Stapel.

- Zugriff auf Element - get (int): Ruft das vom angegebenen Schlüssel zugeordnete int oder 0 ab, wenn keine solche Zuordnung vorgenommen wurde. - get (int, E): Ruft das vom angegebenen Schlüssel zugeordnete int oder den angegebenen Wert ab, wenn keine solche Zuordnung vorgenommen wurde. - valueAt (int): Gibt einen Index im Bereich 0 ... size () - 1 zurück, gibt den Wert aus der indexten Schlüsselwertzuordnung zurück, die dieser SparseIntArray speichert. Indizes werden aufsteigend sortiert.

- index / key search - keyAt (int): Gibt einen Index im Bereich 0 ... size () - 1 zurück, gibt er den Schlüssel aus der indexten Schlüsselwertzuordnung zurück, die dieser SparseIntArray speichert. Indizes werden aufsteigend sortiert. - valueAt (int): Gibt einen Index im Bereich 0 ... size () - 1 zurück, gibt den Wert aus der indexten Schlüsselwertzuordnung zurück, die dieser SparseIntArray speichert. Indizes werden aufsteigend sortiert. - indexOfKey (int): Gibt den Index zurück, für den keyAt (int) den angegebenen Schlüssel zurückgeben würde, oder eine negative Zahl, wenn der angegebene Schlüssel nicht zugeordnet ist. - indexOfValue (E): Gibt einen Index zurück, für den valueAt (int) den angegebenen Schlüssel zurückgeben würde, oder eine negative Zahl, wenn dem angegebenen Wert keine Schlüssel zugeordnet sind. Beachten Sie, dass dies eine lineare Suche ist, im Gegensatz zu Suchvorgängen nach Schlüsseln, und dass mehrere Schlüssel auf denselben Wert abgebildet werden können. Dadurch wird nur einer von ihnen gefunden. Der Unterschied im Speicherbedarf ist auffällig: Das int verwendet 4 Bytes, die Integer verwendet 16 Bytes. ParseArray verwendet int als Schlüsselwert.

Examples

Ein einfaches Beispiel mit SparseArray

```
class Person {
    String name;

    public Person(String name) {
        this.name = name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Person person = (Person) o;

        return name != null ? name.equals(person.name) : person.name == null;
    }

    @Override
    public int hashCode() {
        return name != null ? name.hashCode() : 0;
    }

    @Override
    public String toString() {
        return "Person{" +
                "name='" + name + '\'' +
                '}';
    }
}

final Person steve = new Person("Steve");
Person[] persons = new Person[] { new Person("John"), new Person("Gwen"), steve, new
Person("Rob") };
int[] identifiers = new int[] {1234, 2345, 3456, 4567};
```

```
final SparseArray<Person> demo = new SparseArray<>();

// Mapping persons to identifiers.
for (int i = 0; i < persons.length; i++) {
    demo.put(identifiers[i], persons[i]);
}

// Find the person with identifier 1234.
Person id1234 = demo.get(1234); // Returns John.

// Find the person with identifier 6410.
Person id6410 = demo.get(6410); // Returns null.

// Find the 3rd person.
Person third = demo.valueAt(3); // Returns Rob.

// Find the 42th person.
//Person fortysecond = demo.valueAt(42); // Throws ArrayIndexOutOfBoundsException.

// Remove the last person.
demo.removeAt(demo.size() - 1); // Rob removed.

// Remove the person with identifier 1234.
demo.delete(1234); // John removed.

// Find the index of Steve.
int indexOfSteve = demo.indexOfValue(steve);

// Find the identifier of Steve.
int identifierOfSteve = demo.keyAt(indexOfSteve);
```

Tutorial auf YouTube

Wie verwende ich SparseArray? online lesen: https://riptutorial.com/de/android/topic/8824/wie-verwende-ich-sparsearray-

Kapitel 260: Wi-Fi-Verbindungen

Examples

Verbinden Sie sich mit der WEP-Verschlüsselung

In diesem Beispiel wird eine Verbindung zu einem WLAN-Zugangspunkt mit WEP-Verschlüsselung hergestellt, wobei eine SSID und das Kennwort angegeben werden.

```
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;
    }
}
```

Verbinden Sie sich mit der WPA2-Verschlüsselung

In diesem Beispiel wird eine Verbindung zu einem WLAN-Zugangspunkt mit WPA2-Verschlüsselung hergestellt.

```
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;
  }
}
```

Suchen Sie nach Zugangspunkten

In diesem Beispiel wird nach verfügbaren Zugangspunkten und Ad-hoc-Netzwerken gesucht. `btnScan` aktiviert einen Scan, der von der `WifiManager.startScan()` -Methode initiiert wird. Nach dem Scan `WifiManager` ruft die `SCAN_RESULTS_AVAILABLE_ACTION` Absicht und die `WifiScanReceiver` Klasse verarbeitet das Scan - Ergebnis. Die Ergebnisse werden in einer `TextView` angezeigt.

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

Berechtigungen

Die folgenden Berechtigungen müssen in *AndroidManifest.xml* definiert *werden* :

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

`android.permission.ACCESS_WIFI_STATE` ist zum Aufrufen von `WifiManager.getScanResults()` erforderlich. Ohne `android.permission.CHANGE_WIFI_STATE` kann kein Scan mit `WifiManager.startScan()` .

Beim Kompilieren des Projekts für API Level 23 oder höher (Android 6.0 und `android.permission.ACCESS_COARSE_LOCATION` muss entweder `android.permission.ACCESS_FINE_LOCATION`

oder `android.permission.ACCESS_COARSE_LOCATION` eingefügt werden. Außerdem muss diese Berechtigung angefordert werden, z. B. in der `onCreate` Methode Ihrer Hauptaktivität:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    String[] PERMS_INITIAL={
            Manifest.permission.ACCESS_FINE_LOCATION,
    };
    ActivityCompat.requestPermissions(this, PERMS_INITIAL, 127);
}
```

Wi-Fi-Verbindungen online lesen: https://riptutorial.com/de/android/topic/3288/wi-fi-verbindungen

Kapitel 261: XMPP-Register Anmelden und einfaches Beispiel

Examples

XMPP-Anmelde- und Chat-Beispiel

Installieren Sie Openfire oder einen beliebigen Chat-Server in Ihrem System oder auf dem Server. Für weitere Details klicken Sie hier.

Erstellen Sie ein Android-Projekt und fügen Sie diese Bibliotheken in Gradle hinzu:

```
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'
```

Erstellen Sie anschließend eine xmpp-Klasse aus dem Zweck der xmpp-Verbindung:

```
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();
    }
}

}
```

Zum Schluss fügen Sie diese Aktivität hinzu:

```
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();
    }

}
```

Stellen Sie sicher, dass die Internetberechtigung in Ihrer Manifestdatei hinzugefügt ist.

XMPP-Register Anmelden und einfaches Beispiel online lesen:
https://riptutorial.com/de/android/topic/6747/xmpp-register-anmelden-und-einfaches-beispiel

Kapitel 262: Xposed

Examples

Erstellen eines Xposed-Moduls

Xposed ist ein Framework, mit dem Sie Methodenaufrufe anderer Apps verknüpfen können. Wenn Sie eine Änderung vornehmen, indem Sie eine APK dekompilieren, können Sie Befehle direkt an beliebigen Stellen einfügen / ändern. Sie müssen die APK danach jedoch erneut kompilieren / signieren, und Sie können nur das gesamte Paket verteilen. Mit Xposed können Sie Ihren eigenen Code vor oder nach Methoden einfügen oder ganze Methoden vollständig ersetzen. Leider können Sie Xposed nur auf gerooteten Geräten installieren. Sie sollten Xposed immer dann verwenden, wenn Sie das Verhalten anderer Apps oder des Android-Kernsystems ändern möchten und nicht die Mühe machen müssen, APKs zu dekompilieren, neu zu kompilieren und zu signieren.

Zunächst erstellen Sie eine Standard-App ohne Aktivität in Android Studio.

Dann müssen Sie den folgenden Code in Ihr *build.gradle einfügen* :

```
repositories {
    jcenter();
}
```

Danach fügen Sie die folgenden Abhängigkeiten hinzu:

```
provided 'de.robv.android.xposed:api:82'
provided 'de.robv.android.xposed:api:82:sources'
```

Jetzt müssen Sie diese Tags innerhalb des *Anwendungstags* in *AndroidManifest.xml platzieren,* damit Xposed Ihr Modul erkennt:

```
<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" />
```

HINWEIS: Ersetzen Sie immer **82** durch die neueste Xposed-Version .

Eine Methode haken

Erstellen Sie eine neue Klasse, die `IXposedHookLoadPackage` implementiert, und implementieren Sie die `handleLoadPackage` Methode:

```
public class MultiPatcher implements IXposedHookLoadPackage
{
    @Override
    public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
    {

    }
}
```

Innerhalb der Methode überprüfen Sie `loadPackageParam.packageName` auf den Paketnamen der App, die Sie `loadPackageParam.packageName` möchten:

```
@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }
}
```

Jetzt können Sie Ihre Methode anhängen und entweder bearbeiten, bevor der Code ausgeführt wird oder nach:

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

Xposed online lesen: https://riptutorial.com/de/android/topic/4627/xposed

Kapitel 263: Youtube-API

Bemerkungen

1. Zunächst müssen Sie den neuesten Jar-Link unter folgendem Link herunterladen: https://developers.google.com/youtube/android/player/downloads/
2. Sie müssen dieses Glas in Ihr Projekt aufnehmen. Kopieren Sie dieses Jar in den libs-Ordner und vergessen Sie nicht, es in Abstufungsdateien für Abhängigkeiten hinzuzufügen {compile files ('libs / YouTubeAndroidPlayerApi.jar')}
3. Sie benötigen einen API-Schlüssel, um auf Youtube-APIs zuzugreifen. Folgen Sie diesem Link: https://developers.google.com/youtube/android/player/register , um Ihren API-Schlüssel zu generieren.
4. Reinigen und bauen Sie Ihr Projekt. Jetzt können Sie den YoutubeAndroidPlayerApi verwenden. Um ein Youtube-Video abzuspielen, benötigen Sie eine entsprechende Video-ID, damit Sie es auf Youtube abspielen können. Zum Beispiel: https://www.youtube.com/watch?v=B08iLAtS3AQ , B08iLAtS3AQ ist die Video-ID, die Sie zum Abspielen auf Youtube benötigen.

Examples

StandAlonePlayerActivity starten

1. Starten Sie eine eigenständige Spieleraktivität

```
     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);
```

Aktivität, die YouTubeBaseActivity erweitert

```
public class CustomYouTubeActivity extends YouTubeBaseActivity implements
YouTubePlayer.OnInitializedListener, YouTubePlayer.PlayerStateChangeListener {

    private YouTubePlayerView mPlayerView;
    private YouTubePlayer mYouTubePlayer;
    private String mVideoId = "B08iLAtS3AQ";
    private String mApiKey;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mApiKey = Config.YOUTUBE_API_KEY;
        mPlayerView = new YouTubePlayerView(this);
```

```
        mPlayerView.initialize(mApiKey, this); // setting up OnInitializedListener
        addContentView(mPlayerView, new LayoutParams(LayoutParams.MATCH_PARENT,
                LayoutParams.MATCH_PARENT)); //show it in full screen
    }

    //Called when initialization of the player succeeds.
    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider,
                                        YouTubePlayer player,
                                        boolean wasRestored) {

        player.setPlayerStateChangeListener(this); // setting up the player state change
listener
        this.mYouTubePlayer = player;
        if (!wasRestored)
            player.cueVideo(mVideoId);
    }

        @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
                                        YouTubeInitializationResult errorReason) {

        Toast.makeText(this, "Error While initializing", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onAdStarted() {
    }

    @Override
    public void onLoaded(String videoId) { //video has been loaded
        if(!TextUtils.isEmpty(mVideoId) && !this.isFinishing() && mYouTubePlayer != null)
            mYouTubePlayer.play(); // if we dont call play then video will not auto play, but
user still has the option to play via play button
    }

    @Override
    public void onLoading() {
    }

    @Override
    public void onVideoEnded() {
    }

    @Override
    public void onVideoStarted() {

    }

    @Override
    public void onError(ErrorReason reason) {
        Log.e("onError", "onError : " + reason.name());
    }

}
```

YoutubePlayerFragment im Portrait Activty

Der folgende Code implementiert ein einfaches YoutubePlayerFragment. Das Layout der Aktivität

ist im Hochformatmodus gesperrt. Wenn sich die Ausrichtung ändert oder der Benutzer im YoutubePlayer auf Vollbildschirm klickt, wird die Landschaft in eine Landschaft mit dem YoutubePlayer umgewandelt. Das YoutubePlayerFragment muss keine Aktivität erweitern, die von der Youtube-Bibliothek bereitgestellt wird. Es muss YouTubePlayer.OnInitializedListener implementiert werden, um den YoutubePlayer zu initialisieren. Die Klasse unserer Aktivität ist also die folgende

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

Ein YoutubePlayerFragment kann wie folgt zum Layout-Xaml der Aktivität hinzugefügt werden

```
<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>
```

Zuletzt müssen Sie die folgenden Attribute in Ihre Manifest-Datei im Tag der Aktivität einfügen

```
android:configChanges="keyboardHidden|orientation|screenSize"
android:screenOrientation="portrait"
```

YouTube Player-API

Beziehen des Android-API-Schlüssels:

Zuerst müssen Sie den SHA-1-Fingerabdruck mit Java-Keytool auf Ihrem Computer abrufen. Führen Sie den folgenden Befehl in cmd / terminal aus, um den SHA-1-Fingerabdruck zu erhalten.

```
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);
    }
}
```

Erstellen `Config.java` Datei `Config.java` . Diese Datei enthält den Google Console API-Entwicklerschlüssel und die YouTube-Video-ID

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-Datei

```
<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" />
```

YouTube-Daten-API auf Android nutzen

In diesem Beispiel erfahren Sie, wie Sie Wiedergabelistendaten mithilfe der YouTube-Daten-API von Android abrufen.

SHA-1 Fingerabdruck

Zuerst benötigen Sie einen SHA-1-Fingerabdruck für Ihre Maschine. Es gibt verschiedene Methoden, um es abzurufen. Sie können eine beliebige Methode auswählen, die in diesem Q & A angegeben ist .

Google API-Konsole und YouTube-Schlüssel für Android

Nachdem Sie einen SHA-1-Fingerabdruck erstellt haben, öffnen Sie die Google API-Konsole und erstellen Sie ein Projekt. Rufen Sie diese Seite auf und erstellen Sie ein Projekt mit diesem SHA-1-Schlüssel, und aktivieren Sie die YouTube-Daten-API. Jetzt bekommst du einen Schlüssel. Dieser Schlüssel wird zum Senden von Anforderungen von Android und zum Abrufen von Daten

verwendet.

Gradle Teil

Für die YouTube-Daten-API müssen Sie Ihrer Gradle-Datei die folgenden Zeilen hinzufügen:

```
compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
```

Um den nativen Client von YouTube zum Senden von Anforderungen zu verwenden, müssen wir in Gradle die folgenden Zeilen hinzufügen:

```
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:+'
```

Die folgende Konfiguration muss auch in Gradle hinzugefügt werden, um Konflikte zu vermeiden:

```
configurations.all {
    resolutionStrategy.force 'com.google.code.findbugs:jsr305:3.0.2'
}
```

Nachfolgend wird gezeigt, wie der *gradle.build* endlich aussehen würde.

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:+'
}
```

Nun kommt der Java-Teil. Da wir `HttpTransport` zum Vernetzen und `GsonFactory` zum Konvertieren von JSON in POJO verwenden werden, benötigen wir keine andere Bibliothek zum Senden von Anforderungen.

Jetzt möchte ich zeigen, wie Playlisten über die YouTube-API abgerufen werden, indem die Playlist-IDs bereitgestellt werden. Für diese Aufgabe werde ich `AsyncTask` . Um zu verstehen, wie wir Parameter anfordern, und um den Ablauf zu verstehen, werfen Sie einen Blick auf die YouTube-Daten-API .

```
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;
    }
}
```

Die obige asynchrone Task gibt eine Instanz von `PlaylistListResponse` die eine eingebaute Klasse des YouTube-SDK ist. Es hat alle erforderlichen Felder, so dass wir keine POJOs selbst erstellen müssen.

Schließlich `MainActivity` wir in unserer `MainActivity` Folgendes tun:

```
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);
    }
}
```

Youtube-API online lesen: https://riptutorial.com/de/android/topic/7587/youtube-api

Kapitel 264: Zeichenketten formatieren

Examples

Formatieren Sie eine Zeichenfolgeressource

Sie können Platzhalterzeichen in String-Ressourcen hinzufügen und zur Laufzeit auffüllen:

1. Bearbeiten Sie strings.xml

```
<string name="my_string">This is %1$s</string>
```

2. Formatieren Sie den String nach Bedarf

```
String fun = "fun";
context.getString(R.string.my_string, fun);
```

Formatieren Sie einen Zeitstempel in eine Zeichenfolge

Eine vollständige Beschreibung der Muster finden Sie in der SimpleDateFormat-Referenz

```
Date now = new Date();
long timestamp = now.getTime();
SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy", Locale.US);
String dateStr = sdf.format(timestamp);
```

Formatieren von Datentypen in String und umgekehrt

Datentypen für die String-Formatierung

Datentypen wie int, float, double, long und boolean können mit String.valueOf () als Zeichenfolge formatiert werden.

```
String.valueOf(1); //Output -> "1"
String.valueOf(1.0); //Output -> "1.0"
String.valueOf(1.2345); //Output -> "1.2345"
String.valueOf(true); //Output -> "true"
```

Umgekehrt, formatieren Sie den String mit einem anderen Datentyp

```
Integer.parseInt("1"); //Output -> 1
Float.parseFloat("1.2"); //Output -> 1.2
Boolean.parseBoolean("true"); //Output -> true
```

Zeichenketten formatieren online lesen:
https://riptutorial.com/de/android/topic/1346/zeichenketten-formatieren

Kapitel 265: Zeit Utils

Examples

Datumsformat in Millisekunden umrechnen

Um Ihr Datum im Format TT / MM / JJJJ in Millisekunden umzuwandeln, rufen Sie diese Funktion mit Daten als String auf

```
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();
}
```

Diese Methode konvertiert Millisekunden in Datum des Zeitstempelformats:

```
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;
}
```

Diese Methode konvertiert einen bestimmten Tag, Monat und Jahr in Millisekunden. Es ist sehr hilfreich, wenn Sie `Timpicker` oder `Datepicker`

```
public static long getTimeInMillis(int day, int month, int year) {
    Calendar calendar = Calendar.getInstance();
    calendar.set(year, month, day);
    return calendar.getTimeInMillis();
}
```

Es wird Millisekunden nach Datum zurückgegeben

```
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;
}
```

Das aktuelle Datum wird zurückgegeben

```
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;
}
```

Hinweis: Java bietet Datumsformat-Unterstützung für Datumsformate

Innerhalb eines Zeitraums überprüfen

Dieses Beispiel hilft zu überprüfen, ob die angegebene Zeit innerhalb eines bestimmten Zeitraums liegt oder nicht.

*Um zu überprüfen, ob heute die Zeit ist, können wir die **DateUtils-** Klasse verwenden*

```
boolean isToday = DateUtils.isToday(timeInMillis);
```

Um zu überprüfen, ist die Zeit innerhalb einer Woche,

```
private static boolean isWithinWeek(final long millis) {
    return System.currentTimeMillis() - millis <= (DateUtils.WEEK_IN_MILLIS -
DateUtils.DAY_IN_MILLIS);
}
```

Um die Zeit innerhalb eines Jahres zu überprüfen,

```
private static boolean isWithinYear(final long millis) {
    return System.currentTimeMillis() - millis <= DateUtils.YEAR_IN_MILLIS;
}
```

Um zu überprüfen, ist die Uhrzeit innerhalb einer Reihe von Tagen des Tages einschließlich heute,

```
public static boolean isWithinDay(long timeInMillis, int day) {
    long diff = System.currentTimeMillis() - timeInMillis;

    float dayCount = (float) (diff / DateUtils.DAY_IN_MILLIS);

    return dayCount < day;
}
```

Hinweis: DateUtils ist **android.text.format.DateUtils**

GetCurrentRealTime

Berechnet die aktuelle Gerätezeit und addiert / subtrahiert die Differenz zwischen der tatsächlichen und der Gerätezeit

```
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-basierte Zeitzone abrufen.

```
public static TimeZone getUTCTimeZone() {
    return TimeZone.getTimeZone("GMT");
}
```

Zeit Utils online lesen: https://riptutorial.com/de/android/topic/7138/zeit-utils

Kapitel 266: ZIP-Datei in Android

Examples

ZIP-Datei auf Android

```
import android.util.Log;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;


  public class Compress {
      private static final int BUFFER = 2048;

      private String[] _files;
      private String _zipFile;

      public Compress(String[] files, String zipFile) {
        _files = files;
        _zipFile = zipFile;
      }

      public void zip() {
        try  {
          BufferedInputStream origin = null;
          FileOutputStream dest = new FileOutputStream(_zipFile);

          ZipOutputStream out = new ZipOutputStream(new BufferedOutputStream(dest));

          byte data[] = new byte[BUFFER];

          for(int i=0; i < _files.length; i++) {
            Log.v("Compress", "Adding: " + _files[i]);
            FileInputStream fi = new FileInputStream(_files[i]);
            origin = new BufferedInputStream(fi, BUFFER);
            ZipEntry entry = new ZipEntry(_files[i].substring(_files[i].lastIndexOf("/") +
1));
            out.putNextEntry(entry);
            int count;
            while ((count = origin.read(data, 0, BUFFER)) != -1) {
              out.write(data, 0, count);
            }
            origin.close();
          }

          out.close();
        } catch(Exception e) {
          e.printStackTrace();
        }

      }

    }
```

ZIP-Datei in Android online lesen: https://riptutorial.com/de/android/topic/8137/zip-datei-in-android

Kapitel 267: Zugriff auf SQLite-Datenbanken mithilfe der ContentValues-Klasse

Examples

Zeilen in eine SQLite-Datenbank einfügen und aktualisieren

Zunächst müssen Sie Ihre SQLite-Datenbank öffnen. Dies kann wie folgt durchgeführt werden:

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

Nach dem Öffnen der Datenbank können Sie Zeilen mithilfe der `ContentValues` Klasse problemlos einfügen oder aktualisieren. In den folgenden Beispielen wird davon ausgegangen, dass der Vorname `str_edtfname` und der letzte Name mit `str_edtlname` . Sie müssen `table_name` durch den Namen Ihrer Tabelle ersetzen, die Sie ändern möchten.

Daten einfügen

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

Daten aktualisieren

```
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});
```

Zugriff auf SQLite-Datenbanken mithilfe der ContentValues-Klasse online lesen: https://riptutorial.com/de/android/topic/10154/zugriff-auf-sqlite-datenbanken-mithilfe-der-contentvalues-klasse

Kapitel 268: Zum Aktualisieren wischen

Syntax

1. **setColorSchemeResources** legt die Farben des SwipeToRefreshLayout- **Indikators fest**
2. **setOnRefreshListener** legt fest, was zu tun ist, wenn das Layout gestrichen wird
3. **app: layout_behavior = "@ string / appbar_scrolling_view_behavior"** Wenn Sie über eine Toolbar mit Ihrem Layout verfügen, fügen Sie dies mit den scrollFlags in der Toolbar hinzu. Die Toolbar wird beim Scrollen nach oben und beim Scrollen nach oben verschoben.

Examples

Mit RecyclerView aktualisieren

So fügen Sie ein Layout **zum Aktualisieren** mit einer **RecyclerView** hinzu:

```
<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>
```

Fügen Sie in Ihrem Activity / Fragment Folgendes hinzu, um das **SwipeToRefreshLayout** zu initialisieren:

```
    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
        }
    });
```

So fügen Sie Ihrer App Swipe-to-Refresh hinzu

`build.gradle` Sie sicher, dass die folgende Abhängigkeit zur `build.gradle` Datei Ihrer App unter Abhängigkeiten hinzugefügt wird:

```
compile 'com.android.support:support-core-ui:24.2.0'
```

`SwipeRefreshLayout` Sie dann das `SwipeRefreshLayout` in Ihr Layout ein:

```
<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>
```

Implementieren Sie schließlich den Listener " `SwipeRefreshLayout.OnRefreshListener` .

```
mSwipeRefreshLayout = (SwipeRefreshLayout) findViewById(R.id.swipe_refresh_layout);
mSwipeRefreshLayout.setOnRefreshListener(new OnRefreshListener() {
    @Override
    public void onRefresh() {
         // your code
    }
});
```

Zum Aktualisieren wischen online lesen: https://riptutorial.com/de/android/topic/5241/zum-aktualisieren-wischen

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Android	6londe, Abhishek Jain, Adam Johns, AesSedai101, Ahmad Aghazadeh, Akash Patel, Ala Eddine JEBALI, Aleksandar Stefanović, Andrea, Andrew Brooke, AndroidMechanic, ankit dassor, Apoorv Parmar, auval, Blachshma, Blundering Philosopher, cascal, cdeange, Charlie H, Charuක, ChemicalFlash, Cold Fire, Community , Dalija Prasnikar, Daniel Nugent, Daniele Segato, Doron Behar, Dr. Nitpick, Duan Bressan, EKN, Erik Minarini, Gabriele Mariotti, Gaket, gattsbr, geekygenius, hankide, Harish Gyanani, HCarrasko, Ibrahim, Ichthyocentaurs, inetphantom, Intrications, Irfan, Jeeter, JSON C11, Kevin, Kinjal, Kiran Benny Joseph, Laurel, Mark Yisri, Matas Vaitkevicius, MathaN, Menasheh, Michael Allan, mnoronha, mohit, MrEngineer13, Nick, Nick, opt05, Patel Pinkal, Pavneet_Singh, Pro Mode, PSN, RamenChef, Ravi Rupareliya, rekire, ridsatrio, russt, saul, Seelass, Shiven, Siddharth Venu, Simplans, Sneh Pandya, Sree, sudo, sun-solar-arrow, Tanis.7x, Thomas Gerot, ThomasThiebaud, Tot Zam, Vivek Mishra, Yury Fedorov, Zarul Izham, Ziad Akiki, Zoe, רתוא בהוא ושי
2	9-Patch-Bilder	Knossos, Nissim R, Tomik
3	Absicht	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, noɥʎdʎzɐɹƆ, Oren, Paresh Mayani, Parsania Hardik, Paul Lammertsma, Pavneet_Singh, penkzhou, Peter Mortensen, Phan Van Linh, Piyush, R. Zagórski, Radouane ROUFID, Rajesh, RamenChef, rap-2-h, rciovati, Reaz Murshed, RediOne1, rekire, reVerse, russjr08, Ryan Hilbert, sabadow, Saveen, Simon, Simplans, SoroushA, spaceplane, Stelian Matei, Stephane Mathis, Stephen Leppik, sukumar, tainy, theFunkyEngineer, ThomasThiebaud, ɬolɐəz əɥɬ qoq, Tyler Sebastian, vasili111, Vasily Kabunov, Vinay , Vivek Mishra, Xaver Kapeller, younes zeboudj, Yury Fedorov, Zoe
4	ACRA	Zarul Izham, Zoe
5	ADB (Android Debug Bridge)	3VYZkz7t, adao7000, Ahmad Aghazadeh, Amit Thakkar, AndroidMechanic, Anirudh Sharma, Anup Kulkarni, auval, Barend, Blackbelt, Burak Day, Charuꜩ, Chris Stratton, Da-Jin C, Dale, Daniel Nugent, David Cheung, Erik, Fabio, fyfyone Google, g4s8, Gabriele Mariotti, grebulon, Hannoun Yassir, Hi I'm Frogatto, hichris123, honk, jim, Kashyap Jha, Laurel, MCeley, Menasheh, Natali, Nemus, Pavel Durov, Piyush, R. Zagórski, RishbhSharma, stkent, Sudip Bhandari, sukumar, theFunkyEngineer, thiagolr, Tien, Xaver Kapeller, Yassie, younes zeboudj, Yury Fedorov
6	ADB Shell	3VYZkz7t, Ahmad Aghazadeh, auval, Burak Day, Fabio, fyfyone Google, Hannoun Yassir, Natali, Pavel Durov, R. Zagórski, sukumar, Yury Fedorov
7	AdMob	Carlos Borau, honk, RamenChef, Sukrit Kumar, Zarul Izham, Zoe
8	AIDL	Krishnakanth
9	Aktivität	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
10	Aktivitätserkennung	Pablo Baxter
11	AlarmManager	Daniel Nugent, devnull69, Greg T, honk, TR4Android
12	Alert-Dialoge verbessern	Adil Saiyad, honk

13	Android Authenticator	4444, kRiZ
14	Android Java Native Interface (JNI)	Doron Yakovlev-Golani, Muthukrishnan Rajendran, samgak
15	Android NDK	Alex, astuter, Doron Yakovlev-Golani, Flayn, Onik, samgak, still_learning, Täg, thiagolr
16	Android Paypal Gateway-Integration	A-Droid Tech
17	Android Places-API	busradeniz, honk, Karan Razdan, Murali
18	Android Sound und Medien	johnrao07, Muhammad Umair Shafique, Squidward
19	Android Studio	AndroidMechanic, auval, Blackbelt, Charuක, Daniel Nugent, Gabriele Mariotti, Hiren Patel, Inzimam Tariq IT, Jon Adams, N J, Phan Van Linh, R. Zagórski, Squidward, Sujith Niraikulathan, ThomasThiebaud
20	Android Vk Sdk	alexey polusov
21	Android-Architekturkomponenten	DeKaNszn
22	Android-Dinge	Fabio, honk
23	Android-Kernel-Optimierung	honk, Sneh Pandya
24	Android-Programmierung mit Kotlin	Gian Patrick Quintana, Govinda Paliwal, Oknesif, Zarul Izham
25	Android-Spieleentwicklung	Zoe
26	Android-Versionen	4444, AndroidMechanic, athor, BooleanCheese, Dalija Prasnikar, Daniel Nugent, Fildor, Gabriele Mariotti, H. Pauwelyn, Matt, RediOne1, tynn
27	Android-x86 in VirtualBox	Daniel Nugent, Enrique de Miguel
28	Animatoren	Aryan, Bartek Lipinski, Blundering Philosopher, Brenden Kromhout, Charuක, Daniel Nugent, Eixx, Hiren Patel, Lewis McGeary, Piyush, TR4Android, Uriel Carrillo, Yury Fedorov
29	Animiertes AlertDialogfeld	krunal patel, Thomas Easo

30	Anmerkungsprozessor	krishan
31	Anzeigen von Google-Anzeigen	Egek92, RamenChef, ReverseCold, Stephen Leppik, Zarul Izham
32	Apps mit ADB installieren	Ahmad Aghazadeh, fyfyone Google, Laurel, Xaver Kapeller
33	Arbeit planen	RamenChef, reflective_mind
34	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
35	AudioManager	honk, Nicolai Weitkemper
36	AudioTrack	Ayush Bansal
37	Ausnahmen	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
38	AutoCompleteTextView	Harish Gyanani, Jon Adams, Ricardo Vieira, Vivek Mishra
39	Barcode- und QR-Code lesen	FlyingPumba
40	Bedienung	adao7000, AndroidMechanic, Apoorv Parmar, BadCash, B-GangsteR, Daniel Nugent, g4s8, Hiren Patel, JonasCz, Lazai, Lucas Paolillo, Michael Spitsin, Nougat Lover, rakeshdas, Vinícius Barros
41	Benachrichtigungen	alexey polusov, bricklore, Da-Jin C, Daniel Nugent, Dus, gbansal, Jeeter, piotrek1543, RediOne1, Rupali, TR4Android, weston
42	Benachrichtigungskanal Android O	Lokesh Desai

43	Benutzerdefinierte Ansichten erstellen	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
44	Benutzerdefinierte Schriftarten	Daniel Nugent, Erik Ghonyan, Gabriele Mariotti, Hiren Patel, honk, kit, Nougat Lover, Simon Schubert, Stanojkovic, Sujith Niraikulathan
45	Berechnete Ansichtsmaße erhalten	mnoronha, stkent
46	Berühren Sie Ereignisse	Yvette Colomb
47	Bibliotheksdolch 2: Abhängigkeitsinjektion in Anwendungen	Er. Kaushik Kajavadara, honk
48	Bildansicht	Ahmad Aghazadeh, Ali Sherafat, Chip, Daniel Nugent, DanielDiSu, Dinesh, Gabriele Mariotti, Harish Gyanani, kit, lax1089, Pratik Butani, Squidward, Sup
49	Bildkompression	Hiren Patel, mnoronha
50	Bitmap-Cache	Lokesh Desai
51	Bitmaps effektiv laden	iDevRoids
52	Bluetooth Low Energy	Roberto Betancourt
53	Bluetooth und Bluetooth LE API	antonio, Jon Adams, Lukas, Myon, Pavel Durov, R. Zagórski, Reaz Murshed, V-PTR, WMios
54	BottomNavigationView	Abdul Wasae, Daniel Nugent, Gabriele Mariotti, guik, Pankaj Kumar, Pratik Butani, Priyank Patel, RamenChef, rciovati, Stephen Leppik, sud007
55	Buttermesser	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
56	CardView	Carlos, Dan, Er. Kaushik Kajavadara, Gabriele Mariotti, Kaushik, Nougat Lover, RamenChef, S.R, Sneh Pandya, Somesh Kumar, Stephen Leppik, sud007, WarrenFaith, Yury Fedorov

57	Chipkarte	shadygoneinsane
58	CleverTap	Jordan, judepereira
59	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
60	ConstraintSet	Pratik Butani
61	CoordinatorLayout und Verhalten	Adarsh Ashok, Gabriele Mariotti, honk, RamenChef, Stephen Leppik
62	Countdown-Timer	privatestaticint
63	Crash-Reporting-Tools	Ajit Singh, Charuක, Ekin, Gabriele Mariotti, Ishita Sinha, Jason Bourne, Madhukar Hebbar, pRaNaY
64	Datei in Android entpacken	Arth Tilva, Daniel Nugent, mnoronha
65	Datenbindungsbibliothek	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
66	Datensynchronisation mit dem Sync Adapter	Arpit Gandhi, mnoronha
67	Datenverschlüsselung / Entschlüsselung	honk, HoseinIT, Robert
68	Datums- und Uhrzeitauswahl	adalPaRi, Brenden Kromhout, Daniel Nugent, Harish Gyanani, Ironman, Milad Nouri, RediOne1, Rohan Arora
69	DayNight-Theme (AppCompat v23.2 / API 14+)	Ishita Sinha
70	Definieren Sie den Schrittwert (Inkrement) für die benutzerdefinierte RangeSeekBar	Romu Dizzy
71	Designmuster	Adhikari Bishwash, honk, Steve.P

72	Dialog	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
73	Die Manifestdatei	John Snow, Jon Adams, kit, mayojava, Menasheh
74	Dolch 2	Aurasphere, Cabezas, David Medenjak, EpicPandaForce, honk, mattfred, Tomik
75	Doze-Modus	Daniel Nugent, Fabio, honk, NitZRobotKoder, RamenChef , Rosário Pereira Fernandes, Rupali
76	Drawables	alanv, B001, Daniel Nugent, Greg T, Hiren Patel, Jinesh Francis, Nick Cardoso, TR4Android
77	E-Mail-Bestätigung	Hiren Patel, honk, iravul, Nicolas Maltais
78	Emulator	Ahmad Aghazadeh, Dan Hulme, fyfyone Google, honk, rekire, Rubin Nellikunnathu, ThomasThiebaud
79	Erkennen Sie ein Shake-Ereignis in Android	N-JOY, tynn, Xiaozou
80	Erste Schritte mit OpenGL ES 2.0+	MarGenDo
81	Erstellen Sie benutzerdefinierte Android-ROMs	honk, Pradumn Kumar Mahanta
82	Erstellen Sie Ihre eigenen Bibliotheken für Android-Anwendungen	EpicPandaForce, honk, mnoronha
83	Erstellen von Overlay-Fenstern (immer im Vordergrund)	honk, mnoronha, NitZRobotKoder, Rupali, Sujith Niraikulathan
84	ExoPlayer	Hamed Gh
85	Facebook SDK für Android	Akeshwar Jha, AndiGeeky, Community, Daniel Nugent, honk, Zarul Izham
86	Faden	Daniel Nugent, PRIYA PARASHAR, RamenChef
87	Farbe	Nicolas Maltais
88	Farben	Carlos Borau, Dalija Prasnikar, Daniel Nugent, Erfan

		Mowlaei, Jon Adams, N J, Sujith Niraikulathan
89	Fastjson	KeLiuyue
90	FileIO mit Android	h22, sun-solar-arrow
91	FileProvider	Joost Verbraeken, pedros
92	Fingerprint-API in Android	Doron Yakovlev-Golani, RamenChef, user01232
93	Firebase	Albert, AndiGeeky, AndroidMechanic, Chintan Soni, Cows quack, Daniel Nugent, Egek92, Gabriele Mariotti, krunal patel, Leo, Omar Aflak, ppeterka, RamenChef, Saeed-rz, Sanket Berde, shahharshil46, Sneh Pandya, Stephen Leppik, sukumar
94	Firebase Cloud Messaging	Gabriele Mariotti, shikhar bansal, Shubham Shukla, Zarul Izham
95	Firebase Echtzeitdatenbank	Aawaz Gyawali, drulabs, Gabriele Mariotti, honk, Md. Ali Hossain, RamenChef, Sneh Pandya, Stephen Leppik, yennsarah, Zarul Izham
96	Firebase-Absturzberichterstattung	AndiGeeky, Gabriele Mariotti, honk, RamenChef, Stephen Leppik, Zarul Izham
97	FloatingActionButton	Ahmad Aghazadeh, Charuක, Daniel Nugent, Gabriele Mariotti, mattfred, RamenChef, Shinil M S, Stephen Leppik
98	Fortschrittsanzeige	Gabriele Mariotti, Hiren Patel, mpkuth, Sanoop, shtolik
99	Fragmente	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
100	Freigegebene Elementübergänge	noongiya95
101	Fresko	Alexander Oprisnik, Daniel Nugent, honk, Nilesh Singh, Zarul Izham
102	FuseView zu einem Android-Projekt hinzufügen	Tudor Luca
103	Fusselwarnungen	ben75, Daniel Nugent, Gabriele Mariotti, GensaGames, R.

		Zagórski, rekire, SuperBiasedMan
104	Gemeinsame Einstellungen	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
105	Genymotion für Android	Atef Hares, Harish Gyanani
106	Geräteanzeige-Metriken	Daniel Nugent, Hiren Patel, Talha, W3hri
107	Gestenerkennung	mpkuth
108	Geteilter Bildschirm / Multi-Screen-Aktivitäten	Vishal Puri
109	Gleiten	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
110	Google Awareness-APIs	Dus, honk, Willie Chalmers III
111	Google Drive API	Christlin Joseph, honk
112	Google Maps API v2 für Android	AL., AndroidMechanic, antonio, Aryan, BadCash, Charuක, CptEric, Daniel Nugent, Hiren Patel, jgm, Mina Samy, narko, Onik, Pablo Baxter, RamenChef, Stephen Leppik, stkent, sukumar, Suresh Kumar, Vasily Kabunov
113	Google Play Store	dakshbhatt21, Daniel Nugent, reVerse
114	Google-Anmeldung integrieren	AndiGeeky, RamenChef, Tot Zam
115	Google-Anmeldungsintegration auf Android	jagapathi

116	Gradle für Android	4444, Aaron He, Abdul Wasae, abhi, Abhishek Jain, Ahmad Aghazadeh, Alex T., AndroidMechanic, AndroidRuntimeException, Anirudh Sharma, Ankit Sharma , Arpit Patel, auval, Bartek Lipinski, Ben, Brenden Kromhout, bwegs, cascal, cdeange, Charuක, ChemicalFlash, cricket_007, Daniel Nugent, enrico.bacis, Eugen Martynov, Fabio, Floern, Florent Spahiu, Gabriele Mariotti, hankide, Ibrahim, Ichthyocentaurs, Irfan, jgm, k3b , Kevin Crain, kevinpelgrims, Matt, mshukla, N J, Pavel Strelchenko, Pavneet_Singh, R. Zagórski, RamenChef, rciovati, Reaz Murshed, rekire, Revanth Gopi, Sneh Pandya, sun-solar-arrow, ThomasThiebaud, ɬolɐəz əɥɬ qoq, Vlonjat Gashi, Yassie, yuku, Yury Fedorov
117	GreenDAO	Allan Pereira, Carl Poole, Grundy, MiguelHincapieC, R. Zagórski, RamenChef, Stephen Leppik
118	GreenRobot EventBus	CaseyB, Daniel Nugent, Hamed Momeni, RamenChef
119	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
120	Handler	Daniel Nugent, Floern, Hasif Seyd, Lewis McGeary, Mike Scamell, Muhammed Refaat, Sweeper, Tomik, TR4Android
121	Hardwaretastenereignisse / Absichten (PTT, LWP usw.)	JensV
122	HTTP-Verbindung	Aleks G, Daniel Nugent, Duan Bressan, honk, KDeogharkar, marshmallow, Shantanu Paul, Simone Carletti
123	Im Play Store veröffentlichen	Carlos Borau, Fabio, mnoronha, Zoe
124	Imbissbude	AndroidRuntimeException, Charuක, Daniel Nugent, Gabriele Mariotti, Harsh Pandey, Jinesh Francis, Lithimlin, marshmallow, Mike Scamell, miss C, Mochamad Taufik Hidayat, Patrick Dattilio, Piyush, RamenChef, Rasoul Miri, Rosário Pereira Fernandes, Sneh Pandya, Stephen Leppik, Zarul Izham

125	Implizite Absichten	Blundering Philosopher, Daniel Nugent, mnoronha, Pratik Butani, SoroushA
126	In-App-Abrechnung	Hussein El Feky, Pro Mode, Zoe
127	Indizierung der Firebase-App	shalini, tynn
128	Inhalt Anbieter	Andrew Siplas, cdeange, Daniel Nugent, Dinesh Choudhary, Lewis McGeary, RamenChef
129	Integrieren Sie OpenCV in Android Studio	MashukKhan, RamenChef, ssimm
130	IntentService	Anax, Daniel Nugent, honk, JonasCz, TRINADH KOYA, Yashaswi Maharshi
131	Inter-App-UI-Tests mit UIAutomator	Timo Bähr
132	Internationalisierung und Lokalisierung (I18N und L10N)	Ankur Aggarwal
133	Jackson	KeLiuyue
134	Java auf Android	Eugen Martynov
135	JCodec	Adhikari Bishwash
136	Jenkins CI-Setup für Android-Projekte	honk, Ichthyocentaurs
137	JSON in Android mit 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
138	Kamera 2 API	ChemicalFlash, devnull69, RamenChef, webo80
139	Kamera und Galerie	Ahmad Aghazadeh, carvaq, Daniel Nugent, Hiren Patel, johnrao07, RediOne1, Squidward, Yasin Kaçmaz
140	Konten und	gaara87, systemovich

	AccountManager	
141	Kontext	Will Evers
142	Konvertieren Sie den vietnamesischen String in den englischen Android-String	1SStorm
143	Konvertierung von Sprache in Text	Hitesh Sahu, honk, RamenChef, Stephen Leppik
144	Lader	chandsie, g4s8, jefry jacky, Marcus Becker, RamenChef, Stephen Leppik
145	Laufzeitberechtigungen in API-23 +	Ahmad Aghazadeh, AndroidMechanic, AndroidRuntimeException, Buddy, Daniel Nugent, Erik Minarini, Floern, Gubbel, honk, Jaseem Abbas, Kayvan N, Lewis McGeary, Luksprog, Madhukar Hebbar, nagyben, null pointer, Olu, Pavneet_Singh, Piyush, Prakash Gajera, RamenChef, RediOne1, Vivek Mishra, yuku, Yvette Colomb
146	Layouts	a.ch., Adarsh Ashok, Adinia, AesSedai101, Ahmad Aghazadeh, Aleksandar Stefanović, ankit dassor, Aurasphere, Bartek Lipinski, Björn Kechel, bjrne, Brenden Kromhout, Charuᔕ, 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
147	Leakcanary	Rakshit Nawani, tynn
148	Leinwandzeichnung mit SurfaceView	davidgiga1993
149	Leistungsoptimierung	honk, Jonas Köritz
150	Listenansicht	A.A., brainless, Daniel Nugent, Diti, Douglas Drumond, Fabian Tamp, Gabriele Mariotti, Hiren Patel, Mr.7, Ruben Pirotte, Saeed-rz, shaonAshraf, Squidward
151	Lokalisiertes Datum / Uhrzeit in Android	Geert, honk, mnoronha

152	Lokalisierung mit Ressourcen in Android	AndroidMechanic, electroid, Fabio, Gubbel, Harish Gyanani, honk, Jinesh Francis, mpkuth, RamenChef, USKMobility
153	Looper	tynn
154	LruCache	Daniel Nugent, honk, LordSidious, RamenChef, Stephen Leppik
155	Material Design	Akash Patel, Aleksandar Stefanović, Alex Chengalan, AndroidMechanic, Anirudh Sharma, ankit dassor, Bartek Lipinski, Bulwinkel, cascal, Charuක, dakshbhatt21, Dan Hulme, Daniel Nugent, dev.mi, Eixx, fyfyone Google, Gabriele Mariotti, Gal Yedidovich, Guillermo García, honk, Ibrahim, Ichigo Kurosaki, Ishita Sinha, Jaiprakash Soni, jlynch630, Jon Adams, Lewis McGeary, Lucas Paolillo, Machado, mahmoud moustafa, Marina K., MathaN, Max, Menasheh, mmBs, mpkuth, N J, Nikita Kurtin, noongiya95, oshurmamadov, pavel163, Piyush, Pravin Sonawane, Rajesh, RamenChef, rciovati, Reaz Murshed, RediOne1, ridsatrio, Sagar Chavada, Sanoop, sat, Saveen, Shashanth, Simo, SimplyProgrammer, Sneh Pandya, Stephen Leppik, sud007, sudo, sukumar, Uttam Panchasara, Vasily Kabunov, vguzzi, Vivek Mishra, Willie Chalmers III, X3Btel, Xaver Kapeller, Yasin Kaçmaz, Yury Fedorov
156	Media Player	Ahmad Aghazadeh, Carlos Vázquez Losada, hello_world, Makille, R. Zagórski, Redman
157	MediaSession	Disk Crasher, honk, KuroObi, RamenChef
158	MediaStore	Daniel Nugent, honk, RamenChef, Uttam Panchasara
159	Moshi	Blundell
160	Multidex- und Dex-Methodenlimit	Adarsh Ashok, Ben, bigbaldy, cdeange, Daniel Nugent, Gabriele Mariotti, Mike, Pongpat, R. Zagórski, Shirane85
161	MVP-Architektur	Atif Farrukh, Harish Gyanani, honk, Jon Adams, Magesh Pandian, N J, zmingchun
162	MVVM (Architektur)	Daniel W., RamenChef, Stephen Leppik
163	Nachrüstung2	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
164	Navigationsansicht	Adam Lear, akshay, Charuಬ, Daniel Nugent, Gabriele Mariotti, Kedar Tendolkar, petrumo, RamenChef, rekire, SANAT, Sevle, Stephen Leppik, sud007
165	OkHttp	A-Droid Tech, Daniel Nugent, Gabriele Mariotti, Gubbel, noob, Rohit Arya, Vucko, Zarul Izham
166	Okio	Adhikari Bishwash
167	Optimiertes VideoView	Chip
168	Orientierungsänderungen	EmmanuelMess, k3b, Ricardo Vieira, y.feizi
169	ORMLite in Android	Manos
170	Ort	Alex Chengalan, Aryan, BadCash, Daniel Nugent, Hiren Patel, Mahmoud Ibrahim, MidasLefko, Pablo Baxter, RamenChef, Stephen Leppik
171	Otto Event Bus	gus27, tynn
172	Paginierung in RecyclerView	Muhammad Younas
173	Paketierbar	Alex Sullivan, Andrei T, HoseinIT, Nick Cardoso
174	Paket-Manager	FredMaggiowski, Hi I'm Frogatto, Muthukrishnan Rajendran, Piyush, Squidward
175	Picasso	astuter, Brenden Kromhout, Daniel Nugent, Gabriele Mariotti, Ichthyocentaurs, LoungeKatt, Milad Nouri, once2go, oshurmamadov, Piyush, pRaNaY, Pro Mode, RamenChef, Rucha Bhatt, Sanket Berde, Shinil M S, Ufkoku, VISHWANATH N P, vrbsm, y.feizi
176	Ping ICMP	Carl Poole
177	Port Mapping mithilfe der Cling-Bibliothek in Android	Shinil M S
178	PorterDuff-Modus	Adarsh Ashok, AndroidMechanic, Knossos, PhilLab, S.D., Vasily Kabunov
179	ProGuard - Verschleiern und Verkleinern Ihres Codes	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
180	Projekt-SDK-Versionen	Arnav M., Ranveer, Tanis.7x
181	Protokollierung und Verwendung von Logcat	Adam Ratzman, akshay, Alexander Mironov, alexey polusov, Anand Singh, AndroidMechanic, astuter, auval, Daniel Nugent, Eugen Martynov, faranjit, FromTheSeventhSky, gattsbr, Jeeter, Jon Adams, Laurel, LaurentY, Manan Sharma, Mario Lenci, Piyush, pRaNaY, Pratik Butani, rekire, russt, Sujith Niraikulathan, TDG, thiagolr, Yury Fedorov, Zachary David Saunders
182	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
183	RecyclerView onClickListeners	abhishesh, Braj Bhushan Singh, Bryan, FromTheSeventhSky, fuwaneko, Gabriele Mariotti, honk, RamenChef, Smit.Satodia, Stephen Leppik
184	RecyclerView und LayoutManager	4444, BalaramNayak, Felix Edelmann, Gabriele Mariotti, Kayvan N, MidasLefko, RamenChef, Stephen Leppik
185	RecyclerView-Dekorationen	Barend, David Medenjak, Gabriele Mariotti, Muthukrishnan Rajendran, Peter Gordon, RamenChef, Stephen Leppik, Yasin Kaçmaz
186	Reich	bdash, Dan, EpicPandaForce, Hi I'm Frogatto, iurysza, null pointer, RamenChef, Stephen Leppik, sukumar
187	RenderScript	Ankit Popli, Dalija Prasnikar, Froyo, honk, Rucha Bhatt, Xaver Kapeller
188	Ressourcen	biddulph.r, Brenden Kromhout, Charuක, Dalija Prasnikar, Daniel Nugent, Floern, Gabriele Mariotti, Graham Smith, Harish Gyanani, honk, KDeogharkar, Menasheh, Nick

		Cardoso, Noise Generator, Piyush, R. Zagórski, reVerse, Tanis.7x, ThomasThiebaud, Vivek Mishra, Xavier
189	Retrofit2 mit RxJava	Anand Singh, gaara87, GurpreetSK95, Lukas, mrtuovinen, R. Zagórski, Zarul Izham
190	RoboGuice	AndroidRuntimeException, Lewis McGeary, Rajesh
191	Robolectric	Blundell, g4s8
192	Rückruf-URL	Atif Farrukh, honk, RamenChef, Stephen Leppik
193	Rückwärtskompatible Apps erstellen	Jon Adams, mnoronha, RamenChef, SoroushA
194	Rundfunkempfänger	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
195	Schnelle Möglichkeit, Retrolambda auf einem Android-Projekt einzurichten.	anatoli, Md. Ali Hossain
196	Schnittstellen	appersiano, Brenden Kromhout, Daniel Nugent, RediOne1
197	Screenshots aufnehmen	Ayush Bansal, Daniel Nugent, honk, Onik, sushant kumar, W0rmH0le
198	SearchView	Daniel Nugent, Dmide, Hiren Patel, RamenChef, Stephen Leppik, sud007
199	SensorManager	honk, Simon, TDG
200	ShortcutManager	g4s8, Sukrit Kumar
201	Sichere SharedPreferences	Christlin Joseph
202	Sicherheit	xDragonZ
203	Singleton-Klasse für Toast-Nachricht erstellen	Emad, Ishan Fernando
204	So bewahren Sie Passwörter sicher auf	honk, Jaggs
205	Sofortiges Ausführen in Android Studio	AndroidMechanic, Daniel Nugent, ridsatrio, Zoe

206	SpannableString	S.R
207	Speicherlecks	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
208	Speichern von Dateien im internen und externen Speicher	Amit Vaghela, Andrew Brooke, AnV, Daniel Nugent, Gabriele Mariotti, Nickan B, Uttam Panchasara
209	Speisekarte	Bhargavi Yamanuri, Chip, Daniel Nugent, Hi I'm Frogatto, honk, Iman Hamidi
210	Spinner	AndroidMechanic, Anonsage, Daniel Nugent, Vishwesh Jainkuniya
211	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
212	Startbildschirm erstellen	honk, Kiran Benny Joseph, Zoe
213	Strict Mode Policy: Ein Werkzeug, um den Fehler in der Kompilierzeit zu erkennen.	Shekhar
214	SyncAdapter mit periodischer Synchronisierung der Daten	Bhargavi Yamanuri
215	Systemzeichensatznamen abrufen und Schriftarten verwenden	Adil Saiyad, honk
216	TabLayout	Daniel Nugent, Willie Chalmers III
217	Tastatur	Hiren Patel, Kayvan N
218	Taste	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
219	Telefonnummern mit Muster formatieren.	Pedro Varela
220	TensorFlow	Pratik Butani
221	Testen der Benutzeroberfläche mit Espresso	Daniel Nugent, Gabriele Mariotti, Jason Robinson, Michael Vescovo, Milad Faridnia, N J, RamenChef, Víctor Albertos
222	Text bearbeiten	Daniel Nugent, Gabriele Mariotti, Kaushik NP, Muthukrishnan Rajendran, Rubin Nellikunnathu, Yousha Aleayoub
223	Text zu Sprache (TTS)	Ahmad Aghazadeh, honk, Jordan, Lukas, nibarius, Peter Taylor, RamenChef, Stephen Leppik
224	Textansicht automatisch anpassen	honk, Priyank Patel
225	TextInputLayout	Adarsh Ashok, BrickTop, Gabriele Mariotti, Hi I'm Frogatto, RamenChef, Shashanth, Sneh Pandya, Stephen Leppik
226	Textvorschau	Beena, Daniel Nugent, Eyad Mhanna, gaara87, Gabriele Mariotti, Hiren Patel, honk, keno, Michele, Sohail Zahid, Sujith Niraikulathan, sun-solar-arrow
227	Thema, Stil, Attribut	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
228	Toast	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
229	TransitionDrawable	S.R, Yogesh Umesh Vaity
230	Twitter-APIs	Mahmoud Ibrahim
231	Typedef-Anmerkungen:	Gabriele Mariotti, hardik m, mmBs, Pongpat

	@IntDef, @StringDef	
232	Überholspur	Gokhan Arik
233	Überprüfen Sie die Datenverbindung	sukumar, Suresh Kumar
234	Überprüfen Sie die Internetverbindung	AndiGeeky, Bill, Daniel Nugent, gbansal, Ichigo Kurosaki, Jon Adams, sukumar, TameHog, Yousha Aleayoub
235	UI-Lebenszyklus	Daniel Nugent, Dinesh Choudhary, Floern, Lewis McGeary, orelzion, R. Zagórski, Sergey Glotov
236	UI-Tests schreiben - Android	Atif Farrukh, Daniel Nugent, Gabriele Mariotti, honk, Jon Adams, originx
237	Umgang mit Berührungs- und Bewegungsereignissen	honk, Zoe
238	Umgang mit Deep Links	Doron Yakovlev-Golani, Harsh Sharma, mnoronha, Tanis.7x
239	Unit-Tests in Android mit JUnit	abhi, Andre Perkins, AndroidMechanic, Eugen Martynov, honk, Lewis McGeary, N J, Namnodorel, Patrick Dattilio, Rolf ツ
240	Universal Image Loader	Greg T, honk, Jon Adams, priyankvex, Stephen Leppik
241	Untere Blätter	Daniel Nugent, Gabriele Mariotti, Magesh Pandian, MiguelHincapieC, RamenChef, Stephen Leppik, sud007, Zarul Izham
242	Unterschreiben Sie Ihre Android App zur Veröffentlichung	Gabriele Mariotti, M M
243	Unterstützende Bildschirme mit unterschiedlichen Auflösungen, Größen	Eduardo, Guilherme Torres Castro, kalan, mpkuth, Onur, ppeterka
244	VectorDrawable und AnimatedVectorDrawable	Ahmad Aghazadeh, Aleksandar Stefanović, gaara87, honk , Lewis McGeary, RamenChef, Stephen Leppik
245	Vektor Drawables	Priyank Patel, ShahiM
246	Verbesserung der Android-Leistung mithilfe von Symbol-Schriftarten	Beto Caldas, honk, Neeraj

247	Veröffentlichen Sie die .aar-Datei mit Gradle in Apache Archiva	Marian Klühspies
248	Veröffentlichen Sie eine Bibliothek in Maven Repositories	Farid
249	Vibration	cdeange, Zerntrino
250	VideoView	iravul, Sashabrava
251	ViewFlipper	Anita Kunjir, Daniel Nugent, honk
252	ViewPager	Adarsh Ashok, Adrián Pérez, Daniel Nugent, Gabriele Mariotti, Moustachauve, RamenChef, RediOne1, Rucha Bhatt, Sneh Pandya, Stephen Leppik, Usman, ZeroOne
253	Volley	2943, Ankur Aggarwal, Endzeit, Harsh Dalwadi, herrmartell, honk, Jon Adams, Pablo Baxter, RamenChef, Rubin Nellikunnathu, Rucha Bhatt, sameera lakshitha, Stephen Leppik, VISHWANATH N P
254	Was ist ProGuard? Was ist die Verwendung in Android?	Ayush Bansal, Daniel Nugent, Ghanshyam Sharma, Pratik Butani
255	WebView	Amod Gokhale, Daniel Nugent, g4s8, j2ko, jasonlam604, JonasCz, Mohammad Yahia, ppeterka, Prakash Bala, shtolik, Squidward, Sukrit Kumar, sukumar
256	Werkzeugeigenschaften	Dalija Prasnikar, Gabriele Mariotti, Harsh Sharma, Kayvan N, TR4Android
257	Widgets	4444, Alex Ershov, Daniel Nugent, Don Chakkappan, Imdad, nenofite, sun-solar-arrow
258	Wie verwende ich SparseArray?	honk, Robert Banyai
259	Wi-Fi-Verbindungen	4444, AndroidMechanic, Daniel Nugent, gus27
260	XMPP-Register Anmelden und einfaches Beispiel	4444, RamenChef, Saveen
261	Xposed	Medusalix
262	Youtube-API	abhishesh, Giannis, honk, MashukKhan, Zarul Izham, Zeeshan Shabbir

263	Zeichenketten formatieren	Beena, Daniel Nugent, gaara87, Greg T, Michele, RamenChef, Suresh Kumar
264	Zeit Utils	Burhanuddin Rashid, Mukesh Kumar Swami, Muthukrishnan Rajendran
265	ZIP-Datei in Android	Adnan
266	Zugriff auf SQLite-Datenbanken mithilfe der ContentValues-Klasse	Adil Saiyad, emecas, honk
267	Zum Aktualisieren wischen	Chirag SolankI, Daniel Nugent, Gabriele Mariotti, Malek Hijazi, RamenChef, Stephen Leppik